SEARCHABLE PUBLIC KEY ENCRYPTION

FINAL REPORT

SHORT TERM RESEARCH PROJECT
(304/PKOMP/636030)

PROJECT LEADER:
PROF MADYA DR. AZMAN SAMSUDIN

DATO' PROF. MUHAMMAD IDIRIS SALEH
Timbalan Naib Canselor
(Penyelidikan & Inovasi)

COMPREHENSIVE TECHNICAL REPORT

Table of Content:

Abstract:

Encrypted data are being kept in remote server for purposes like backup and space savings. In order to retrieve these encrypted data, efficient search methods were proposed that enable the retrieval of the dataset without leaking too much information thus ensuring better security and less information leakage.

Some of the current searching methods were proposed by Song, Wagner and Perrig (SWP) which is SWP Linear Scan and SWP Encrypted Index [1]. SWP Linear Scan is a method that encrypts the words in the document one by one and then sent to the server for safekeeping. To retrieve the document, a search is performed on each of the encrypted word until a match is found. SWP Encrypted Index was proposed later on to speed up the search where selected keywords are chosen and encrypted with a list of pointer to documents that contains the keywords.

In the year of 2003, Eu-Jin Goh proposed a new search method that utilizes bloom filter hash coding method with allowable error by Burton B. Bloom [3]. This search method is referred as Goh Bloom Filter [2], where each document will have a set of keyword that is hash coded into a bloom filter. Each document will be link to a bloom filter where the search is done on the bloom filter for a matched keyword.

An improved method is proposed here-on for an efficient search on encrypted data which implements a keyword list in a hash table for each encrypted document. The keyword is encrypted in such a way that by providing the file server with required search information known as "a capability for a certain keyword" [11], searches can be performed without leaking any information.

Changes to the way of encrypting the keyword list in the improved method allows the usage of different hash techniques like Pearson Perfect Hash Function [10] for keeping the keyword list enables fast searches and space savings. The actual data is unhampered thus not limiting the data type to only text document and can be compressed and encrypted with any method of the users' choice.

Keywords: Search, Encrypted Data, Bloom Filter, Linear Scan, Encrypted Index

1 *Introduction*

As we advance into the digital age, more and more information are stored in computers. These data are becoming much increasingly important as it consists either personal details, money account or technology researches. To thwart people from reading the contents of the information stored, encryption is introduced where the owner have the 'key' that allows the accessing of the information. These encrypted data is stored in a database for safekeeping. In order to retrieve the information, the owner will have to select the correct file and decrypt it. As the amount of documents grow, it would not be feasible to decrypt all documents to find the needed document. Furthermore, if the encrypted data is kept in an untrusted storage on a different location, it would be unwise to decrypt the data. Therefore a search method is needed to find the needed document without decrypting first to ensure better security and less information leakage.

Due to this an efficient search method in getting the correct encrypted document based on certain keyword by the user is needed. This saves time involving in decrypting the documents and does not leak any information on the untrusted storage area.

2 *Literature Survey*

Song, Wagner and Perrig (SWP) [1] presented two methods of searching encrypted data, which are Linear Scan and Encrypted Index method. The first method will be known as SWP Linear Scan (Section 2.1) while the later as SWP Encrypted Index (Section 2.2). Other than SWP methods, Goh[2] presented a method that uses Bloom Filter[3] to search on encrypted data. This method will be known as Goh Bloom Filter (Section 2.3).

2.1 SWP Linear Scan

In the paper by SWP, four schemes were introduced as proof of concept for SWP Linear Scan. The four schemes; Basic Scheme (Section 2.1.1), Controlled Searching (Section 2.1.2), Hidden Searches (Section 2.1.3), Final Scheme (Section 2.1.4) will be studied in detail below

2.1.1 Basic Scheme

In this scheme, we have Alice, owner of a set of documents D represented as $d_1, d_2, \ldots, d_t$ where t represents the number of documents and Bob the owner of the file server FS where the documents are to be kept. Before Alice gives the document to Bob, the document is arranged in a sequence of words $W_1, W_2, \ldots, W_l$ where l is the number of words in a document. Each of this word will be allocated a fixed length of n bits. The encrypted document that is sent to Bob is derived from an XOR function of each word W_i with another fixed length random bit array T_i for every position i in the document. The result of the XOR function will be the cipher text $C_i = W_i \oplus T_i$. See Appendix A on "Mathematic of XOR Function".

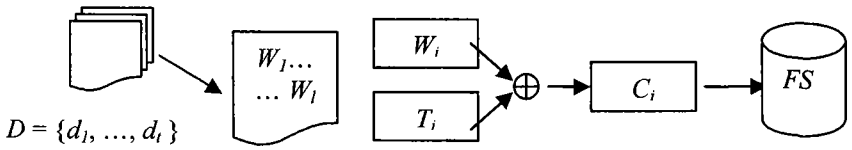


Figure 1: Basic Scheme

In order to generate the T_i inputs for the XOR function, a stream of pseudorandom bits $S_1, S_2, \ldots, S_l$ where l is the number of words in the document will be generated from a pseudorandom number generator G_i with a secret seed. These pseudorandom bits are n - m bits long. See Appendix B on "Mathematic of Pseudorandom Number Generator". The pseudorandom bit S_i will act as an input for a function F with a key k_i to generate the rest of the m bits. The key k_i used here can be same or different for all i position. Both the combined pseudorandom n - m bits S_i and the generated m bits $F_{ki}(S_i)$ will serve as the input of the XOR function $T_i := \langle S_i , F_{ki}(S_i) \rangle$ (Figure 2).

With the generated input T_i and word W_i, Alice can now XOR every word in the document and sent the list of cipher text C_i to Bob for safekeeping. The same process is done to every document in set D.

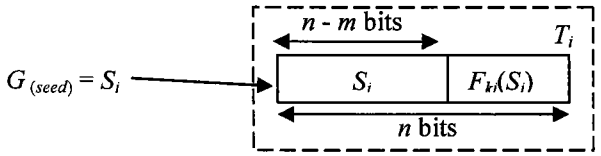


Figure 2: Value of Ti

2.1.1.2 *Basic Scheme Searching Technique*

Now that the documents are with Bob, Alice would like to find documents that contain a certain word W_i that Alice wants. Here, Alice will send the key k_i with the word W_i where Bob can do an XOR function to get the value of $T_i = W_i \oplus C_i$ where $T_i := \langle L_i , R_i \rangle$ and $L_i = S_i$; $R_i = F_{ki}(S_i)$. By using the key k_i with the function F, Bob will be able to check whether $F_{ki}(S_i)$ equals to R_i, the m bits of the cipher text block (Figure 3).

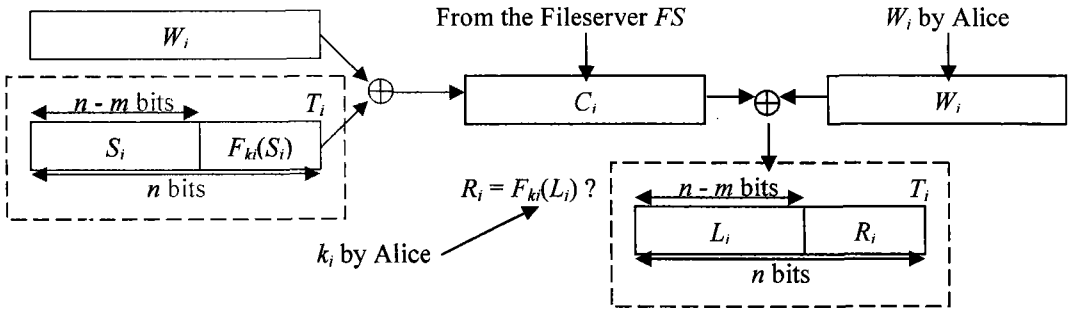


Figure 3: Basic Scheme Searching Technique

However, SWP found that this scheme has the problems as below to Alice

1. Alice has to disclose the word W_i that she wants. This allows Bob to know the word and know which documents contain W_i. (*Problem 1*)
2. Alice have to provide all the key k_i for Bob to be able to check $R_i = F_{ki}(L_i)$ and thus risks exposing the whole document (if k_i same for every instance of i). (*Problem 2*)
3. Knows the position i of the word W_i to provide key k_i for the comparison as not to reveal other key k_j where $j \neq i$. (if k_i different for every instance of i). (*Problem 3*)

2.1.1.3 *Basic Scheme Decryption Technique*

Searches that return the cipher text $C_1, C_2, \ldots, C_l$ will be split into two blocks, $C_{i\,(left)}$ which size is equal to L_i and $C_{i\,(right)}$ which size is equal to R_i . Using the pseudorandom number generator G with function F will generate the required $T_i := \langle L_i , R_i \rangle$ where Alice will decrypt the cipher text by applying XOR function to find two blocks of the word, $W_{i\,(left)} = C_{i\,(left)} \oplus L_i$ and $W_{i\,(right)} = C_{i\,(right)} \oplus R_i$. The original word is $W_i = W_{i\,(left)} + W_{i\,(right)}$ (Figure 4).

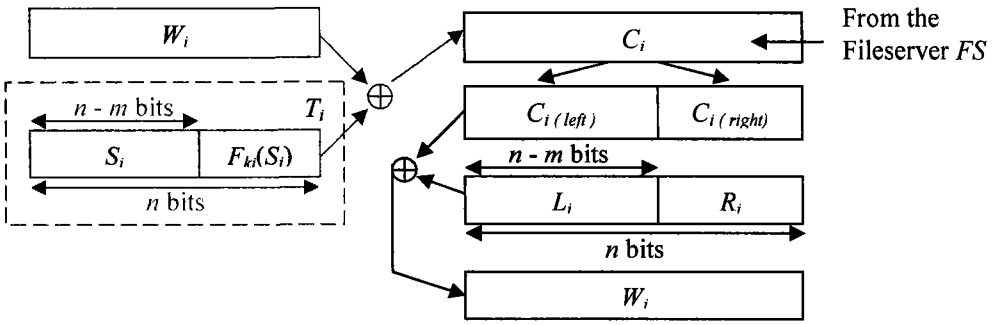


Figure 4: Basic Scheme Decryption Technique

2.1.2 *Controlled Searching Scheme*

Due to the *Problem 2* and *Problem 3* of Basic Scheme Searching Technique (Section 2.1.1.2), SWP made improvement by introducing controlled searching. Here, another function f is used to generate the value of k_i. The word W_i will be applied to the function resulting a newly generated key k_i

$= f_k(W_i)$ (Figure 5). The value of k' is kept secret from Bob, and only the generated value k_i is given to Bob. By doing this, the value k_i is independent on the position of words and thus Alice does not need to know the location of the word prior to the search. The search is performed identical to the Basic Scheme Searching Technique (Section 2.1.1.2). The values of $f_k(W)$ is dependent on word W and this allow Bob to reveal all the position i where W occurs but not other position where $W_i \neq W$ (Figure 6).

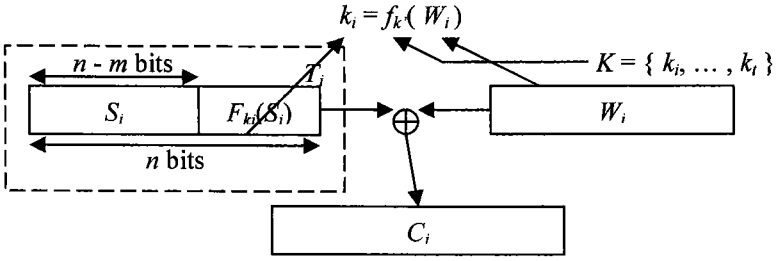


Figure 5: Controlled Searching Scheme

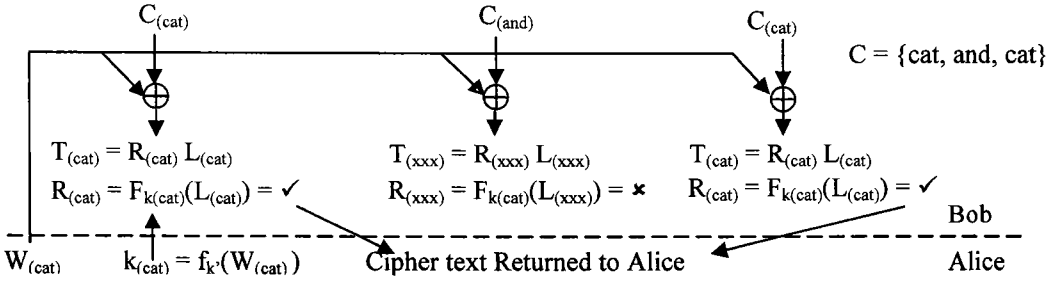


Figure 6: Controlled Searching Scheme Example

From this scheme SWP also presented further improvement of search method by changing the way key k_i is generated. Search can be focused when documents are divided into blocks like chapters. In order to group cipher text into chapters, the key is constructed by having an additional parameter, $k_i = f_{k'}(\langle C, W \rangle)$, where C is the chapter and W the word in that chapter. This controls the result of search done by Alice where only words W for chapter C_i is returned and not for other chapters C_j, $i \neq j$. The key can be created hierarchical based on the document where a key k_i can act as a key for another key such as $k'' = f_{k'}(\langle 1, W \rangle)$ be used for $k_i = f_{k''}(\langle 0, C \rangle)$. This allows word W be search in chapter C_i by revealing $k_i = f_{f_{k'}(\langle 1,W \rangle)}(\langle 0, C \rangle)$ or just $k_i = f_{k'}(\langle 1, W \rangle)$ for Bob to search in all chapters for word W.

2.1.3 *Hidden Search Scheme*

Both the Basic Scheme (Section 2.1.1) and Controlled Search Scheme (Section 2.1.2) allows Bob to know what word W that Alice is searching (Problem 1). To prevent this, hidden searches method is introduced where the W is first encrypted using a deterministic algorithm $E_{k''}$. The prerequisite in this method is that the encryption method E is not allowed to use any randomness and must rely on W only without the knowledge of position i of the word. An implementation of this scheme is to use Electronic Codebook mode (*ECB*) on the word W. For a longer document, the Cipher Block Chaining Mode (*CBC*) can be used where word W is encrypted using a constant initialization vector (*IV*) but must be same for every position.

Now Alice will take every word in the document $W_1, W_2, \ldots, W_l$ and encrypt it using the function E with a key k''. This will result in the cipher text $X_1, X_2, \ldots, X_i$ where $X = E_{k''}(W)$. The input for the XOR operation, T_i is generated with a change where encrypted word X_i is used with pseudorandom bits S_i, resulting $T_i := \langle S_i, F_{ki}(S_i) \rangle$, $k_i = f_{k'}(X_i)$.

2.1.3.1 *Hidden Search Scheme Searching Technique*

With this scheme, Alice will send the encrypted word X_i and the key $k_i = f_k(X_i)$ and sends it to Bob. This allows Bob to search for W without revealing W because it is encrypted by function E (Figure 7). This fixes the Problem 1 of Basic Scheme Searching Technique (Section 2.1.1.2).

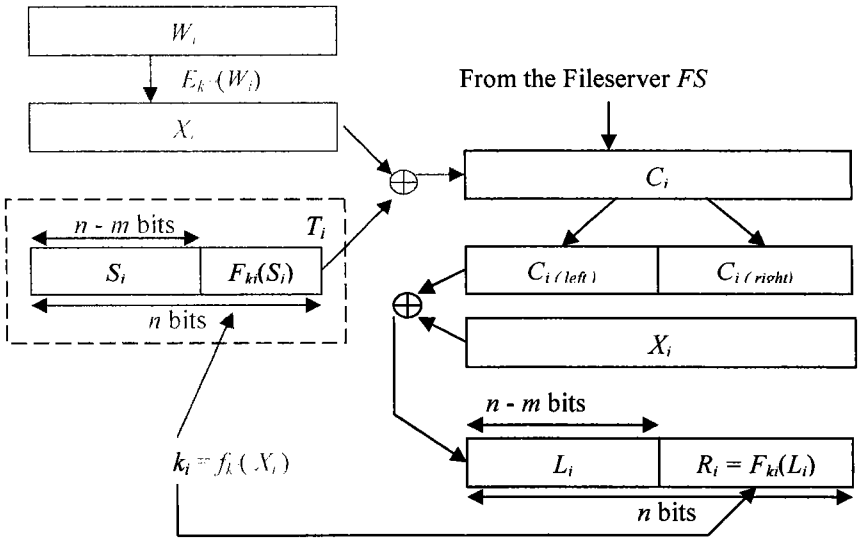


Figure 7: Hidden Search Scheme Searching Technique

2.1.4 *Final Scheme*

The Final Scheme presented by SWP allows the returned cipher text of the document to be decrypted. The words other than what Alice searched for cannot be decrypted because Alice is unable to determine the value of $k_i = f_k(E_{k''}(W_i))$ to generate the R_i m bits of T_i. Alice will not know the encrypted word $E_{k''}(W_i)$ for every position i of the document (Figure 8).

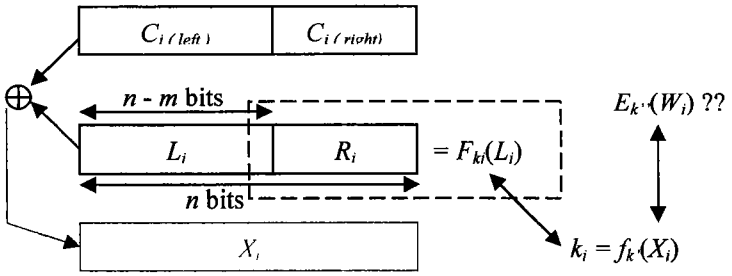


Figure 8: Final Scheme

2.1.4.1 *Final Scheme Encryption Technique*

To enable Alice to do the decryption on the document, the setup of this method has to be changed. The encrypted $E_{k''}(W_i)$ will need to be split into two blocks with the size same as T_i blocks, resulting $E_{(left)\,k''}(W_{(left)})$ with size of $n - m$ bits and $E_{(right)\,k''}(W_{(right)})$ size of m bits. Now, the key k_i generation will be on just the $n - m$ bits $k_i = f_k(E_{(left)\,k''}(W_{(left)}))$ instead of the full word W. Both this two blocks are then XOR with L_i and R_i to generated the cipher text C_i of n bits (Figure 9).

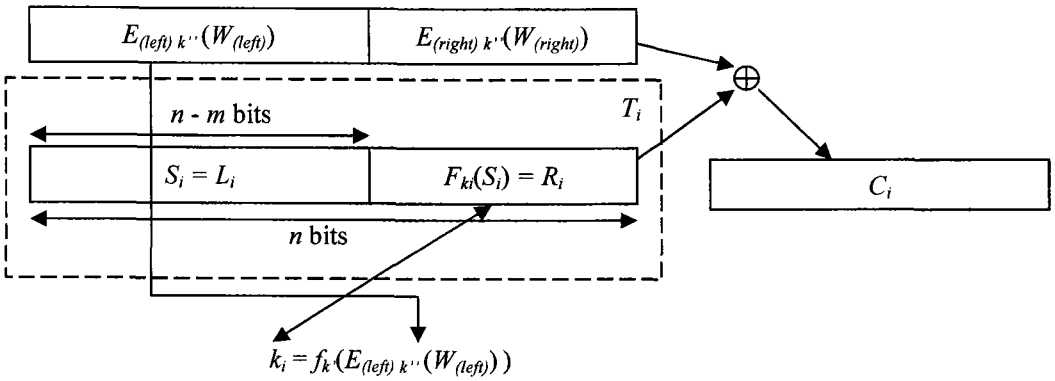


Figure 9: Final Scheme Encryption Technique

2.1.4.2 *Final Scheme Decryption Technique*

The key generation using just the encrypted word of n - m bits allows Alice to use the pseudorandom bits S_i generated by Alice to be XOR with $C_{i\ (left)}$ and thus recovering the encrypted word $E_{(left)\ k''}(W_{(left)})$ .This is then applied the key generator $k_i = f_{k'}(E_{(left)\ k''}(W_{(left)}))$ for the pseudorandom function $F_{ki}(S_i)$ to generate the remaining m bits which is cipher text $C_{i\ (right)}$. Next, the cipher text $C_{i\ (right)}$ is then XOR with the m bits to get the encrypted word $E_{(right)\ k''}(W_{(right)})$. With both the blocks of encrypted words, using the decryption function will result the original word W_i (Figure 10). All other encrypted words are decrypted using this method.

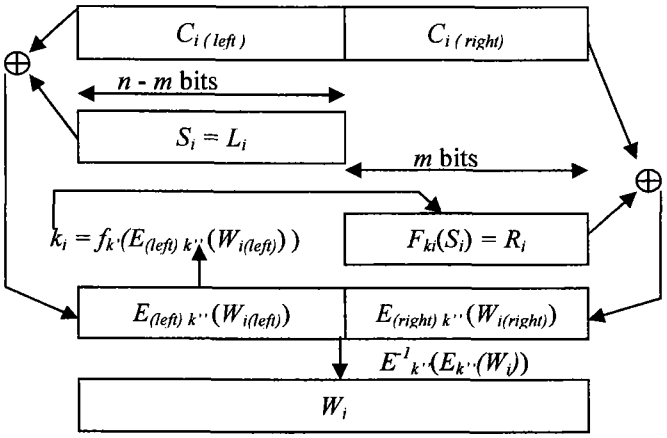


Figure 10: Final Scheme Decryption Technique

2.1.5 *SWP Linear Scan Improvements*

This scheme can be further improved to support queries as below

1. Boolean Operators ➔ W AND W_1 by providing both encrypted word and their respective keys k
2. Proximity Queries ➔ W near W_1 by searching on position i and i+1 on both words
3. Limited Wildcard Queries ➔ W = a[a-z] = {aa, ab, … az} by sending ${}^{n}C_{r}$ of words
4. Word Occurrences ➔ by providing a counter for the number of the same word W = ⟨0, cat⟩ and W_1 = ⟨1, cat⟩.

2.1.6 *SWP Linear Scan Word Length*

Other than that, this scheme can only use fixed length words. This poses a problem of having a document with every word sharing a same length and for different language. Furthermore some languages other than English might have difficulty in separating each word to be encrypted. Due to this SWP presented two varieties to support variable length words, which are SWP Padded Length Word Linear Scan (Section 2.1.6.1) and SWP Variable Length Word Linear Scan (Section 2.1.6.2)

2.1.6.1 *SWP Padded Length Word Linear Scan*

SWP proposed a method that uses a fixed size length n based on the longest word in the document. Shorter words are then padded with characters that only Alice will know. To search for the word shorter than the fixed size n, Alice will need to pad it with the character before performing the search. This method generates a bigger size document and therefore wastes space and processing time linear to the size of n.

2.1.6.2 *SWP Variable Length Word Linear Scan*

Another alternative to this method is to use a variable size length. The encrypted word will need to be attached together with the word before performing the XOR operation to create the cipher text (Figure 11).

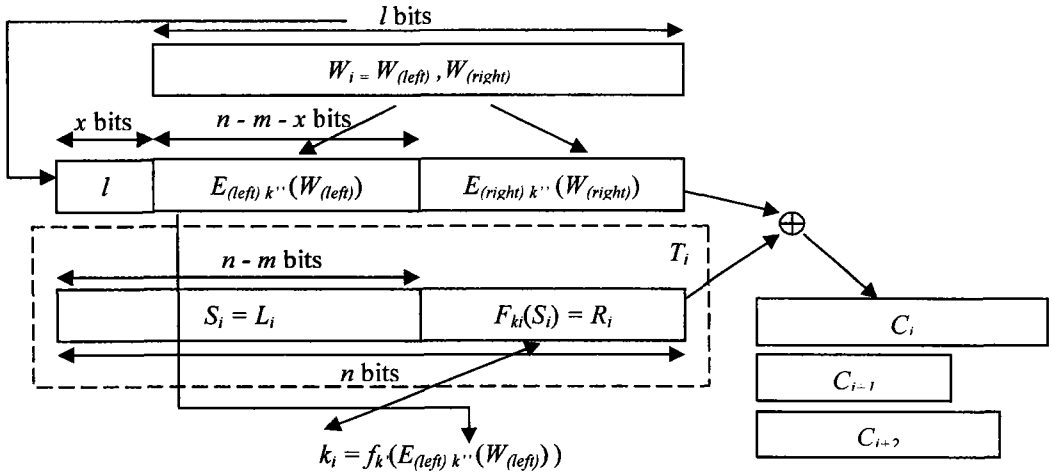


Figure 11: SWP Linear Scan Variable Word Length

The length of the words in the document needs to be kept secret from Bob in order to prevent statistical attack on the knowledge of the words' length [1] [2]. With variable length cipher text, Bob is unable to search on n bits size blocks and need to search on bits boundary instead. This increases the cost of search as it is now done in bits and not blocks while provide better space savings. The decryption method was not mentioned by SWP; a proposed method of decryption is as below (Figure 12)

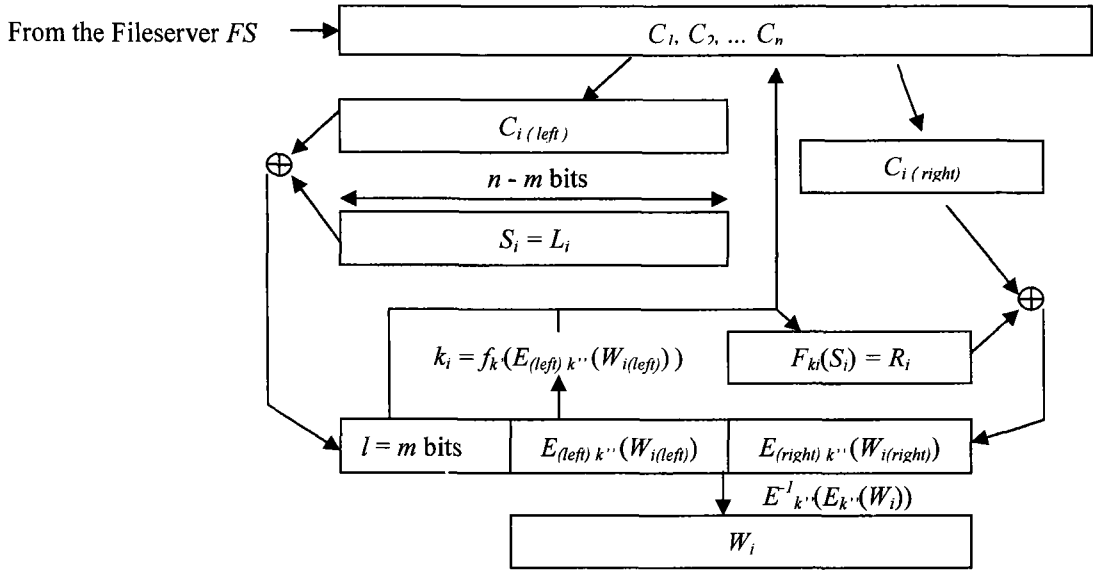


Figure 12: SWP Variable Word Length Linear Scan Decryption Technique

2.2 *SWP Encrypted Index*

2.2.1 *SWP Encrypted Index Method*

SWP proposed the use of an index to speed up the search for document based on keywords. In this method each keyword W_i is attached to a list of document pointer P where each pointer in the list p_i points to a document d_i, $p_i \rightarrow d_i$. The keywords and pointers form the rows of the index I (Figure 13).

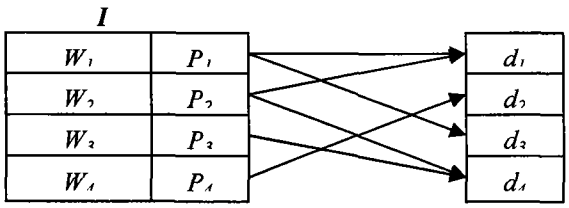


Figure 13: SWP Encrypted Index

The keyword and the document pointers in each list in the index are first encrypted. Alice will send the encrypted word $E(W_i)$ and $E(P_i)$ to Bob for safe keeping. When Alice wants to retrieve the documents, Alice will send the encrypted word $E(W_i)$ and get the returned encrypted list of pointers $E(P_i)$. With this, Alice can decrypt the encrypted list and send another request for the documents. As noted, this method will take two trips.

To save a trip, Alice can encrypt the list of document pointer in the index $E_{kp}(P_i)$ using key $kp = F_{kw}(E(W))$ related to the encrypted word. Searching can be done when Alice reveal $\{E(W), kp\}$. Bob will be able to decrypt the encrypted pointer list $E_{kp}(P_i)$ and perform another search for the document on behalf of Alice (Figure 14).

Bob can be prevented from doing a statistical analysis on the index if the list of pointers is kept in fixed size list where infrequent keywords are padded up to fixed size with false documents (document that does not contain the keyword). Common words are split into few where several search queries have to be merged and done in parallel.

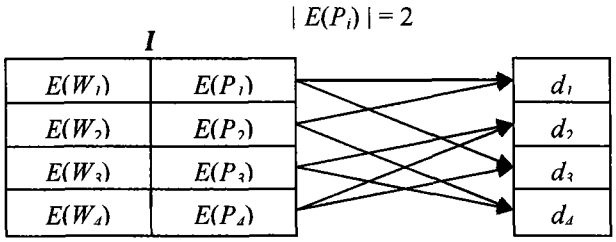


Figure 14: SWP Encrypted Index with Encryption

2.2.1 *SWP Encrypted Index Advantages and Disadvantages*

Using this method allows all documents to be searched at once, therefore making this method the best no matter how many words or documents contained in Bob's fileserver. Keywords are shared among all documents reducing the space needed.

However, SWP Encrypted Index suffers when there are changes to the document where the index has to be update as well. By observing the changes in the length of the list, Bob can deduce how many keywords are contained in the new document and the codeword for the keyword. With enough sampling, Bob can find out the values based on Goh [2]. To prevent this, Alice need to do a mass update of the pointers in the index to hide real updates. This leads to costly update on both time and workload.

2.3 *Goh Bloom Filter*

When searches are performed, efficient search can be achieved by utilizing hash table technique. The definition of hash table can be summarized as

"Hash table is a data structure that implements an associative array. Like any associative array a hash table is used to store many key - value associations (this is a many to one relationship as the hash table is almost universally smaller than the number of keys)" [9]

2.3.1 *Hash Coding*

In a hash coding method, the hash area is organized into cells where messages are stored. An iterative pseudorandom computational process is used on the messages to generate the hash address to each cell of the hash area.

2.3.2 *Conventional Hash Coding Method*

In the conventional hash coding method, each messages m_1, m_2, ..., m_n are b bits long. The hash area in this method is prepared by dividing the area into h number of cells where each is b+1 bits long, therefore the length of the cells are longer than the message. The number of cells are also more than then number of messages $| h | > | n |$. The extra bit for each cells acts as an indicator whether the cell is empty or occupied. Due to this, messages in the occupied cells are treated as first bit always being the value of 1.

The process of allocating the message is done with a pseudorandom number called hash address k that will be generated from a function F where k is in the range of the hash area cells $0 \leq k \leq h$-1. Once the hash address k is generated from the messages, the k^{th} cell in the hash area is checked to see if it is empty (first bit equals 0). If the cell is empty, the message is then stored with the additional first bit value of 1.

For occupied cells, the hash address is regenerated again until an empty cell is found for the message (Figure 15).

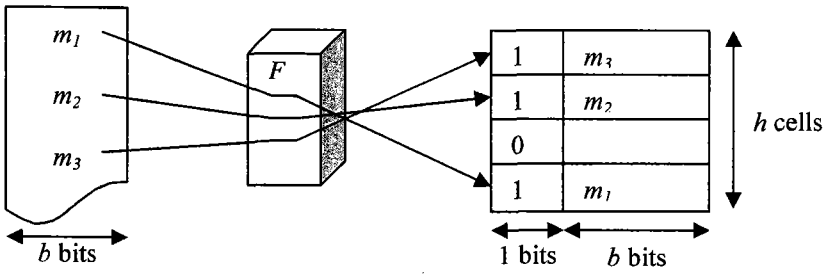


Figure 15: Conventional Hash Coding

The messages are tested by similar process of hash address creation, where the hash address k of the message to be tested is compared with the k^{th} cell in the hash area. A match indicates the test message is a member of the set. Empty cell indicates otherwise.

2.3.3 *New Hash Coding Method*

Bloom then presented two new methods, Method 1 and Method 2, which will be known as Hash Coding with Coded Message method and Bloom Filter Hash Coding method.

2.3.3.1 *Method 1 – Hash Coding with Coded Message*

The change in this method is a reduction in hash area size. In this method, the hash area is divided into cells that are smaller and only keep a code instead of the entire message like the conventional method. Here a function F: $\{0,1\}^a \rightarrow \{0,1\}^c$ where $a > c$ is used to generate a code with a length shorter than the original message. Bloom noted that the allowable fraction of error is defined as P where the range is of P is $(1 >> P >> 2^{-b})$ [2]. The size of cell c will be chosen that the fraction of error will be close to or smaller than P. The hash area is then divided into h cells based on the allowable error size of cells c. Each message are encoded into c bit using the function F. Due to shorten message code; it would not be entirely unique and allow error in testing the membership of message (Figure 16).

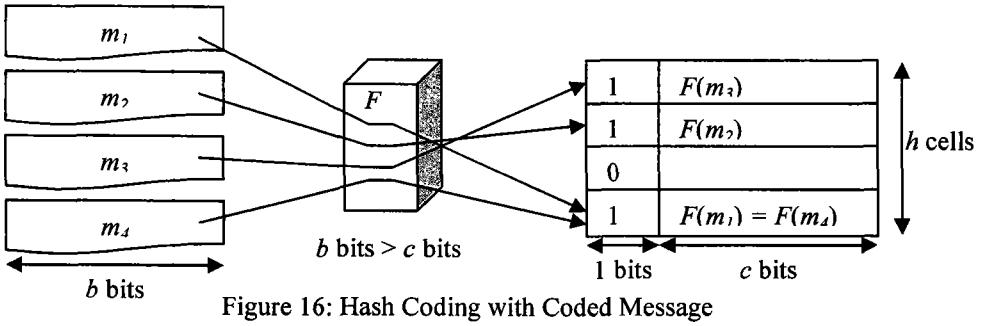


Figure 16: Hash Coding with Coded Message

2.3.3.2 *Method 2 – Bloom Filter Hash Coding*

Burton Bloom first introduced Bloom filter as a technique that test a series of messages one by one for membership in a given set of message with tradeoffs in hash coding computational factors. The factors considered here are hash area (space) and time required to identify a message as a nonmember of a given set (reject time and an allowable error frequency [3].

In this method, instead of dividing the hash area into cells, it is divided into N individual bit addresses from 0 to N-1. Here the hash area bit addresses are all set with the value of 0. Each message in the set that need to be stored is hash coded into a distinct bit address $d_1, d_2, \ldots, d_l$ where l is the bit

length of the message. The hash area bit addresses that matches the bit addresses of the message from the hash function method are set to the value of 1 (Figure 17).

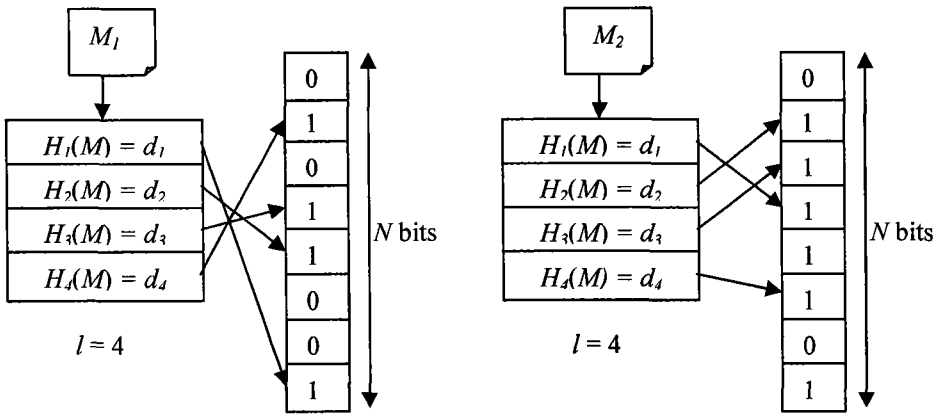


Figure 17: Bloom Hash Coding

To test whether a message belongs to the set, the message will be arranged in a sequence of l bit address d'_1 to d'_l and hash coded to the respective bit address in the hash area. The bits are tested individually with the value of the bit address in the hash area. If either one of the value of the bit address in the hash area is 0, it is proven that the message is not a member in that set (Figure 18).

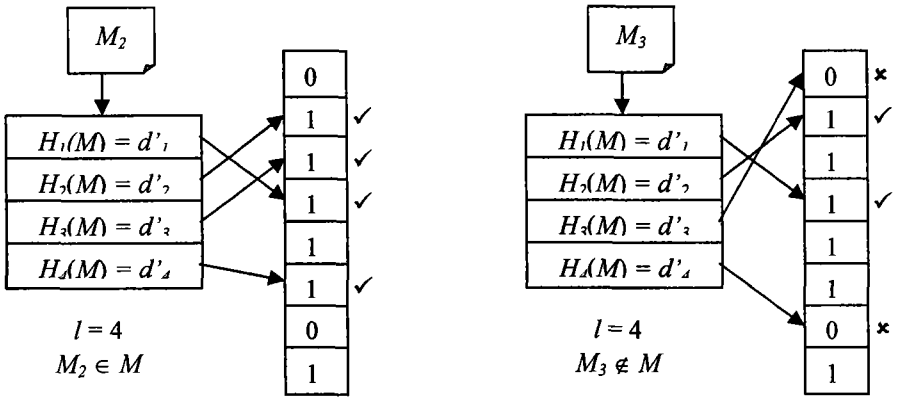


Figure 18: Bloom Hash Coding Membership Test

2.3.4 *Goh Bloom Filter Method*

Goh introduced the method that uses the bloom filter hash coding by Bloom. In this method the document D are represented with a set of words $S = \{s_1, s_2, \ldots, s_n\}$ where n is the number of words chosen by Alice. Each elements of set S represents an array of m bit. The conversion of the words in set S is done by applying r independent hash function h_1 to h_r where $h_i : \{0,1\}^* \rightarrow [1,m]$ for $1 \leq I \leq r$. For each element in S the array bits are hashed $h_1(s_i)$, $\ldots, h_r(s_i)$. The location of each distinct bit of the hashed value will set the bit address in the hash area to 1. Bit addresses with multiple set are not changed and remain the value of 1.

To determine membership of a word s_i in set S the hash value of the generated word $h_1(s_i)$, $\ldots, h_r(s_i)$ must all have the value of 1 in the hash area. Bloom filter sacrifices space and time for allowable error [3]. These allowable errors are known as false positives. False positives are words s_i that are not a member of the set S but proven by bloom filter checks as member. This is due to the bits set by a collection of other words in set S (Figure 19).

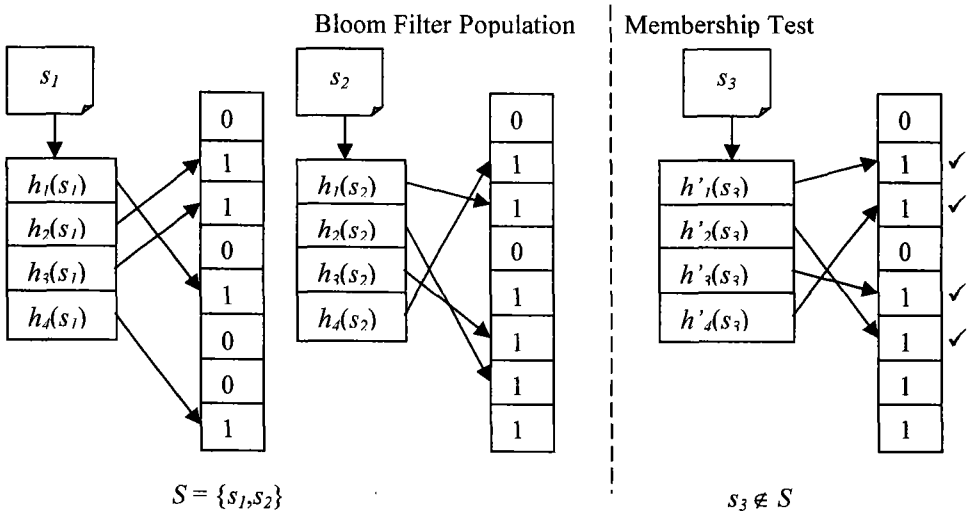


Figure 19: Goh Bloom Filter

2.3.4.1 Design of Goh Bloom Filter method

In this method, every document that Alice has will have a bloom filter that tracks the keywords for that document.

- **Preventing Statistical Analysis**

The hash function to generate the keyword digest in SWP Encrypted Index leaks a lot of information when an update is done. This applies to Goh Bloom Filter method in creating the distinct bits in the hash area. To prevent this, Goh Bloom Filter changed the method in creating the keyword digest. Goh noted that the statistical attack could be effective only if the keyword digest of the keyword remains the same for all document. To prevent this, the keyword digest of the documents will be different for each document on each update.

Here, Alice has a set of documents $D = \{d_1, d_2, \ldots, d_n\}$, where each document has a set of keywords $x_1, x_2, \ldots, x_i$. First, Alice needs to create a suitable Bloom filter with an array size of m bits and r key for the hash function. The r keys are then chosen from the key space R uniformly at random. Now the hash function will be $F = h_1(x), \ldots, h_r(x)$ where x is the keyword for a document.

The improvement here is that before the hashed value of $x_1, x_2, \ldots, x_i$ is inserted into the Bloom filter for a document d_n, it is rehashed again with the document number. The new hash function would be $F = H_{h1(x)}(j)\ H_{hr(x)}(j)$ where j is the document number acting as another 'key'. This ensures that the bit array m that is created for each document will be different although the keywords might be the same (Figure 20).

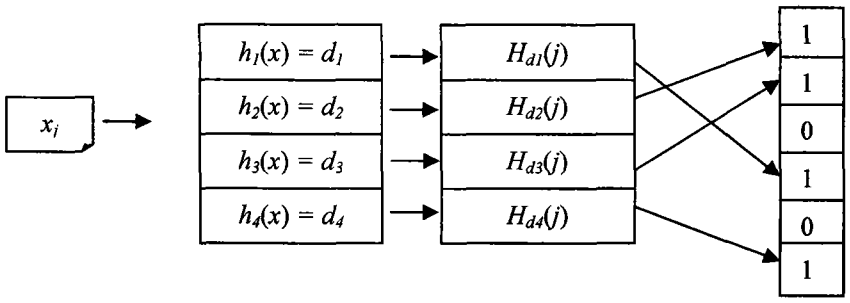


Figure 20: Goh Bloom Filter with Document Number

To search, Alice derives the keyword digest for the keyword y by applying it to the function $F = h_1(y), \ldots, h_r(y)$. The keyword digest is given to Bob where Bob will apply the document number I with the keyword digest to find the match for $F = H_{h1(x)}(i), \ldots, H_{hr(x)}(i)$. This is done all the documents until all the matches are found.

The searching for this method consists of generating the pseudorandom number from a pseudorandom function, computing the modified keyword digest and checking r location for a bloom filter match. Keywords can be combined in the search and only require one pass through the document's bloom filter. Keyword digest from F is relatively small therefore communication overhead is low.

- **Less Revealing Queries**

After queries are performed, the server will know that keyword x is contained in the documents that is returned. To make queries less revealing, Alice can try to make the queries a little more specific as in checking for word x in document I only. This can be done by giving the value of the specific hashed word with the document number $H_{h1(x)}(i), \ldots, H_{hr(x)}(i)$.

- **Boolean Queries**

Bloom filter can handle quires with Boolean command like "AND" and "OR" on multiple queries. For "AND" command, both the bits digest are check one after another with the bloom filter. This is the same with "OR" command.

- **Regular expression searches**

Expression like "*ab*[*a-z*]" can be done where the query is expanded to form each keyword from "*aba*" to "*abz*" with a single keyword query time. Expression like "*ab**" is possible but might result is an uncontrolled overgrown query size.

- **Update – Adding/Deleting/Altering documents**

Whenever Alice wishes to delete a document, she just tell Bob to delete both the document with the bloom filter. This approach does not have any information leakage and affect other document with bloom filter security. Adding a document is just creating a bloom filter, binding it to the document and sending it to Bob. Deleting a document is a constant time where a command to delete is sent to Bob. If any alteration needs to be done, a new document number is generated and a totally new bloom filter is created. The cost for this is linear to the size of the document.

2.3.4.2 Goh Bloom Filter Method Properties

- **Compressed and Encrypted Data**

Bloom filter just acts as an index to the data and this allow the data to be altered after the bloom filter is created. The documents can be compressed and encrypted by Alice before sending to Bob. Compression results a less space cost while encryption allows the desired security for the document.

- **Variable length Keyword**

By using the r key hash function to encrypt the message, it will take any length of message and output it to a constant length based on the hash function F.

- **Key management**

The keys required in this method are *r* keys, which is generated from a pseudorandom number generator with a secret seed.

- **Exact keyword location**

This method does not allow Alice to know exact locations of keywords. This is not a disadvantage as Alice can find the location after downloading the document. Another way is to divide the document into chunks of data and use bloom filter for each chunk, the location is based on the chunk size granularity however exact location is not known.

- **Occurrence**

In order to find occurrence of a keyword, the bloom filter must be changed slightly where the number of occurrence of the keyword is appended to the word before creating the keyword digest. The disadvantage of doing this is that more unique keyword is created.

3 *Methodology*

Based on the search methods analyzed earlier, a hybrid method of all three methods will be proposed.

The motivation behind the creation of this method is

1. Have a method that allows the owner of the data to find the required data from a remote and untrusted storage
2. Supports any types of data
3. Allows Alice to choose just the required keywords describing the data
4. Preserve the keywords where the keywords can be retrieved if needed.
5. Time complexity of O(1) to search for a keyword
6. Easy integration with any existing indexing scheme
7. Good performance time in terms of encryption, decryption and search

With these objectives in mind, the new search method is described as below. It also maintains the needed securities from the methods analyzed earlier.

3.1 Scheme 1

When it comes to having a fast and efficient search, methods like hash tables and trees are deployed to reduce the time needed. The common architecture is that each of them has to build a kind of index representing the data which can be accessed based on a certain function. This results in an O(1) time complexity search time for the best case while the worst case will never be more than O(n) time complexity. Scheme 1 will incorporate indexing.

3.1.1 Setup / Encryption Phase

In this method, the keywords $W_1, W_2, \ldots, W_t$ where t is the number of keywords belonging to a document D will be organized into a hash table known as HT. The keywords are allocated to different location of the hash table with the use of a hash function $H:\{0,1\}^m \rightarrow \{0,1\}^n$ where m represents number of binary of the word to be hashed and n represent the number of binary digit for the allocated cells in the hash table HT.

To ensure the safety of the keywords, it will be first encrypted using key k' resulting the encrypted word $E_{k'}(W)$. This will be used to help in finding the location of the cipher text in the hash table (Figure 21).

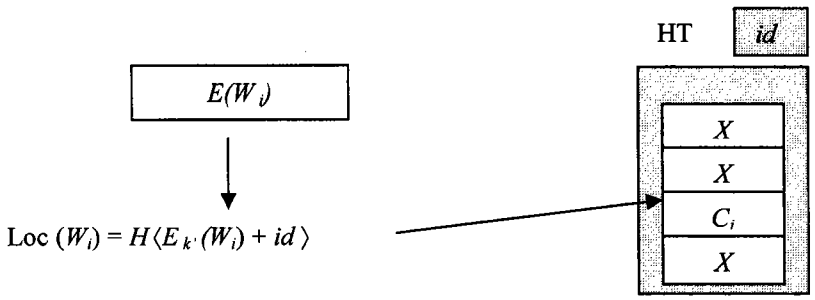


Figure 21: Defining the location for the encrypted word

It would be tempting to just insert the encrypted word into the location defined by the hash function and thus creating a complete encrypted index. However, this can be dangerous as the encrypted word the single encrypted word is prone to analysis attack where the same encrypted word will record the same value in different document within the hash table.

Due to this, it would be better to insert a different value in the hash table. However the value should allow the keyword to still be searchable. This brings Scheme 1 to utilize the SWP idea that generates a different value for each encrypted word. In SWP method, the random number generator allows this attribute to work. Therefore the creation of the cipher text C_i is done through the XOR product of the encrypted word $E_{k'}(W)$ with the random number block T_i. With $Loc(W_i)$ determining the location in the hash table HT, the value C_i can be stored (Figure 22).

At this point, the cipher text C_i can also be a candidate for determining the location instead of the encrypted word $E_{k'}(W)$. The reason cipher text C_i is not used is due to the search phase(Section 3.1.2) where it can skip the process of recreating cipher text C_i just to find the location of the encrypted word.

Additional feature of having search properties like exact location and proximity search can be enabled with the help of encrypting the word location W_{Loc} with the key dependent word k_i. This will allow the server to decrypt the location value for that particular searched word.

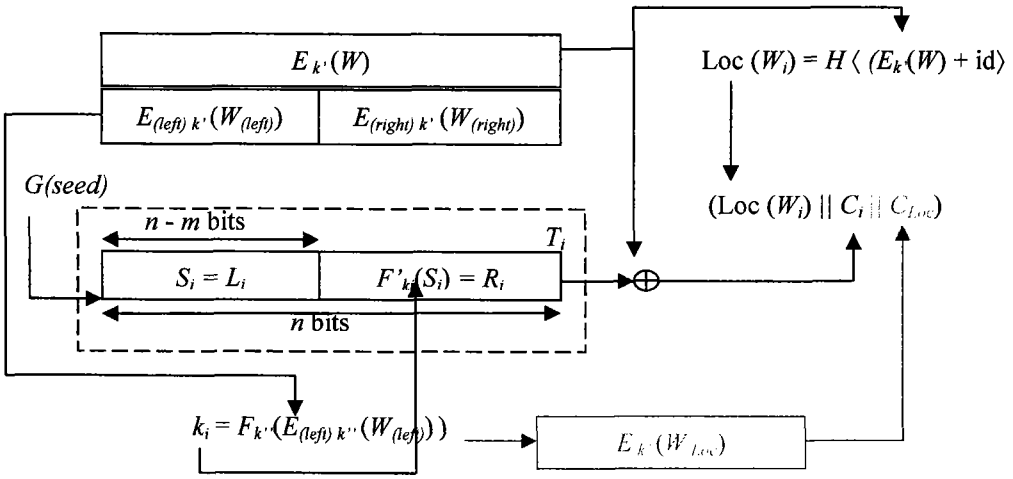


Figure 22: Generating different cipher text for storing

However the usage of the hash table does not allow the arrangement of keywords to be in the correct order. This poses a problem during the decryption phase (Section 3.1.3). The decryption phase requires the pseudorandom number generator G to generate different random number S_i for different cipher text C_i to be XOR, creating the encrypted word $E_{k''}(W)$. This process needs the data to be in the right order. In the hash table, the order of the cipher text is not recorded, thus disabling the decryption phase.

It may be possible to encrypt a single word at a time, resetting the generator for every encryption but this does no longer create random value resulting similar S_i for every position i. Another possible way is to attach a single number on the cipher text identifying its order at the expense of increasing the size needed. This way would effect the searching and would not be a good way.

A better way is to have a generated number from a function that is based on the keyword which will create uniqueness of each S_i and the document number id creating uniqueness of S_i among documents. This results an additional function $F''_{k'''}$ where $S_i = F''_{k'''}(E_{k'}(W), id)$. As this value is needed in the generation of the S_i it would be a 'must' requirement that this value is known. The parameters for this function matches the values used to generate Loc $(W_i) = H\langle E_{k'}(W_i) + \text{id}\rangle$; allowing the Loc value to be reused to generate S_i.

The Loc value cannot be used raw for S_i as this allow the server to XOR a portion of cipher text C_i to generate the encrypted text left portion for every word in the hash table. The function $F''_{k'''}$ will generate new values for different words with the key k''' ensuring it cannot be generated by Bob.

In this phase, three keys are needed to perform the setup for the keywords. Keeping three different keys would be hard to be managed. Therefore a master key *mk* can be kept by Alice where a pseudorandom number generator *PRNG* will be used to generate the three keys for Scheme 1 from the master key.

This completes the setup / encryption phase for Scheme 1(Appendix C)

3.1.2 Search Phase

Although the setup phase and decryption phase consist of quite a number of steps, the search phase is still quite simple. The server will just require either 3 or 4 value depending on the search mode.

3.1.2.1 Single Document Searching Mode

For searching on a single document, the server would require the document number and location of the cipher text in the hash table to perform a direct search of O(1) time complexity (Figure 23).

To check whether the word exists on the server, Bob will need to do an XOR operation of the encrypted word and cipher text, generating the other half of S_i with function F' and key k_i. A comparison of the generated portion and the existing portion will check if the encrypted word is the one that is being searched

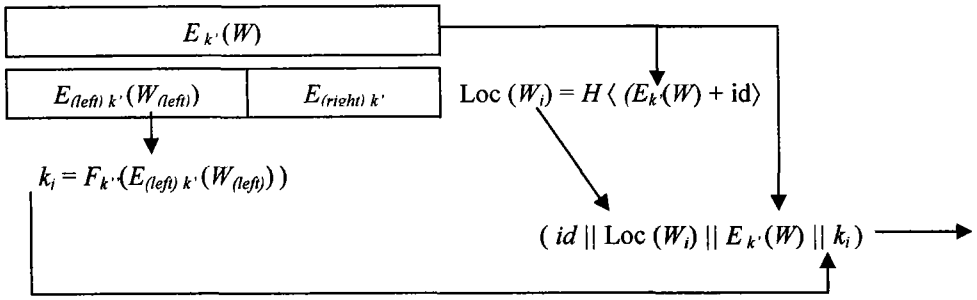


Figure 23: Single Mode Search

3.1.2.2 Multiple Documents Search Mode

This search is performed when there is a need to find a certain word in multiple documents or the document number is unknown. Without the document number, search can still be performed. This is possible as the hash function just required the encrypted word value to enable the hash function H to find *Loc*. This allows Bob to do the hash function H on behalf of Alice. The only information needed by Bob would only be the encrypted word $E_{k'}(W)$ and key k_i. Bob would need to find the possible location of the word by doing the hash function H on the given value $E_k(W)$ with the document id for all hash tables (Figure 24).

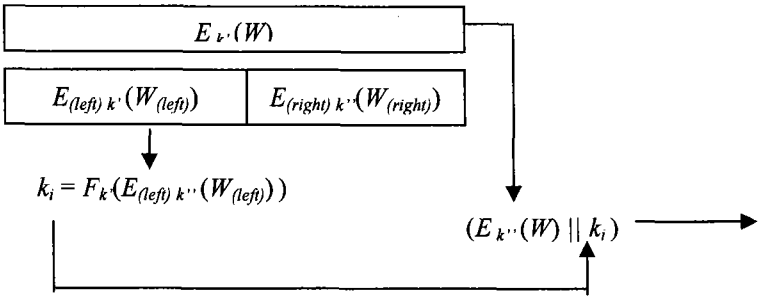


Figure 24: Multi Mode Search

This completes the search phase for Scheme 1(Appendix D, E)

3.1.3 *Decryption Phase*

With two public values known for every cipher text C_i, Alice would need to be able to decrypt the whole keyword list. The location for each C_i now plays an important part here as the value is used to generate the S_i for each C_i to be XOR resulting $E(W_i)$. Without *Loc*, cipher text C_i cannot be decrypted. The decryption process is similar to SWP method where half of the encrypted word will be derived from the C_i allowing the other half to be derived next.

Both portion of the encrypted word would allow decrypting of the word possible (Figure 25).

If the word location W_{Loc} is available it can be decrypted with the key dependent word k_i.

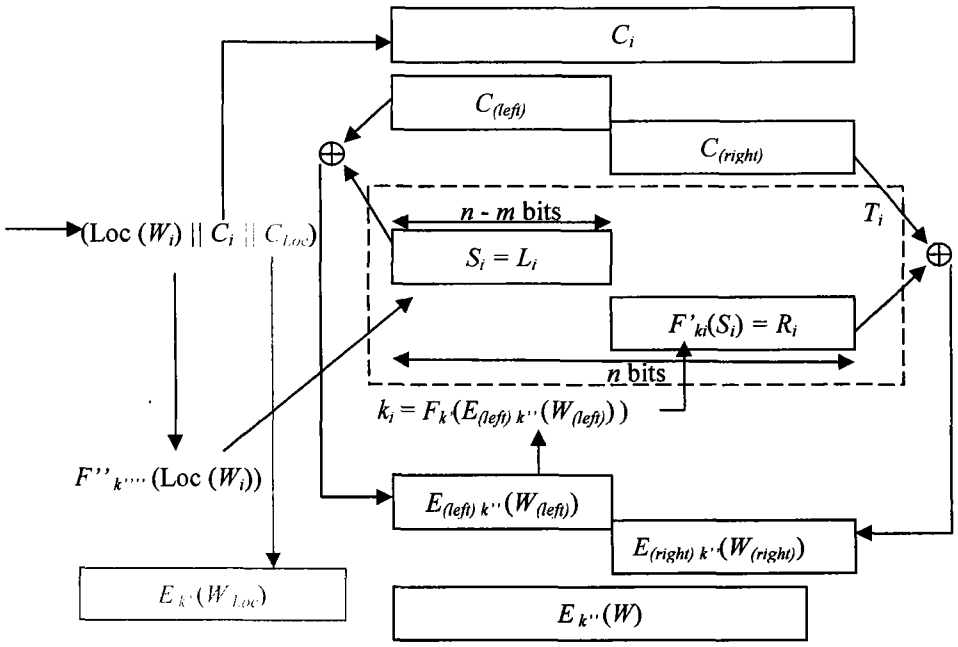


Figure 25: Decrypting the keywords

This completes the decryption phase for Scheme 1(Appendix F).

3.1.4 *Hash Method*

The hash method usage in Scheme 1 is important where it has a direct impact to the performance in size growth, setup, search, decryption time complexity. This is because the value of C_i

is required to be kept in the hash table to enable search and decryption of the keywords. The hash method to be deployed here however differs from Goh's Bloom Filter [11] where the hash function is a cryptographic hash function [9]. The reason for this is that the keywords used in Goh Bloom Filter are in plain text form and therefore requiring the hash function to also be cryptographically secure to prevent data leakage. Without having to keep the hash value and reducing the value to a single bit, Goh's Bloom Filter Method requires the hash function to be generated more than once to ensure the false positive value is within an acceptable range.

3.1.4.1 Hash Coding on Hash Table with Separate Chaining

An example of a common and easy hash method "hash table with separate chaining" will be analyzed with Scheme 1. In this hash scheme, an initial size of array will be chosen with a fixed length of m depending on the number of keywords and collision rate is to be constructed. For every keyword inserted, a link list object for each position is allocated n bits in size where n is equal to the size of the link list holding the cipher text. When a collision is detected in the hash table, the collided cipher text will be linked to the last link list in that location. Based on this implementation, it can be seen that the storage size grows in linear to the number of keyword therefore creating an O(n) growth.

When a search is performed on a hash table with separate chaining, it allows a constant time complexity which is an O(1) to be achieved. The hash function which converts the key value to a location value in the hash table allows a direct access. However the time complexity of O(1) represents the best case time complexity as there can be few values in the same location due to collision. So the worst case time complexity would be O(n) where all the keywords "accidentally" hashed into the same location of the hash table. Although this is unlikely it must still be taken into consideration, therefore creating an average time complexity of finding a keyword using this method as $O(\lambda)$ where λ represents the load factor of n/m (number of keywords per table size).

This gives a great amount of improvement over SWP methods. However the method that will be used in Scheme 1 would be Pearson Perfect Hash Function (Section 3.1.4.2).

3.1.4.2 Pearson Perfect Hash

Currently there have been researches done on the area of perfect hashing. One of them is the hashing method proposed by Pearson [10]. This method is known as Pearson's Perfect Hash [9]. The definition of a perfect hashing is defined as

"A hashing function is perfect, with respect to some list of words, if it maps the words in the list onto distinct values, that is, with no collisions. A perfect hashing function is minimal if the integers onto which that particular list of words is mapped form a contiguous set, that is, a set with no holes."[10]

This will improve the search time by reducing the collision among keywords thus reducing search time and saving space size.

However this hash function is reversible, therefore in Scheme 1, the keywords are encrypted first together with the id to prevent data leakage. With the possibility to find the original value from the final hash value, the value that can be derived from Scheme 1 would be the encrypted word plus id which cannot be used XOR every value in the hash table.

In this hash function, a sequence of bytes is processed one byte per time using only a single XOR function. However the end value is defined from a prepared table containing random bytes. The hash function algorithm proposed by Pearson is as below

```
h[0] := 0;
      for i in 1..n loop
            h[i] :- T[h[i-1] xor C[i]];
      end loop;
return h[n];
```

Code 1: Pearson Hash Function [10]

Due to the processing of a single byte per time, a variable length word W can be supported by this algorithm. A single word W may contains up to n characters that is represented as $C_1, C_2, \ldots, C_n$. Each character equals a byte where a single character will be XOR with the hash value of the previous same function iteration. If the character is the first one, it will be XOR with an empty value of 0. The product of the XOR will act as the index value of table T containing the random bytes. The random bytes in the range of 0-255 act as a one to one permutation of the index value which has the same range 0-255.

A sample table T showing all permutation of 0-255 (Table 1) and an example of perfect hash (Table 2).

Table 1: T Table Permutation of Pearson Perfect Hash

39	159	180	252	71	6	13	164	232	35	226	155	98	120	154	69
157	24	137	29	147	78	121	85	112	8	248	130	55	117	190	160
176	131	228	64	211	106	38	27	140	30	88	210	227	104	84	77
75	107	169	138	195	184	70	90	61	166	7	244	165	108	219	51
9	139	209	40	31	202	58	179	116	33	207	146	76	60	242	124
254	197	80	167	153	145	129	233	132	48	246	86	156	177	36	187
45	1	96	18	19	62	185	234	99	16	218	95	128	224	123	253
42	109	4	247	72	5	151	136	0	152	148	127	204	133	17	14
182	217	54	199	119	174	82	57	215	41	114	208	206	110	239	23
189	15	3	22	188	79	113	172	28	2	222	21	251	225	237	105
102	32	56	181	126	83	230	53	158	52	59	213	118	100	67	142
220	170	144	115	205	26	125	168	249	66	175	97	255	92	229	91
214	236	178	243	46	44	201	250	135	186	150	221	163	216	162	43
11	101	34	37	194	25	50	12	87	198	173	240	193	171	143	231
111	141	191	103	74	245	223	20	161	235	122	63	89	149	73	238
134	68	93	183	241	81	196	49	192	65	212	94	203	10	200	47

Table 2: Minimal Perfect Hash Result

1	a	9	for	17	in	25	the
2	and	10	from	18	is	26	this
3	are	11	had	19	it	27	to
4	as	12	have	20	not	28	was
5	at	13	he	21	of	29	which
6	be	14	her	22	on	30	with
7	but	15	his	23	or	31	you
8	by	16	i	24	that		

As seen in Table 2, a set of 32 words using Pearson hash function have created a minimal perfect hash where the words are hashed nicely leaving no empty cells in between.

However there might be a need for a larger indices size more than 256. This can be done by adding 1 to the first character of the processed string *H1*. However, this might allow an overflow of the range 0-255. To fix this, a (modulo 256) is applied bringing back the result within the correct range. The hash function is reapplied to H1 resulting H2. Both H1 and H2 are concatenated then forming a larger size which is 65535.

As for Scheme 1, the hash will be applied to the encrypted keyword where all the keywords have a fixed length due to the output of the encryption algorithm. This would be where 1 byte can be the input of Pearson hash function per iteration. Due to the fact Pearson Hash Function can take any input length, it would not be a problem for taking an encrypted word. As for the hash table size, it can be increased to 65535 in size which would be considered large where keyword list does not reach that total amount.

Considering that all the keyword is hashed 'perfectly', the worst time complexity of a search in the index will be the load factor which is n/m (number of keywords per table size). Therefore, if the keywords are within the range of allocated space, it is definitely a time complexity of O(1). So using this hash function will give an average $O(\lambda)$ time complexity where λ represents the load factor of n/m (number of keywords per table size). As and indexing scheme, the worst case would be $O(n)$

4 *Discussion*

The discussion will revolve around the three main methods studies earlier, which are SWP Linear Scan, SWP Encrypted Index and Goh Bloom Filter in comparison with the new proposed method (Scheme 1). Aspects that will be discussed are search properties, data types supported, space cost; time/work cost, key management, encryption methods, decryption and precision.

4.1 Search Properties

- **Exact location**

In SWP Linear Scan and SWP Encrypted Index, the exact location of the keyword in the document can be found. This is due to the way of setup for SWP Linear Scan where, each word is encrypted and send to Bob, $C_i = W_i \oplus T_i$. When Alice sends Bob the word W_i and key k_i, Bob will check each cipher text by XOR operation with W_i and finding a match of R_i and $F_{ki}(L_i)$ of T_i. With a counter for each checked word, Bob can tell the exact location of the word to Alice.

As for SWP Encrypted Index, this method is an extension of the SWP Linear Scan where selected keyword is put into an index that contains a list of pointers to the document that has the keyword. Due to this exact position of the word is unknown.

Goh Bloom Filter does not allow exact location of word information to be given to Alice. The setup of this method, where all words are hashed into an array of distinct bit address inserted in the bloom filter does not keep information of the location. Bob can only search for the word in an array of m bits bloom filter only. An improvement here is that the document is divided into a few chunks where each has its own bloom filter. This still does not allow an exact location but nearer location information. However this increases the space cost, and more time/work to setup and search for a word.

This new method does not allow Alice to know the exact location of the word like SWP Linear Scan. It is possible to enable this feature with the help of the word location encrypted with the key dependent word resulting $E_{kw}(WL)$. The key should derive from the hash function with the left n-m bits of the encrypted word $E(W)$. This is so that Bob is able to decrypt the word location when a search is performed. Downside of this is that Bob knows the location of the word in the document. This downside is clearly visible in SWP Linear Scan as the encrypted word are arranged in order

However if the document is not text based like movies or images, exact location would not work but it will enable proximity search for keywords (Section 4.1 Search Properties - Boolean Queries).

- **Controlled search**

Instead of asking Bob to search for every document for a certain keyword, Alice can control the search to a specific document or area in the document.

For both SWP methods, the key can be generated in a way that more information like chapter and document number is included where $k_i = f_{k'}(\langle D, C, W \rangle)$ where D is the document, C is the chapter and W is the word. This allows Alice to find a word in one of the document of Alice's choosing and within a chapter in that document.

Goh Bloom Filter method does allow a control of search on a certain document because the generating of hash bit address for the bloom filter uses the document number as a parameter. As for searches within a document, the document has to be divided into chunks. This leads to space cost increments.

Due to the document number usage similar to Goh Bloom Filter method, the new method allows a controlled search on a certain documents because Bob will be given the document number to search on. With the usage of SWP Linear method, words can be defined to a certain part of the document by changing the key generation algorithm

- **Variable length keyword**

The main disadvantage of SWP Linear Scan is the length of the keyword where every keyword has to be a same length. For languages like English, it is almost impossible to have a document with only same length words. With this shortcoming, SWP introduced padding of keyword that has shorter length and splitting of keyword of longer length (Section 2.1.6). Another way is to use variable length keyword and results cipher texts of different length. This makes the searching of a word to be harder where search can only be performed on bits boundary instead of blocks due to changing size of cipher text. Although space cost will decrease, the time/work needed to find a keyword increase a lot.

SWP Encrypted Index does not suffer from this problem as the encrypted word is just inserted into the index.

Goh Bloom filter does not have this problem because keywords are applied to a hash function of r keys and results an array of distinct address bits for the bloom filter.

The new method does not allow variable length keyword blocks as the keyword are encrypted first. This requires padding of the word either by Alice or by the encryption class using the padding like PKCS7.

- **Boolean queries**

SWP Linear Scan, SWP Encrypted Index and Goh Bloom Filter allow Boolean queries to be performed. For Boolean command like "AND", different keywords can be sent to Bob where Bob will find document that includes both the keywords and "OR" command where the document contains either one of the keywords.

The only advantage of SWP Linear Scan to Goh Bloom Filter and SWP Encrypted Index is 'proximity search' where this method allows two words be search where one precedes another. This is possible due to exact location capability.

Like the three methods, the new method also allows Boolean queries because it has the same architecture as SWP Linear Scan.

The new method has the capability of finding proximity with the help of the additional location word attached to the cipher text that enables the exact location property (Section 4.1 Search Properties – Exact Location).

- **Regular expression search**

All methods allow regular expression searches like "$ab[a-z]$" but does not allow the use of wildcard like "*" as the combination of keywords will be too much.

- **Occurrences**

Occurrence of word can be found by appending the occurrence number to the keyword.

For SWP methods, the word can be change to $W = \langle 0, \text{test} \rangle$ and $W_i = \langle 1, \text{test} \rangle$ for the next occurred word. The size of the word increases by a constant bit size for every keyword.

As for Goh Bloom Filter, the word is also appended $W = \langle 0, \text{test} \rangle$. However instead of having one keyword, Goh Bloom Filter will create another unique keyword for every occurrence. The increase of unique keyword will result in more false positives. To get the desired false positives rate, the array size will need to be increased accordingly.

For the new method, due to the usage of keywords like Goh Bloom Filter, occurrences of a word leads to more keywords where 3 occurred words will need 3 different keywords.

- **Query isolation**

SWP methods use a key based on the word for the generation of T_i. By doing this, SWP prevents Bob to know about the other words in the document, thus the search for one word does not leak information about other different words.

Goh Bloom Filter has this property as well because the value generated by the hash function is different for each document.

Together with SWP methods that uses key based on the word for the generation of T_i, this prevents Bob to know about the other words in the document, thus the search for one word does not leak information about other different words.

4.2 Data Type

The data type that will be considered here refers to the document that Alice wish to keep at Bob's file server.

In SWP Linear Scan, the document used must be text based like English. Difficulty sets in when the words are variable lengths, as SWP Linear Scan must be changed to either use padding method or variable length setup method. Some text-based languages may pose a hard time, to identify atomic word. When SWP Linear Scan is applied to different type of document like images and DNA sequence, it would not work. The reason to this is the data type has no "atomic word" for SWP Linear Scan to encrypt.

SWP Encrypted Index does not face this problem as the search is based on index table and not the data leaving the data to be of any type.

Goh Bloom Filter has this flexible as well where the data type of the document can be anything as the search is based on index of bloom filter document and not the document itself.

Keywords are chosen by Alice for both SWP Encrypted Index and Goh Bloom Filter thus eliminating any physical link to the document. Unlike SWP Linear Scan the search words are taken from the encrypted document only

By using the architecture of Goh Bloom filter, the data type of the new method can be anything, as it is not affected by the keyword list. A link is done between each keyword list and document.

4.3 Key Management

The number of keys for each method that is required for Alice

- **SWP Linear Scan**
  1. Key for Encrypted Word $E_{k''}(W_i)$
  2. Key/seed for the pseudorandom number generator G to create S_i
  3. Key for generating k_i for T_i

- **SWP Encrypted**
  1. Key for index keyword encryption $E_{k''}(W)$
  2. Key for document pointer list $E_{kp}(P)$, $kp = F_{kw}(E_{k''}(W))$

- **Goh Bloom Filter**
  1. Key/seed for a pseudorandom number generator that generates r keys for the hash function h

- **New method (1 key version)**
  1. Key for generating 3 pseudorandom key from a pseudorandom number generator
     a. Key for encryption of the $E_{k'}(W)$

b. Key for creating the S_i block
c. Key for generating the other half of S_i block

The keys required for Goh Bloom Filter is only one, followed by SWP Encrypted Index of two and SWP Encrypted Index of three keys. SWP Linear can be reduced to two if the key k_i for T_i and the encrypted document is the same.

However by using pseudorandom number generator, one master key will only be needed where it will generate the three keys for this method. With one of the keys compromised, the attacker will still not be able to deduce the other two keys due to the properties of the pseudorandom number generator. The attacker would require all three keys in order to decrypt the encrypted word or just the master key.

4.4 Time/Work Cost

To measure the time/work cost each process/step is assumed to be a unit time operation r. With the r operations identified, the time complexity of the whole process is defined in the O notation. Throughout the analysis, variable n defines the number of keywords for each document while variable m defines the number of documents.

4.4.1 Setup

The setup cost required

- **SWP Linear Scan**
  1. Generating of S_i from the pseudorandom number generator G
  2. Encryption of $E_{k''}(W)$
  3. Generation of key $k_i = f_{k'}(E_{k''}(W))$
  4. Generating the remaining T_i which is $F_{ki}(S_i)$
  5. XOR operation to get $C_i = E_{k''}(W) \oplus T_i$.

- **SWP Encrypted Index**
  1. Encryption of document
  2. Encryption of keyword $E_{k''}(W)$
  3. Generation of key $k_p = F_{kw}(E_{k''}(W))$
  4. Encryption of document pointer $E_{kp}(P)$

- **Goh Bloom Filter**
  1. Encryption of document
  2. Generating of r keys from the pseudorandom number generator G
  3. Hashing of keyword $F = h_1(x), \ldots, h_r(x)$
  4. Hashing of hash word with document number. $F = H_{h1(x)}(i), \ldots, H_{hr(x)}(i)$.

- **New Method**
  1. Encryption of $E_{k''}(W)$ - Encryption of $E_{k''}(WL)$
  2. Hash function to get Loc(W)
  3. Generating of S_i from F'' with Loc(W) and k''
  4. Generation of key $k_i = F_{k'}(E_{k''}(W))$
  5. Generating the remaining T_i which is $F'_{ki}(S_i)$
  6. XOR operation to get $C_i = E_{k''}(W) \oplus T_i$ - Append $E_{k''}(WL)$

The setup cost for SWP Linear Scan is $5r$, SWP Encrypted Index $4r$ and Goh Bloom Filter $4r$. Goh Bloom Filter has the same setup cost of a single document compare to SWP Encrypted Index method. The time for setup of all three method have a constant time giving O(1) time complexity for each word. However work/time cost increases linear to the number words in the document giving an O(n) time complexity.

The setup cost for the new method is $6r$, resulting much more steps that the other three methods. The time for setup are still a constant time giving O(1) time complexity for each word and O(n) time complexity for a keyword list. If the exact location is needed, it will require $8r$.

4.4.2 Deletion

As for the deletion of document of each method

- **SWP Linear Scan**
  1. Deleting the document

- **SWP Encrypted Index**
  1. Deleting the document
  2. Recreation of the index with padding of false pointer

- **Goh Bloom Filter**
  1. Deleting the document with bloom filter

- **New Method**
  2. Deleting the document with bloom filter

The deletion cost for SWP Linear Scan and Goh Bloom Filter is $1r$, while SWP Encrypted Index requires an additional for the recreation of index making it $2r$. In this step, SWP Encrypted Index has to recreate all the indexes for deleting a document giving it an O(n) time complexity on the numbers of keywords. SWP Linear Scan and Goh Bloom Filter has a constant O(1) time complexity. This increases linear to the number of documents giving it O(m) time complexity on multiple documents for SWP Linear Scan and Goh Bloom filter. SWP Encrypted Index on the other hand has an O(n) * O(m) = O(nm) time complexity making this method inappropriate for deletion or updating.

The deletion for the new method follows the same idea of Goh Bloom Filter giving only $1r$ resulting O(1) time complexity for a document and increases linear to the number of document giving it O(n) cost for multiple documents.

As for the updating of documents in each method, it requires the cost of deletion and re-setup.

4.4.3 Searching

As for the search of document of each method

- **SWP Linear Scan requires**

  Alice Side
  1. Encryption of $E_{k''}(W)$
  2. Generation of key $k_i = f_{k'}(E_{k''}(W))$

  Bob Side (Padded Word/Variable Length)
  1. Search every cipher text for a match of $Ri = F_{ki}(L_i)$ per block/per bit

- **SWP Index Scan requires**

  Alice Side
  1. Encryption of keyword $E_{k''}(W)$
  2. Generation of key $k_p = F_{kw}(E_{k''}(W))$

  Bob Side
  1. Decryption of the encrypted pointer document list
  2.

- **Goh Bloom Filter**

Alice Side
1. Generation of the r keys from the pseudorandom number generator G
2. Hashing of keyword $F = h_1(x), \ldots, h_r(x)$
3. Hashing of hash words with document number $F = H_{h1(x)}(i), \ldots, H_{hr(x)}(i)$. (with knowledge of the document number)

Bob Side
1. Hashing of hash words with document number $F = H_{h1(x)}(i), \ldots, H_{hr(x)}(i)$. (without knowledge of the document number)
2. Check every bit address of the hash value for every Bloom filter (document)

- **New Method**

Alice Side
1. Encryption of $E_{k'}(W)$
2. Hashing of $E_{k'}(W)$ to generate $Loc(W)$
3. Generation of key $k_i = f_{k'}(E_{k'}(W))$

Bob Side
1. Search every cipher text for a match of $Ri = F_{ki}(L_i)$ for $HT[Loc]$

The preparation for SWP methods are $2r$ work cost and the search on the server side cost $1r$ work. A $2r$ work cost for Goh Bloom Filter preparation is needed where the document number is not known, allowing him to search in a multiple documents setting. This puts an additional r on the searching side, where the server needs to calculate a new hash value based on the document number. However, if Alice knows the document number, Alice would be able to search only that particular document thus transferring the additional r to the client side.

The preparation for the method requires $3r$ work cost.

The preparation time complexity for all three method has a constant O(1) time complexity on a single keyword and O(n) time complexity for n keywords and O(nm) time complexity for m documents assuming each document have the same number of keywords.

As for the searching side, the time complexity differs.

SWP Linear Scan have an O(n) time complexity because the search is done to every block/word in the document. As for multiple documents, it will need another O(m) search cost on the number of documents in the server. This gives the method an O(nm) time complexity.

On the other hand, SWP Encrypted Index will have an O(1) time complexity whether on a single or multiple document environment as a single index represents any number of documents

Goh Bloom Filter will have an O(1) time complexity for a search on a particular document required if he knows the document number. As for multiple document or the document number is unknown, this method has an O(m) time complexity.

New Method searching side has a constant O(1) time complexity if there is a perfect hashing. However it will differ giving the worst time as O(n).

4.5 Space Cost

The space cost for SWP Linear Scan is linear to size of the document as every word is encrypted thus having an O(n) growth. For variable length method, the space cost can be reduced. The cost of time/work increases as more words are contained in the document, the growth in the space cost then would be O(nm) for multiple documents.

If the padded method is use; the space cost will be much larger than the variable length method due to the fixed size of the encrypted word whether the word is a short one. The growth in space cost is similar though.

SWP Encrypted Index requires fixed space size for the index. Even when keywords are inserted, the size space will still be same, which is O(1). The way the document is kept is not mentioned in the method, so it would be best to compress and encrypt it. As for multiple documents, the index size space does not grow and remains the same. However additional document requires O(m) growth in size space. Total space would be O(1) + O(m) resulting growth of O(1+m).

Goh Bloom Filter only requires an array of b bits with the document. The formula for choosing a good bloom filter parameter is given as $r = ln2 \times b/d$ and false positive as $F = ½^k$[2]. As seen in this formula, the size of b bits increases linear with the number of document d for a fixed r key hash function required for a good false positive rate. The false positive F in Goh Bloom Filter gives some additional security measures. Increment of the false positive will lead to the increase of b bits of the bloom filter. The document in this method can be encrypted with any method Alice wishes and be compressed which gives an advantage of space saving of documents. Therefore the space is the encrypted & compressed document with m bits (size of Bloom Filter). However, as more keyword is inserted, the space does not increase, as all the words are hash coded into the b bit array bloom filter. This gives a constant space cost of O(1) and O(m) growth for multiple document.

The space cost for the new method is constant in respect to the desired load factor collision rate therefore give a constant O(1) on the space allocated. However it would grow linearly if as collision happens O(n).

4.6 Encryption Methods

All methods allow Alice to choose an encryption method for the document or words.

4.7 Decryption

SWP methods and the new method preserve the word by encrypting it and storing it as a searchable value. Goh Bloom Filter however does not as its main objective is to have small data storage space and fast setup and search. This is done through hash functions and no storage space for the data.

- **Rebuilding of Encrypted Search**

The usage of SWP Linear Scan for the keyword index gives Alice the advantage of rebuilding the keyword list without first remembering the keywords that were selected for the document. As more and more documents are kept in Bob's fileserver, Alice would not be able to remember all the keyword for each document; therefore there must be a way for Alice to retrieve the keywords used before.

Using this method, Alice can download both the keyword index and the document and decrypt the keyword list. With the list of plain text word, Alice can remove the unnecessary keywords and add additional required one. After that, a new document number must be selected and the setup process is performed again.

One of the benefits of decrypting the keyword is the use of document type like movies and images. Let's say when Alice leaves an organization; the data stays with the organization. A friend of Alice takes over the data will require the rebuilding of the whole database with a new password (keyword and seed). The decryption allows Alice's friend to do it effortless by providing the new passwords and not to find or rethink the keywords of each document again. For non-text based document, Alice's friend might not even have the slightest idea on what the data is about.

Another possible usage of decryption of keyword index is when an adversary has found the password and it has to be re-encrypted. For prevention, the change of password can be scheduled periodically as a maintenance plan.

This cannot be done on Goh Bloom Filter Method as it is only one way hash coding and due to the merging of keywords, there is no way of separating them or knowing them.

4.8 Precision

All the methods has false positive due to the fact of having hash function reducing the data size.

However as the new method also allows the decryption on the keyword, the owner has the ability to do an extra check will allow only the correct document be downloaded thus enforcing precision. This process of checking would be a disadvantage due to the requirement of an extra trip of command to the server and decryption process at the owner's side. On the other hand, this could also be an advantage where the data to be retrieved is large in size, for example movies and audio files. Downloading unwanted data would take up extra bandwidth, time and processing power to decrypting it.

4.9 Summary of Properties

The below table shows a list of the properties for all four search method analyzed in this paper (Table 3).

Table 3: Summary of Properties

	SWP		**Goh**	**Improved**
	Linear Scan	**Encrypted Index**	**Bloom Filter**	**New Method**
Exact Location	✓	×	×	×/✓
Controlled Search	✓	✓	✓	✓
Variable Keyword Length	×/✓	✓	✓	×
Boolean Queries	✓	✓	✓	✓
Proximity Search	✓	×	×	×/✓
Regular Expression	✓	✓	✓	✓
Occurrences	✓	✓	✓	✓
Data Type	Text	Any	Any	Any
Key Management (no[subkeys])	3	2	1[4]	1[3]
Time/Work Cost (Setup) (per word / per doc)	5r O(n) / O(nm)	4r O(n) / O(nm)	4r O(n) / O(nm)	6r / 8r O(n) / O(nm)
Time/Work Cost (Deletion) (per word / per doc)	1r ----- / O(m)	2r ----- / O(nm)	1r ----- / O(m)	1r ----- / O(m)
Time/Work Cost (Search) (per word / per doc)	2r O(n) / O(nm)	2r O(1) / O(1) O(n) / O(n)	1-2r O(1) / O(m)	1-2r O(1) : O(m) O(n) : O(nm)
Space Cost (per word / per doc)	O(n) / O(nm)	O(1) / O(1+m)	O(1) / O(m)	O(1) : O(m) O(n) / O(nm)
Encryption Method	Any	Any	Any	Any
Decryption	✓	✓	×	✓
Precision	×	×	×	×/✓

5 *Prototype and Results*

The three methods SWP Linear Scan, SWP Encrypted Index and Goh Bloom Method will be studied in detail. The details of the processes are converted into pseudo code in preparation to be coded into a programming language as prototypes for testing and proof of concept of the methods researched.

5.1 SWP Linear Scan Prototype

5.1.1 Pseudo Coding

SWP Linear Scan consists of three usages which are encryption, search and decryption.

Due to the architecture of SWP Linear Scan, it can only support fixed text length words. In order to support longer words, the words have to be pre processed first where it will be split into two consecutive words. As for punctuations, it has to be separated and represented as a single encrypted word to enable search be done on the word. Together with the punctuation will create different encrypted text value thus preventing normal keyword searches. Paragraphs' marker is included to preserve the original formatting in the file.

The pseudo code for SWP Linear Scan can be found in Appendix H

5.2 SWP Encrypted Index Prototype

5.2.1 Pseudo Coding

SWP Encrypted Index Scan consists of three usages which are encryption, search and decryption.

As this method uses SWP Linear Scan to create the index, it can only support fixed text length words thus longer word need to be split into two consecutive words. As this is an index there will be no need for punctuations or any formatting details to be recorded.

The document number is needed to identify the documents which consist of the encrypted search keyword. To enable this, pairs of data will be kept in the hash table where the encrypted word will act as the search index while the data contains all the associated document number.

The pseudo code for SWP Linear Scan can be found in Appendix I

5.3 Goh Bloom Filter Prototype

5.3.1 Pseudo Coding

Goh Bloom Filter method uses an array of Boolean to mark each hash value generated. Due to the size of the hash value which is large, the hash area needs to be created using bucket method where hash values are broken into ranges and kept in different smaller hash area.

There is no special decryption way as this method does not support decryption of bloom filter. The decryption will be confined to just decrypting the encrypted file.
The pseudo code for Goh Bloom Method can be found in Appendix J

5.4 *Scheme 1 Prototype*

5.4.1 *Pseudo Coding*

Scheme 1 method uses SWP Linear Scan Method for encryption of the keywords. The encrypted keywords are kept in an index similar to SWP Index Method with new enhancement on the indexing scheme where the location is defined by Pearson Perfect Hash Function. By using keywords for indexing, the actual data is separated from the search data like Goh Bloom Method

The pseudo code for Scheme 1 Method can be found in Appendix K

5.5 *Results*

The prototype for the three methods are coded and ran on the below computer specifications.

Processor	:	Intel(r) Pentium(r) 4 2.60GHz
Memory	:	496 DDR RAM
Operating System	:	Windows Server 2003 Standard Edition
Programming Language	:	C# .NET
Encryption Algorithm	:	AES 128 bit key, 128 bit block, 128 bit iv, CBC mode, PKCS7 padding
Cryptographic Hash Algorithm	:	HMAC-SHA1

The tables below (Table 4, 5, 6, 7) show the result of executing all the four methods. A detailed graph representing these values can be seen on Appendixes L, M, N and O.

Table 4: SWP Linear Scan

	No of Words = 980 **Encryption(seconds)**	No of Words = 980 **Decryption(seconds)**	No of Words = 980 **Search(seconds)**
Prep Time	0.207021562246983	0.00071693719397000 4	0.00043308478235725
Post Time	0.00137466322242356	0.00530519335312488	
Processing Time	0.311217135	0.267862917	0.11645776
Average Time (per word)	0.000319525	0.000275013	0.00011956648866203 10

Table 5: SWP Encrypted Index

	No of Words = 375 **Encryption(seconds)**	No of Words = 375 **Decryption(seconds)**	**Search(seconds)**
Prep Time	0.00134303890195474	0.00293470331958206	0.00061659559702355 7
Post Time	0.0146203045841848	0.00652851756178432	
Processing Time	0.07026838	0.099963504	0.00006335351042911 67
Average Time (per word)	0.000190429	0.00027090380453868 10	

Table 6: Goh Bloom Method

	No of Words = 375		
	Encryption(seconds)	**Decryption(seconds)**	**Search(seconds)**
Prep Time	0.00222837799282878		0.00120500443382041
Post Time	14.136435927054	0.00314664301962447	
Processing Time	0.438177299		0.00223311
Average Time (per word)	0.001190699		

Table 7: Scheme 1 Method

	No of Words = 375	No of Words = 375	
	Encryption(seconds)	**Decryption(seconds)**	**Search(seconds)**
Prep Time	0.00325021243783013	0.0284156517716004	0.00025845086170335 8
Post Time	0.0366460161159733	0.00567701893048541	
Processing Time	0.183642046	0.1246849	0.017614753
Average Time (per word)	0.000494992	0.00033607789639980 80	

5.5.1 Number of Words

From the results, SWP Linear Scan will require more numbers of words compared to SWP Encrypted Index, Goh Bloom Method and Scheme 1. This is due to the SWP Linear Scan Method requiring long words to be split into two individual words. Punctuations will also count as a word as well as formatting properties in the original file. SWP Encrypted Index, Goh Bloom Method and Scheme 1 does not require as many word as it only takes unique words to be encrypted. Punctuations and formatting properties can be totally ignored as the original file is encrypted separately.

5.5.2 Preparation Time

- **Encryption**

Due to the increase in numbers of words required, the preparation time for SWP Linear Scan is the highest among the three methods

- **Decryption**

Only SWP Linear Scan methods need a real decryption preparation time to initialize the encryption class as well as reading the encrypted words to be decrypted. SWP Encrypted Index and Scheme 1 have the ability to decrypt the index if needed which is just reading the values in the index for decryption. On the other hand Goh Bloom Method does not have the ability to do so.

- **Search**

Both SWP methods and Scheme 1 have short preparation time, where it would just require the generation of encrypted word and a key to enable the search. Goh Bloom Method would require the process of hashing the word with the document number a number of times based on the amount of pseudorandom number keys. This requires a longer preparation time.

5.5.3 Post Processing Time

- **Encryption**

In SWP Linear Scan, it just requires outputting all processed cipher text into a single file. As for SWP Encrypted Index and Scheme 1, it will require outputting the processed cipher text into a single index file. Both SWP Encrypted Index and Scheme 1 will have longer time compare to SWP Linear Scan due to the additional time required to encrypt the original file. SWP Linear Scan output cipher text is the content of the encrypted file which does not require any additional encryption process.

Goh Bloom Method is require the longest time among all the methods due to the large hash table value that needs to be kept into separate files based on the bucket size. The higher the bucket size, the longer the time would be, while smaller bucket size will result in more files to be handled.

- **Decryption**

Only SWP Linear Scan has real post processing where it needs to write all the decrypted cipher text into a file while reconstructing the long words, punctuations and formatting. SWP Encrypted Index and Scheme 1 will just output the decrypted index with document number into a file. SWP Encrypted Index, Goh Bloom Method and Scheme 1 will require an amount of time to decrypt the original file that was encrypted for storage in the server.

- **Search**

None of the methods require post processing time except displaying the results

5.5.4 Processing Time

- **Encryption**

SWP Encrypted Index require the least time as it only needs to encrypt the word and the document number. SWP Linear Scan has a longer time due to the extra generation of hash values, keys and XOR operation. As Scheme 1 uses SWP Linear Scan together with Pearson Perfect Hashing method, it will require a longer time. Goh Bloom Method has the longest time due to the requirement to perform multiple cryptographic hashing in respect to the number of keys predefined. In addition to that, it requires twice that for the document number hashing to be implemented.

- **Decryption**

SWP Linear Scan needs to go thru a process of decrypting every single cipher text in the encrypted document. This would require the help of hash values generation, key generation and XOR operation thus increasing the processing time. This also applies to Scheme 1; however with the lesser number of keywords, it does not require as much time as SWP Linear Scan. SWP Encrypted Index has a shorter time also due to the number of keywords and does not need to perform as many steps as SWP Linear Scan. However SWP Encrypted Index does two decryption processes which is on the encrypted word and encrypted document, increasing the time needed. Goh Bloom Method does not have the ability for decryption

- **Search**

SWP Linear Scan would need to make a comparison on every cipher text to find the correct match. This takes the longest time as compared to the other methods where indexing is used. SWP Encrypted Index, Goh Bloom Method and Scheme 1 use indexing which is a constant time search regardless of the number of words. Among the three methods that use indexing, Goh Bloom Method is the fastest as it is just a bit comparison.

5.5.5 *Method Comparison Discussion*

Below is the total processing time of each method where the preparation time, post processing time and processing time is added up (Table 8). A comparison of time for all method can be seen from Appendix P, Q, R, S and T.

Table 8: Total Processing Time

Total Time	**Encryption(seconds)**	**Decryption(seconds)**	**Search(seconds)**
SWP Linear Scan	0.51961	0.27389	0.11689
SWP Encrypted Index	0.08623	0.10943	0.00068
Goh Bloom Filter	14.57684	0.00315	0.00344
Scheme 1	0.22354	0.15878	0.01787

- **SWP Linear Scan**

Has a long processing time where each word of the file is encrypted and preserved. The search time increases as with the number of word making this method not suitable for files with many words

- **SWP Encrypted Index**

This method has the fastest processing time. However this method is not feasible where a single master index manages all the documents. The reason for this is that any changes to the documents in the file server whether adding, removing or editing a single document will affect the whole index which promotes information leakage. This will also require a pool of keywords to be maintained.

- **Goh Bloom Filter**

Goh Bloom Filter has very high security where the keywords are hashed and thus irretrievable thru any means. With each document having a single bloom filter which acts as an index for searching, this allows a fast search time without leaking much information. However bloom filter does not allow preservation of keywords and may prove to be a problem if there is a need for keyword retrieval.

Usage of bloom filter has a disadvantage where it requires a large index size to ensure that the false positive percentage is of acceptable level [2]. With the increase of index size, it requires a longer processing time for creation of file buckets and management to hold the index information. This disadvantage makes this method not suitable for active file server where changes to document occur frequently.

- **Scheme 1**

Scheme 1 which is a hybrid of Goh Bloom Filter, SWP Encrypted Index and SWP Linear Scan allow this method to inherit good properties/attributes from these methods which gives an average processing time for setup, decryption and search.

In term of setup, it only keeps meaningful searchable keyword like Goh Bloom Filter and SWP Encrypted Index which gives a much better performance time as well as support for different file types. This method follows the same setup model as SWP Linear Scan during its encryption thus preserving the keywords for retrieval if needed.

Using a single index to document model like Goh Bloom Filter allow changes to a document does not affect the security of other files.

In term of searching, indexing allows a good search time however not as fast as Goh Bloom Filter and SWP Encrypted Index. With an average time, this method is also suitable for active file server where changes to document occur frequently.

6 *Summary and Future Work*

Three different encrypted search methods which are SWP Linear Scan, SWP Encrypted Index and Goh Bloom Filter were analyzed in detail and evaluated on. From the studies we find that an efficient search method on encrypted data has the following attributes. The list of attributes can be seen in Table 7.

Table 7: List of Attributes of Efficient Search on Encrypted Data Method

Attributes	**Description**
Controlled Search	Able to control and focus the search to a particular portion of encrypted data
Variable Keyword Length	Support different keywords length
Boolean Queries	Combine searches using commands like AND, OR and NOT
Proximity Search	Able to search on keyword phrases which is combination of single keywords
Regular Expression	Supports wildcards like '*' and '?' in queries
Data Type	Able to support any kind of data type where the actual data is separated from the search data
Key Management	Does not require keeping multiple secret keys
Space Cost	Linear to the number of document
Search Cost	Constant O(1) time complexity regardless the number of words

Focus has been put into the new proposed method to incorporating the good attributes listed above. With the specification of the new proposed method outlined, Table 8 shows the attributes that are incorporated into the new method.

Table 8: List of Attributes in New Proposed Method

Attributes	**Description**
Controlled search	Able to search on a particular encrypted data based on given data ID
Boolean Queries	Process multiple queries and results merge based on Boolean command
Variable Keyword Length	Partial support by splitting long words
Regular Expression	Wildcards in queries are preprocessed as multiple queries
Data Type	A separate search index created where searches are performed on the index without any dependencies on the actual data
Key Management	Utilization of pseudorandom number generator to create the required sub keys from a single master key
Space Cost	Each document has an index with a document
Search Cost	Uses hash table for a constant O(1) time complexity access at best

However there are still many improvements and new functionality which was not able to be designed into the new method.

Table 9 shows the improvements that can be done over the current proposed method by either enhancement or a redesign of method framework.

Table 9: Current and Future Works

	Current Work	Future Work
1	**Number of Users** Uses symmetric encryption method where a secret key is used for encryption of data and search index. In order for another person to make a search on the encrypted data, the person will require the identical key. Does not have a central KDC for managing key distribution to multiple users	Uses asymmetric encryption method where encrypted search can be perform by another person and data be decrypted [13]. Single way of sending an encrypted message and can only be search and decrypted by the receiver (e.g. email system). Unsuitable for shared fileserver environment where the person that uploaded the encrypted data will still have the ability of searching and decrypting of the encrypted data. Will require a new method or a central KDC
2	**Indexes – Search Time and Space Cost** Every encrypted document requires an index. Increases search time and space cost linear to the number of document.	A single index for multiple documents. A constant time complexity of O(1) and a fixed space cost.
3	**Modification of search data** Cannot support editing, insertion and deletion of keywords as this will leak information of the encrypted word	Single index that supports editing, insertion and deletion of keywords by accessing just a portion of the index without comprising the security of other document and keywords
4	**Data Types Search** Searches be done on different encrypted data types like pictures, sounds, movies with help of keywords thus not confined to search using words	Searches be done on different encrypted data types like pictures, sounds, movies without help of keywords and providing different data type
5	**Search data pattern leakage** Any search will allow server to know the search data being search and the document consisting this search data	Absolute no search data pattern learned. Research done on this attribute such as Private Information Retrieval (PIR) [14] and Oblivious RAM [15]. However this would requires many servers to mask the search pattern where a distributed search is done instead [14] or hardware security modification [15]. A better way of reorganizing the index as well as the encrypted document

The current method serves as an improved efficient search method on encrypted data even though this method has some functionality and improvements not incorporated as listed in Table 9.

However the studies done in this report gives an overview of existing methods proposed to date and attributes contained in an efficient search method. This helps provides a guideline of good attributes or methods to be used in creation of new efficient search method.

References

[1] - Dawn Xiodong Song, David Wagner, Adrian Perrig. Practical Technique for Searches on Encrypted Data. In proceedings of IEEE symposium on Security and Privacy, IEEE, 2000

[2] - Eu-Jin Goh. How to Search Efficiently on Encrypted Data. October 7, 2003

[3] - Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. Communication of the ACM, Vol 13 / Number 7 / pg 422-426, July 1970

[4] - Sarang Dharmapurikar. CSE 535: Lecture 5 – String Matching with Bloom Filters. Washington University, Fall 2003

[5] - Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone. Handbook of Applied Cryptography. CRC Press. October 1996 (Fifth Printing – August 2001)

[6] - William Stallings. Cryptography and Network Security: Principles and Practice (3rd Edition). Prentice Hall. August 27, 2002

[7] - Paul Garrett. Making, Breaking Codes: Introduction to Cryptology (1st Edition). Prentice Hall. August 9, 2000

[8] - Bruno R. Preiss. Data Structures and Algorithms with Object-Oriented Design Patterns in C#. http://www.brpreiss.com/ (E-book) 2001

[9] - Wikipedia Encyclopedia. http://en.wikipedia.org/wiki Wikimedia Foundation, Inc. Jan 2001

[10] - Peter K. Pearson. Fast Hashing of Variable-Length Text Strings. Communications of the ACM, Vol 33 / Number 6. June 1990

[11] - Eu-Jin Goh. Secure Indexes. May 5, 2004

[12] - Yedidyah Langsam, Moshe J. Augenstein, Aaron M. Tenenbaum. Data Structures Using C and C++. Prentice Hall 1996 (Second Edition)

[13] - Dan Boneh, Rafial Ostrovsky, Giovanni Di Crescenzo, Giuseppe Persiano. Public Key Encryption with Keyword Search. In proceedings of Eurocrypt 2004, LNCS 3027, pp. 506-522, 2004

[14] - Benny Chor, Oded Goldreich, Eyal Kushilevitz, Madhu Sudan. Private Information Retrieval, April 21 1998

[15] - Oded Goldreich, Rafail Ostrovsky. Software Protection and Simulation on Oblivious RAMs

Appendices

Appendix A *Mathematic of XOR*

The XOR function is as shown

0	⊕	0	=	0
0	⊕	1	=	1
1	⊕	0	=	1
1	⊕	1	=	0

Figure A1: XOR Function Characteristic

XOR function has a unique characteristic where if an XOR function is applied twice with any binary number, the original number will be obtained; therefore the function is its reverse function as well.

Example

Original Value	0	0	1	1	0	1	1	0
⊕	0	1	1	1	0	0	1	1
Intermediate Result	0	1	0	0	0	1	0	1
⊕	0	1	1	1	0	0	1	1
Original Value	0	0	1	1	0	1	1	0

Figure A2: XOR Function Example

Due to this characteristic, it is widely used in cryptology to obtain the original message from a encrypt one.

Appendix B *Mathematic of Pseudorandom Number Generator*

A pseudorandom number generator is used to generate a sequence of random number. Usually a seed (number) acts as an input to the generator to output a never-ending sequence of number. However, it is important to note that the numbers generated by the generator is periodic. The periodic cycle here may be very large but eventually the numbers will repeat itself. The seed is used to determine at which point it will begin to generate the numbers (Figure A3).

Here, the pseudorandom number generator is used where the periodic is very large to ensure that the number is harder to predict (next consecutive number).

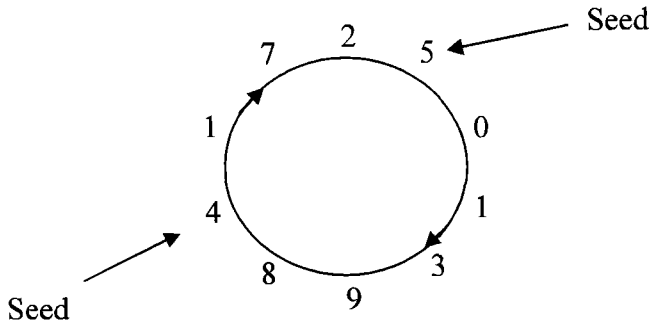
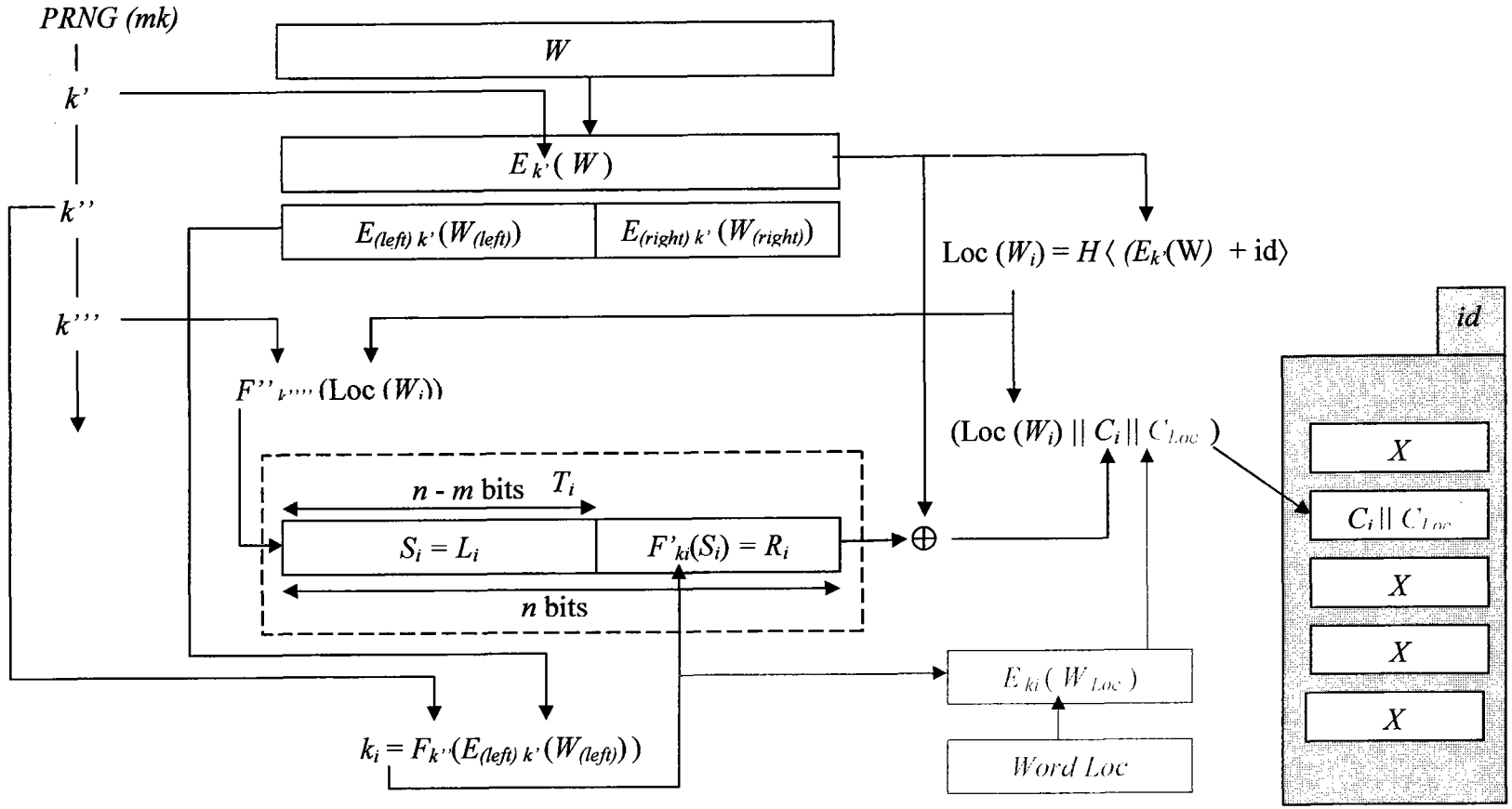
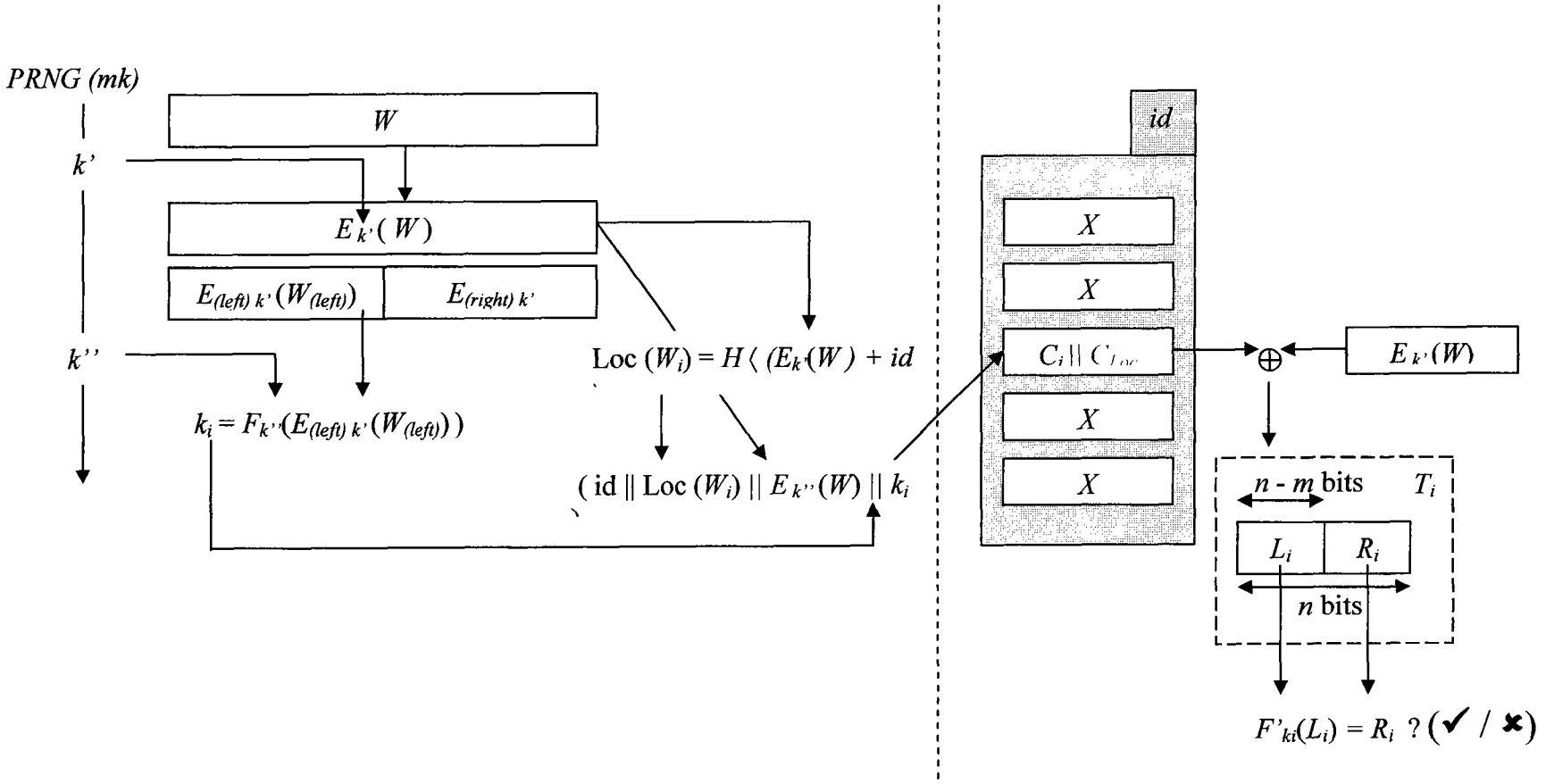


Figure A3: Pseudorandom Number Generator

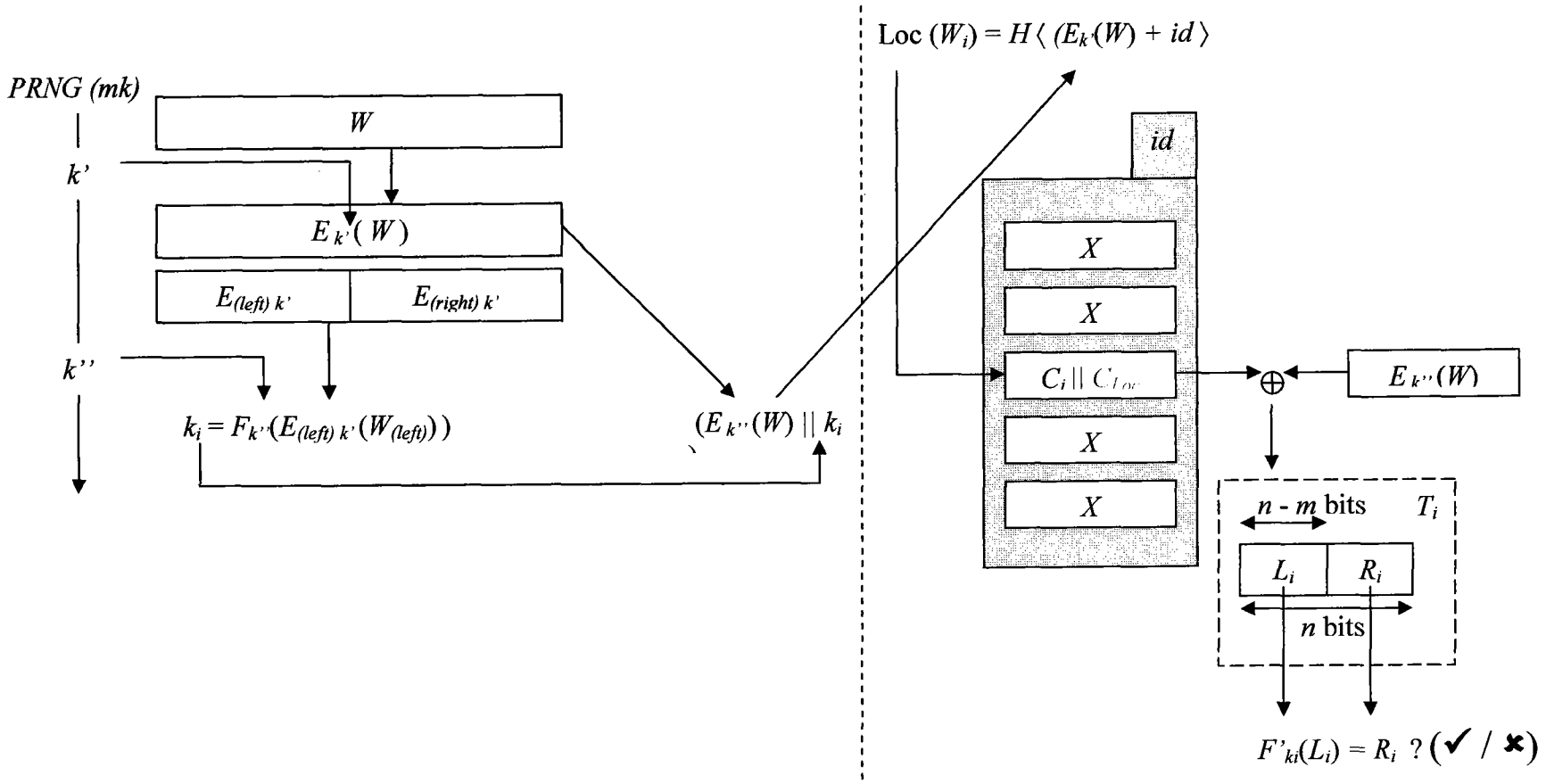
Appendix C *Scheme 1 – Setup / Encryption Phase*



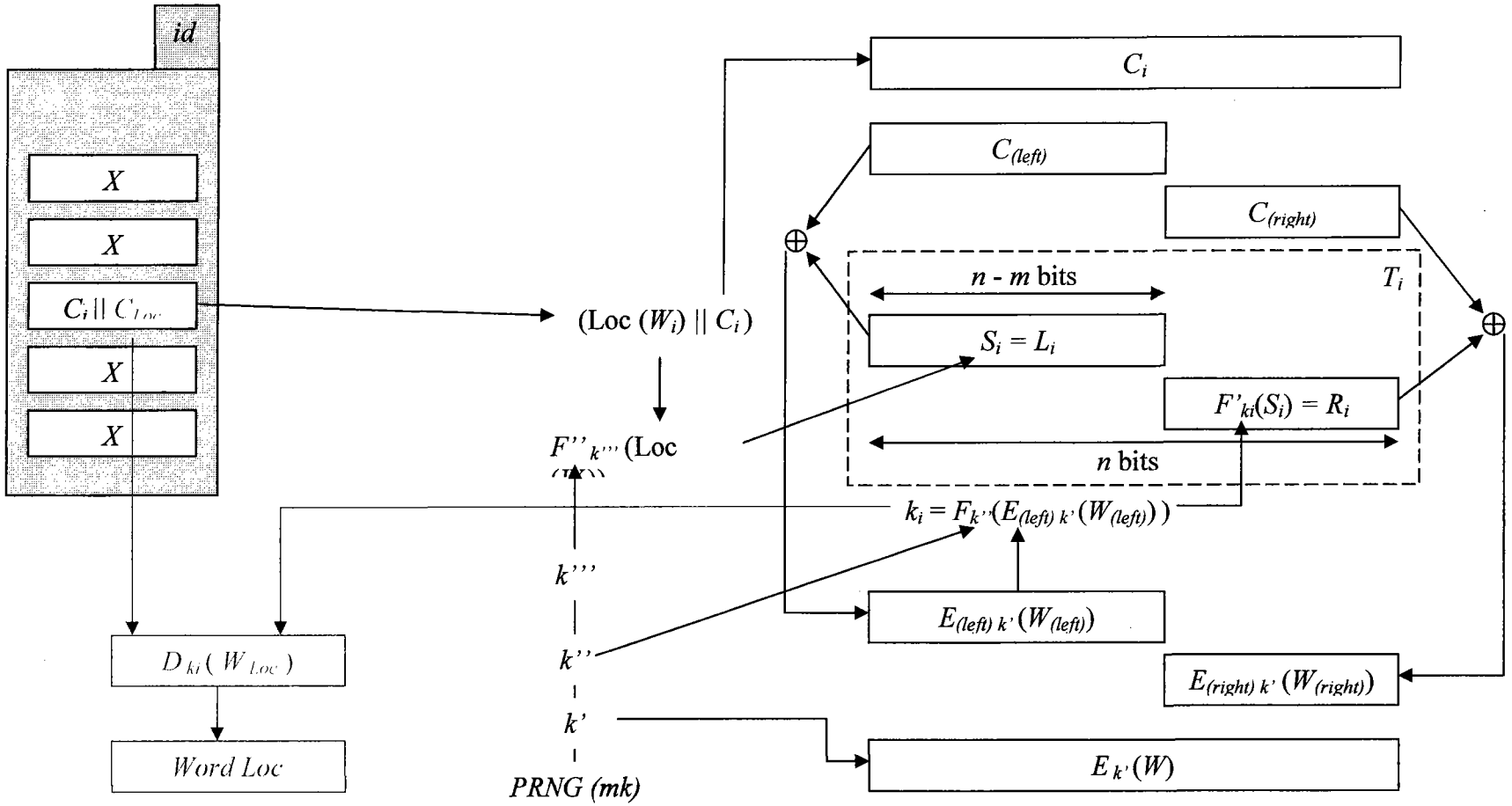
Appendix D *Scheme 1 – Search Phase (Single Mode)*



Appendix E *Scheme 1 – Search Phase (Multi Mode)*



Appendix F *Scheme 1 – Decryption Phase*



Cryptography is, traditionally, the study of ways to convert information from its normal, comprehensible form into an incomprehensible format, rendering it unreadable without secret knowledge the art of encryption. In the past, cryptography helped ensure secrecy in important communications, such as those of spies, military leaders, and diplomats. In recent decades, the field of cryptography has expanded its remit in two ways. Firstly, it provides mechanisms for more than just keeping secrets: schemes like digital signatures and digital cash, for example. Secondly, cryptography has come to be in widespread use by many civilians who do not have extraordinary needs for secrecy, although typically it is transparently built into the infrastructure for computing and telecommunications, and users are not aware of it.

The study of how to circumvent the use of cryptography is called cryptanalysis, or codebreaking. Cryptography and cryptanalysis are sometimes grouped together under the umbrella term cryptology, encompassing the entire subject. In practice, cryptography is also often used to refer to the field as a whole; crypto is an informal abbreviation.

Cryptography is an interdisciplinary subject, drawing from several fields. Before the time of computers, it was closely related to linguistics. Nowadays the emphasis has shifted, and cryptography makes extensive use of technical areas of mathematics, notably number theory, information theory, computational complexity, statistics and finite mathematics. It is also a branch of engineering, but an unusual one as it must deal with active, intelligent and malevolent opposition .

Associated fields are steganography the study of hiding the very existence of a message, and not necessarily the contents of the message itself and traffic analysis, which is the analysis of patterns of communication in order to learn secret information.

The original information which is to be protected by cryptography is called the plaintext. Encryption is the process of converting plaintext into an unreadable form, termed ciphertext, or, occasionally, a cryptogram. Decryption is the reverse process, recovering the plaintext back from the ciphertext. Enciphering and deciphering are alternative terms. A cipher is an algorithm for encryption and decryption. The exact operation of ciphers is normally controlled by a key — some secret piece of information that customises how the ciphertext is produced. Protocols specify the details of how ciphers are to be used to achieve specific tasks. A suite of protocols, ciphers, key management, user-prescribed actions implemented together as a system constitute a cryptosystem; this is what an end-user interacts with, like PGP or GPG.

In ordinary parlance, a "code" is often used synonymously with "cipher". In cryptography, however, the term has a specialised technical meaning: codes are a method for classical cryptography, substituting larger units of text, typically words or phrases. In contrast, classical ciphers usually substitute or rearrange individual letters. The secret information in a code is specified in a codebook.

"Cipher" is alternatively spelt "cypher"; similarly "ciphertext" and "cyphertext", and so forth. Both spellings have long histories in English, and there is occasional tension between their adherents.

A cryptanalyst might appear to be the natural adversary of a cryptographer, and to an extent this is true: one can view this contest all through the history of cryptography. However, it is possible, in fact preferable, to interpret the two roles as complementary: a thorough understanding of cryptanalysis is necessary to create secure cryptography.

There are a wide variety of cryptanalytic attacks, and it is convenient to classify them. One distinction concerns what an attacker can know and do in order to learn secret information, example does the cryptanalyst have access only to the ciphertext? Does he also know or can he guess some corresponding plaintexts? Or even: Can he choose arbitrary plaintexts to be encrypted?. While these example scenarios all view the cipher as an abstract black box, other attacks are based on the implementation of the cipher. If a cryptanalyst has access to, for example, timing or power consumption, he may be able to break a cipher otherwise resistant to analysis.

If a cryptosystem uses a key or a password, it is at risk from an exhaustive search; this is very commonly the weakest point in such systems. Linear and differential cryptanalysis are general methods for symmetric key cryptography. When cryptography relies on hard mathematical problems, as is usually the case in asymmetric cryptography, algorithms for tasks such as factoring become potential tools for cryptanalysis.

Appendix H SWP Linear Scan Pseudo code

Encryption

[Initialization]
1.1 Initialize Encryption Method
1.2 Initialize pseudorandom number generator with a seed

[Pre Processing]
2.3 Open the file to be encrypted and read every word
2.4 For (every word in the file, format the word)
2.4.1 Split long words
2.4.2 Split punctuation as single word
2.4.3 Insert paragraph marker
2.5 Put all formatted word into an array for encryption

[Encryption]
3.1 For (every word in array)
3.1.1 Every word W is encrypted $e(W)$ using selected encryption method
3.1.2 Split the encrypted word $e(W)$ into two halves left encrypted word $le(W)$ and right encrypted word $re(W)$
3.1.3 Generate a key kw using a cryptographic hash function with the left encrypted word $le(W)$
3.1.4 Generate a block of bits S with pseudorandom number
3.1.5 Generate another block of bits $F_{kw}(S)$ from cryptographic hash function with the key kw
3.1.6 Merge both blocks of bits to form another block of bits $(T= (S, F_{kw}(S)))$
3.1.7 Apply operator xor to block of bits T and encrypted word $e(W)$ to create cipher text C
3.2 Put all cipher text C into an array for output

[Post Processing]
4.1 For (every word in array)
4.1.1 Write into file

Search

[Initialization]
1.1 Initialize Encryption Method

[Pre Processing]
2.1 If (keyword long)
2.1.1 Split long keywords W
2.2 For(every keyword)
2.2.1 Encrypt the word $e(W)$
2.2.2 Generate a key kw using a cryptographic hash function with the left encrypted word $le(W)$
2.3 Put each pair, encrypted word $e(W)$ and key kw into an array for searching

[Search]
3.1 Open encrypted file and read every cipher text C
3.2 For (every cipher text in the file)
3.2.1 Apply operator xor with encrypted word $e(W)$ to generate block of bits T
3.2.2 Split the block of bits T into two halves left block of bits lS and right block of bits rS
3.2.3 Using the key kw and block of bits lS on a cryptographic hash to generate another block of bits $F_{kw}(S)$ 3.2.4 Compare both blocks of bit $F_{kw}(S)$ and rS
3.2.5 If (match)
3.2.5.1 Keep position number
3.3 Put all position number into an array as result

[Post Processing]
4.1 For (every position number in array)
4.1.1 Output to screen

Decryption

[Initialization]
1.1 Initialize Decryption Method
1.2 Initialize pseudorandom number generator with a seed

[Pre Processing]
2.1 Open the file to be decrypted and read cipher text
2.2 For (every cipher text in the file)
2.3 Put all formatted word into an array for encryption

[Decryption]
3.1 For (every cipher text in array)
3.1.1 Split the cipher text *C* into two halves left cipher text *lC* and right cipher text *rC*
3.1.2 Generate a block of bits *S* with pseudorandom number
3.1.3 Apply operator xor on *lC* and *S* to generate left encrypted word *le(W)*
3.1.4 Generate the key *kw* using a cryptographic hash function with the left encrypted word *le(W)*
3.1.5 Generate another block of bits $F_{kw}(S)$ from cryptographic hash function with the key *kw*
3.1.6 Apply operator xor on right cipher text rC with block of bits $F_{kw}(S)$ to generate *re(W)*
3.1.7 Merge both halves of encrypted Word to generate *(e(W)= (le(W), re(W))*
3.1.8 Decrypt the encrypted word *e(W)* to get word *W*
3.2 Put all word *W* into an array for output

[Post Processing]
4.1 For (every word in array, format the word)
4.1.1 Join long words
4.1.2 Join punctuation as single word
4.1.3 Insert paragraph marker
4.2 Write into file

Appendix I SWP Encrypted Index Pseudo code

Encryption

[Initialization]
1.1 Initialize Encryption Method

[Pre Processing]
2.1 Open the file to be encrypted and read every word
2.2 For (every word in the file, format the word)
2.2.1 Split long words
2.2.2 Destroy punctuation
2.2.3 Ensure only unique words are recorded
2.3 Put all formatted word into an array for encryption

[Encryption]
3.1 For (every word in array)
3.1.1 Every word *W* is encrypted *e(W)* using selected encryption method
3.1.2 Generate a key *kw* using a cryptographic hash function with the encrypted word *e(W)*
3.1.3 Encrypt the document number using key *kw* creating encrypted document number *e(docNo)*
3.1.4 Insert both encrypted word *e(W)* and *e(docNo)* into a hash table

[Post Processing]
4.1 Encrypt the original file using selected encryption method
4.2 Serialize the hash table into an index file

Search

[Initialization]
1.1 Initialize Encryption Method

[Pre Processing]
2.1 If (keyword long)
2.1.1 Split long keywords *W*
2.2 For (every keyword)
2.2.1 Encrypt the word *e(W)*
2.2.2 Generate a key *kw* using a cryptographic hash function with the encrypted word *e(W)*
2.3 Put each pair, encrypted word *e(W)* and key *kw* into an array for searching

[Search]
3.1 Open the index file and deserialize the hash table
3.2 Lookup in the hash table for the value of encrypted word *e(W)*
3.3 If (exist)
3.3.1 Decrypt the encrypted document number *e(docNo)* to get the document number *docNo*
3.4 Put all document number into an array as result

[Post Processing]
4.1 For (every document number in array)
4.1.1 Output to screen

Decryption

[Initialization]
1.1 Initialize Decryption Method

[Pre Processing]
2.1 Open the index file and deserialize the hash table

[Decryption]
3.1 For (every pair, encrypted word and encrypted document number in array)
3.1.1 Decrypt the encrypted word *e(W)* using selected decryption method
3.1.2 Generate a key *kw* using a cryptographic hash function with the encrypted word *e(W)*
3.1.3 Decrypt the encrypted document number *e(docNo)* using key *kw*
3.1.4 Write each pair, word *W* and document number *docNo* into an array

[Post Processing]
4.1 For (every pair, word and document number in array)
4.1.1 Output to screen
4.2 Decrypt the encrypted original file

Appendix J Goh Bloom Method Pseudo code

Encryption

[Initialization]
1.1 Initialize Hash Method
1.2 Initialize hash area as boolean array based on bucket size
1.3 Create r pseudorandom number keys

[Pre Processing]
2.1 Open the file to be encrypted and read every word
2.2 For (every word in the file, format the word)
2.2.1 Destroy punctuation
2.2.2 Ensure only unique words are recorded
2.3 Put all formatted word into an array for encryption

[Encryption]
3.1 For (every word in array)
3.1.1 For (every r keys)
3.1.1.1 Every word W is cryptographic hashed $h(W)$ using selected hash method using key r
3.1.2 Output all intermediate hash value into an array
3.1.3 For (every intermediate hash value in array)
3.1.3.1 Hash the intermediate hash value with docNo to create $h_{h(W)}(docNo)$
3.1.4 Output all final hash value into an array

[Post Processing]
4.1 For (every final hash value in array)
4.1.1 Write/Update all boolean array position hash value into files based on bucket size
4.2 Encrypt the original file using selected encryption method

Search

[Initialization]
1.1 Initialize Hash Method
1.2 Initialize hash area as boolean array based on bucket size
1.3 Create r pseudorandom number keys

[Pre Processing]
2.1 For (every r keys)
2.1.1 Every word W is cryptographic hashed $h(W)$ using selected hash method using key r
2.2 Output all intermediate hash value into an array
2.3 For (every intermediate hash value in array)
2.3.1 Hash the intermediate hash value with docNo to create $h_{h(W)}(docNo)$
2.4 Output all final hash value into an array

[Search]
3.1 For (every final hash value in array)
3.1.1 Open files based on bucket size
3.1.2 Retrieve boolean value from boolean array position hash value based on bucket size
3.2 If (all value true)
3.3 Put file number into an array as result

[Post Processing]
4.1 Output to screen

Decryption

[Initialization]
1.1 Initialize Decryption Method

[Pre Processing]

[Decryption]
3.1 Decrypt the File using selected decryption method

[Post Processing]

Appendix K Scheme 1 Method Pseudo code

Encryption

[Initialization]
1.1 Initialize Encryption Method
1.2 Initialize pseudorandom number generator with a seed
1.3 Generate the required sub keys

[Pre Processing]
2.3 Open the file to be encrypted and read every word
2.4 For (every word in the file, format the word)
2.4.1 Split long words
2.4.2 Destroy punctuation
2.5 Put all formatted word into an array for encryption

[Encryption]
3.1 For (every word in array)
3.1.1 Every word *W* is encrypted *e(W)* using selected encryption method
3.1.2 Split the encrypted word *e(W)* into two halves left encrypted word *le(W)* and right encrypted word *re(W)*
3.1.3 Generate the *Loc* for the location of the index using the encrypted word *e(W)* merged with *id*
3.1.4 Generate a block of bits *S* using a cryptographic hash function on the *Loc* value generated for indexing
3.1.3 Generate a key *kw* using a cryptographic hash function with the left encrypted word *le(W)*
3.1.5 Generate another block of bits $F_{kw}(S)$ from cryptographic hash function with the key *kw*
3.1.6 Merge both blocks of bits to form another block of bits $(T= \langle S, F_{kw}(S) \rangle)$
3.1.7 Apply operator xor to block of bits *T* and encrypted word *e(W)* to create cipher text *C*
3.2 Put all cipher text *C* into the array based on *Loc* for output

[Post Processing]
4.1 Serialize the array into an index file
4.2 Encrypt the original file using selected encryption method

Search

[Initialization]
1.1 Initialize Encryption Method

[Pre Processing]
2.1 If (keyword long)
2.1.1 Split long keywords *W*
2.2 For(every keyword)
2.2.1 Encrypt the word *e(W)*
2.2.2 Generate a key *kw* using a cryptographic hash function with the left encrypted word *le(W)*
2.3 Generate *Loc* using the encrypted word *e(W)* merged with *id*
2.4 Put together, index location *Loc*, encrypted word *e(W)* and key *kw* into an array for searching

[Search]
3.1 Open the index file and deserialize the array index
3.2 Lookup in the index for the value of encrypted word *e(W)* using *Loc*
3.3 If (exist)
3.3.1 Apply operator xor with encrypted word *e(W)* to generate block of bits *T*
3.3.2 Split the block of bits *T* into two halves left block of bits *lS* and right block of bits *rS*
3.3.3 Using the key *kw* and block of bits *lS* on a cryptographic hash to generate another block of bits $F_{kw}(S)$ 3.3.4 Compare both blocks of bit $F_{kw}(S)$ and *rS* for a match

[Post Processing]
4.1 For (every document number in array)
4.1.1 Output to screen

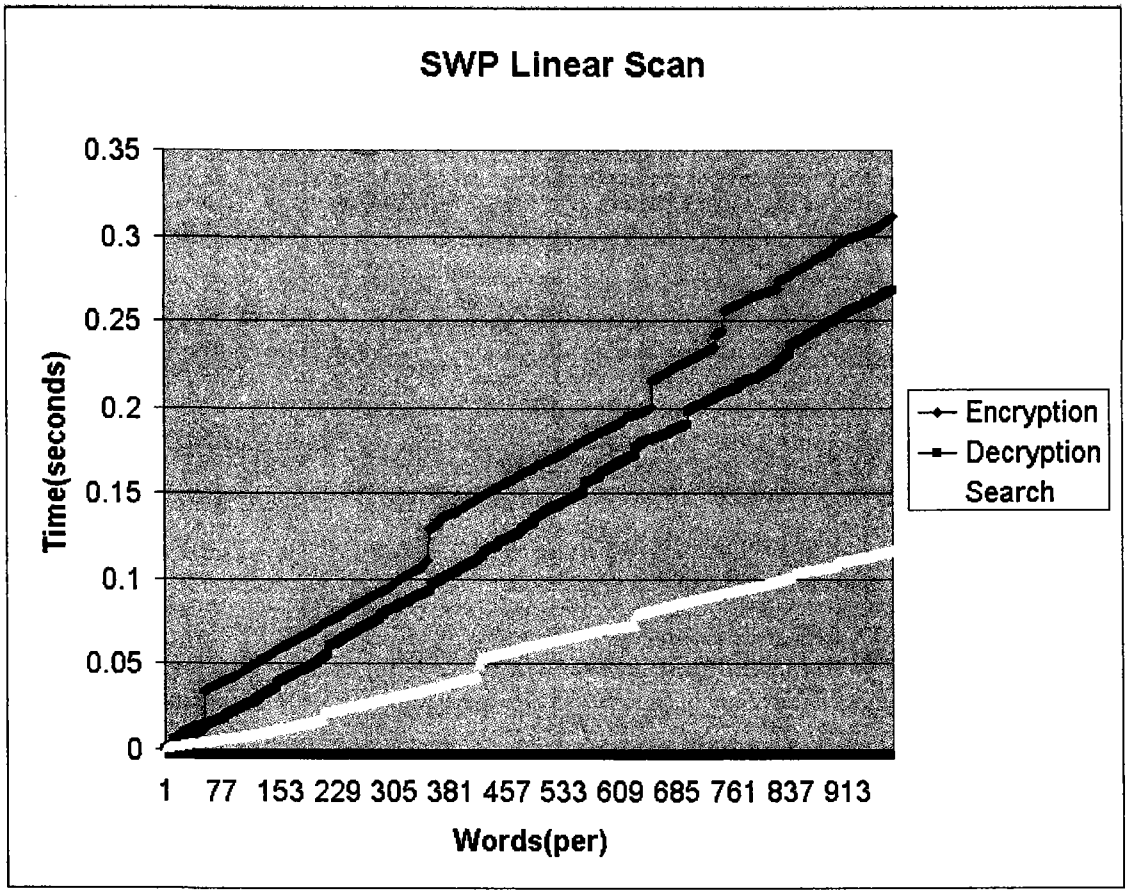
Decryption

[Initialization]
1.1 Initialize Decryption Method
1.2 Initialize pseudorandom number generator with a seed
1.3 Generate the required sub keys

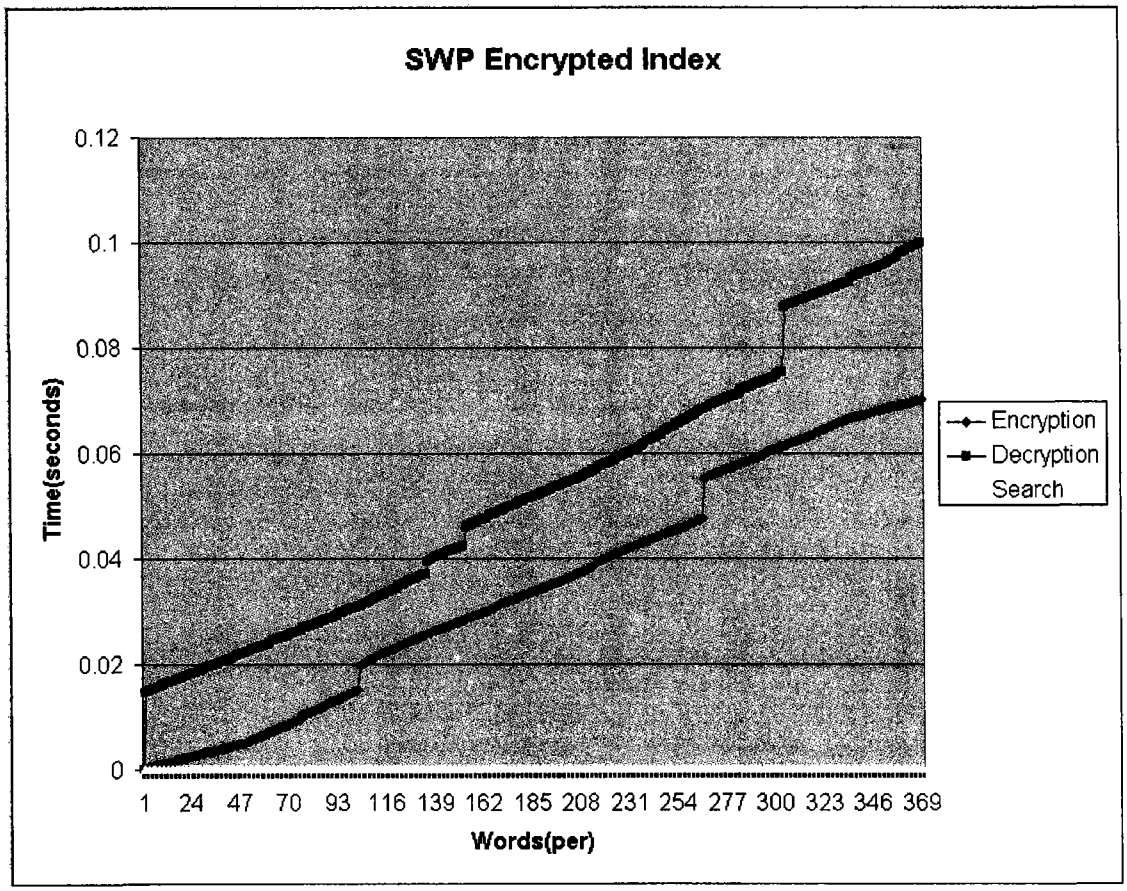
[Pre Processing]
2.1 Open the index file and deserialize the index array

[Decryption]
3.1 For (every encrypted word in array)
3.1.1 Split the cipher text *C* into two halves left cipher text *lC* and right cipher text *rC*
3.1.2 Generate a block of bits *S* using a cryptographic hash function with the *Loc* based on the index
3.1.3 Apply operator xor on *lC* and *S* to generate left encrypted word *le(W)*
3.1.4 Generate the key *kw* using a cryptographic hash function with the left encrypted word *le(W)*
3.1.5 Generate another block of bits $F_{kw}(S)$ from cryptographic hash function with the key *kw*
3.1.6 Apply operator xor on right cipher text rC with block of bits $F_{kw}(S)$ to generate *re(W)*
3.1.7 Merge both halves of encrypted Word to generate *(e(W)= (le(W), re(W))*
3.1.8 Decrypt the encrypted word *e(W)* to get word *W*
3.2 Put all word *W* into an array for output

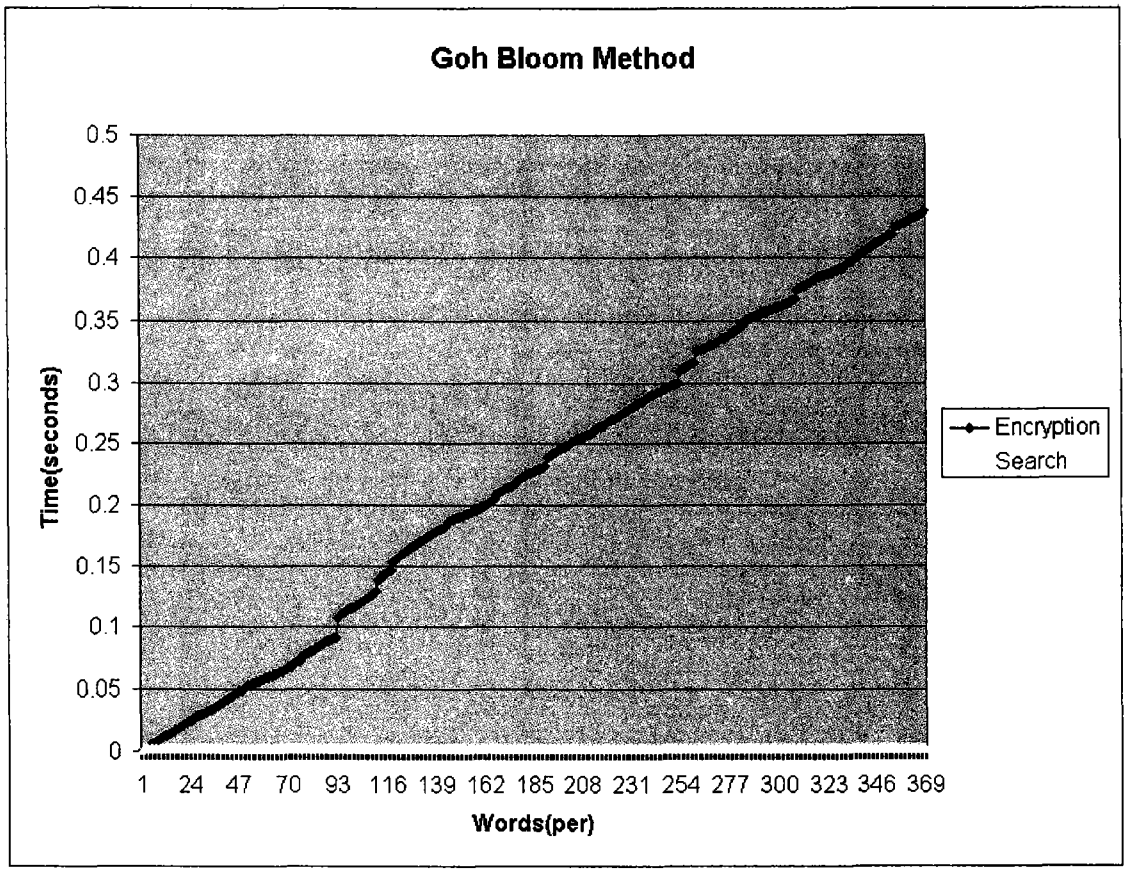
[Post Processing]
4.1 For (every word in array)
4.1.1 Join long words
4.2 Write into file



	No of Words = 980 **Encryption(seconds)**	No of Words = 980 **Decryption(seconds)**	No of Words = 980 **Search(seconds)**
Prep Time	0.207021562246983	0.00071693719397000 4	0.00043308478235725
Post Time	0.00137466322242356	0.00530519335312488	
Processing Time	0.311217135	0.267862917	0.11645776
Average Time (per word)	0.000319525	0.000275013	0.00011956648866203 10

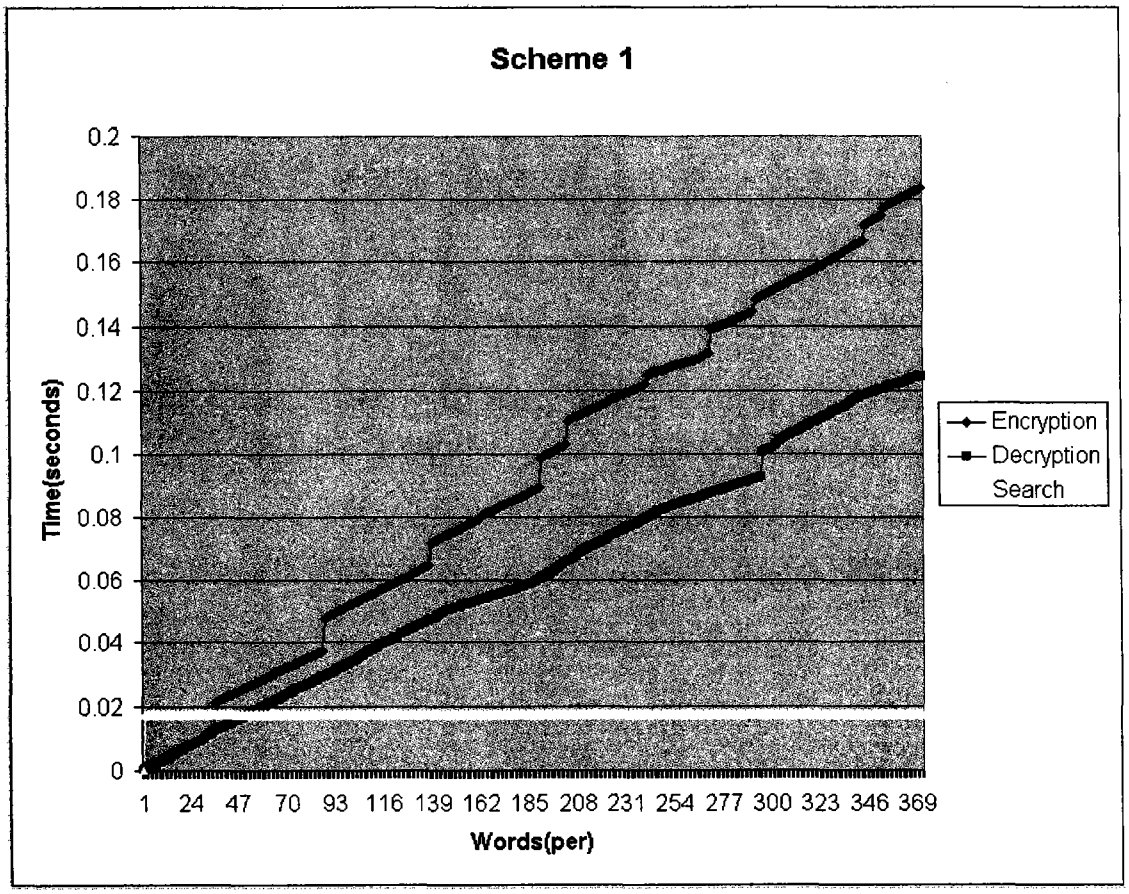


	No of Words = 375 **Encryption(seconds)**	No of Words = 375 **Decryption(seconds)**	**Search(seconds)**
Prep Time	0.00134303890195474	0.00293470331958206	0.00061659559702355 7
Post Time	0.0146203045841848	0.00652851756178432	
Processing Time	0.07026838	0.099963504	0.00006335351042911 67
Average Time (per word)	0.000190429	0.00027090380453868 10	

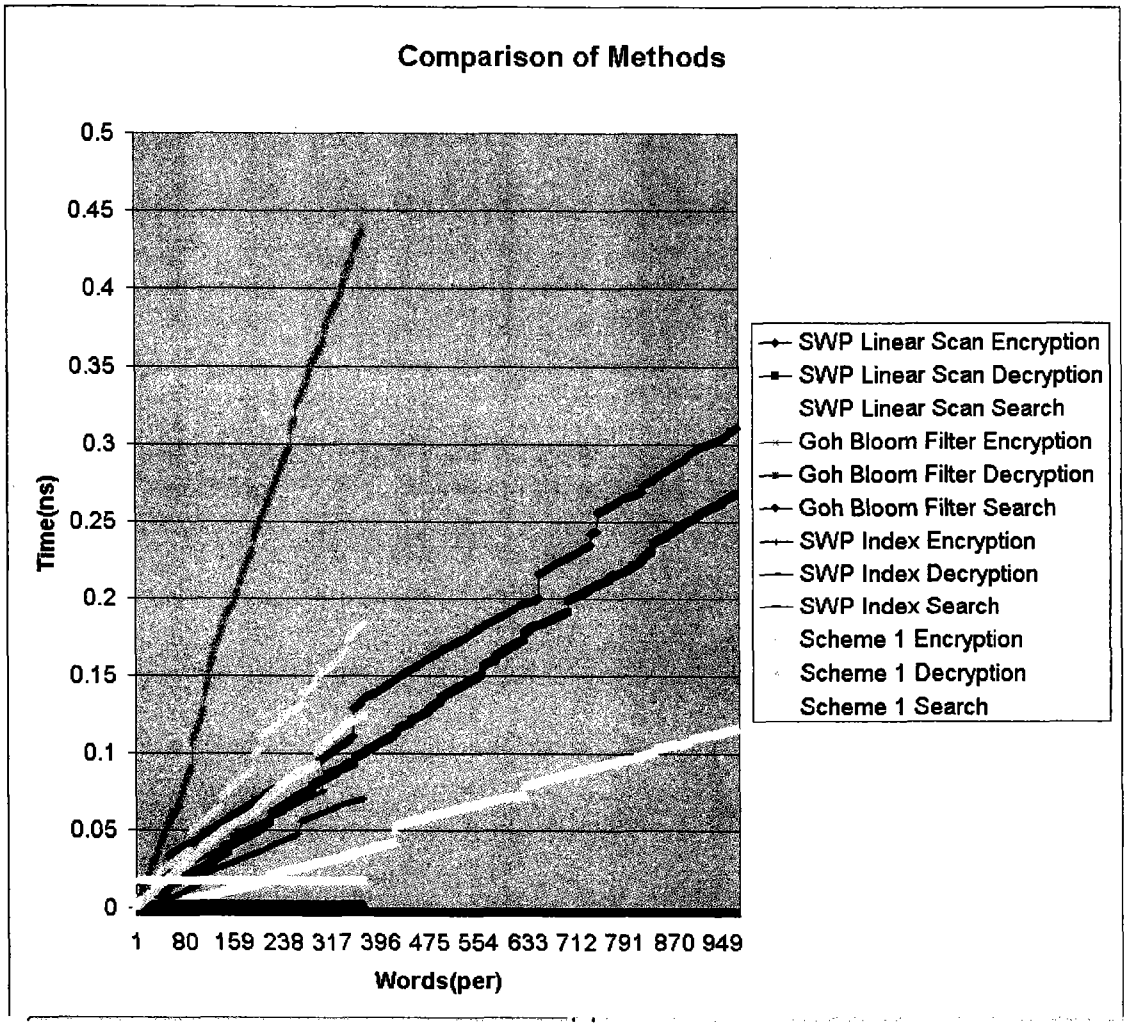


No of Words = 720

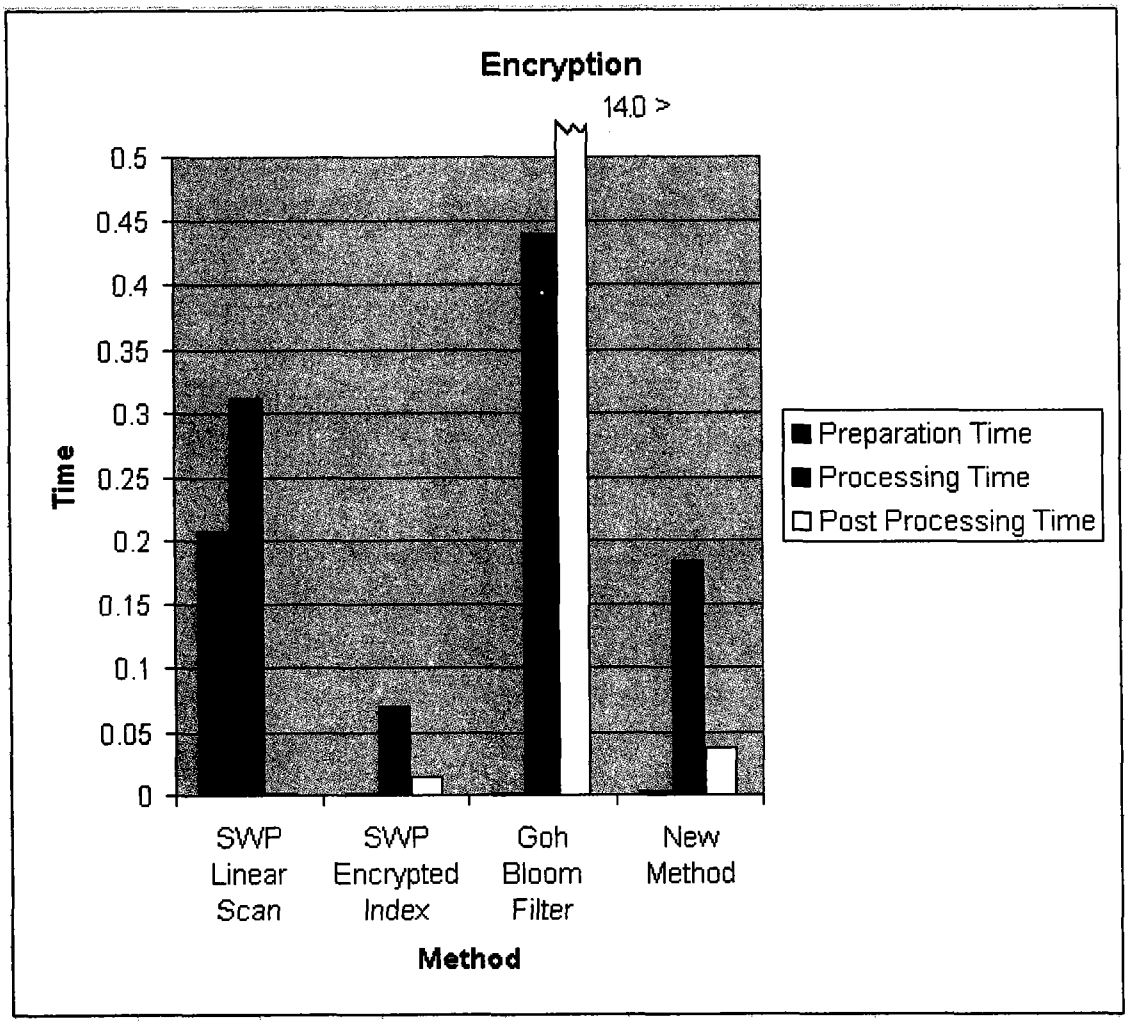
	Encryption(seconds)	Decryption(seconds)	Search(seconds)
Prep Time	0.0006430007865151 21		0.00117076939685086
Post Time	31.1854238483722	0.107297944249528	
Total Time	0.586498543		0.00014381
Avg Time(per word)	0.00081799		

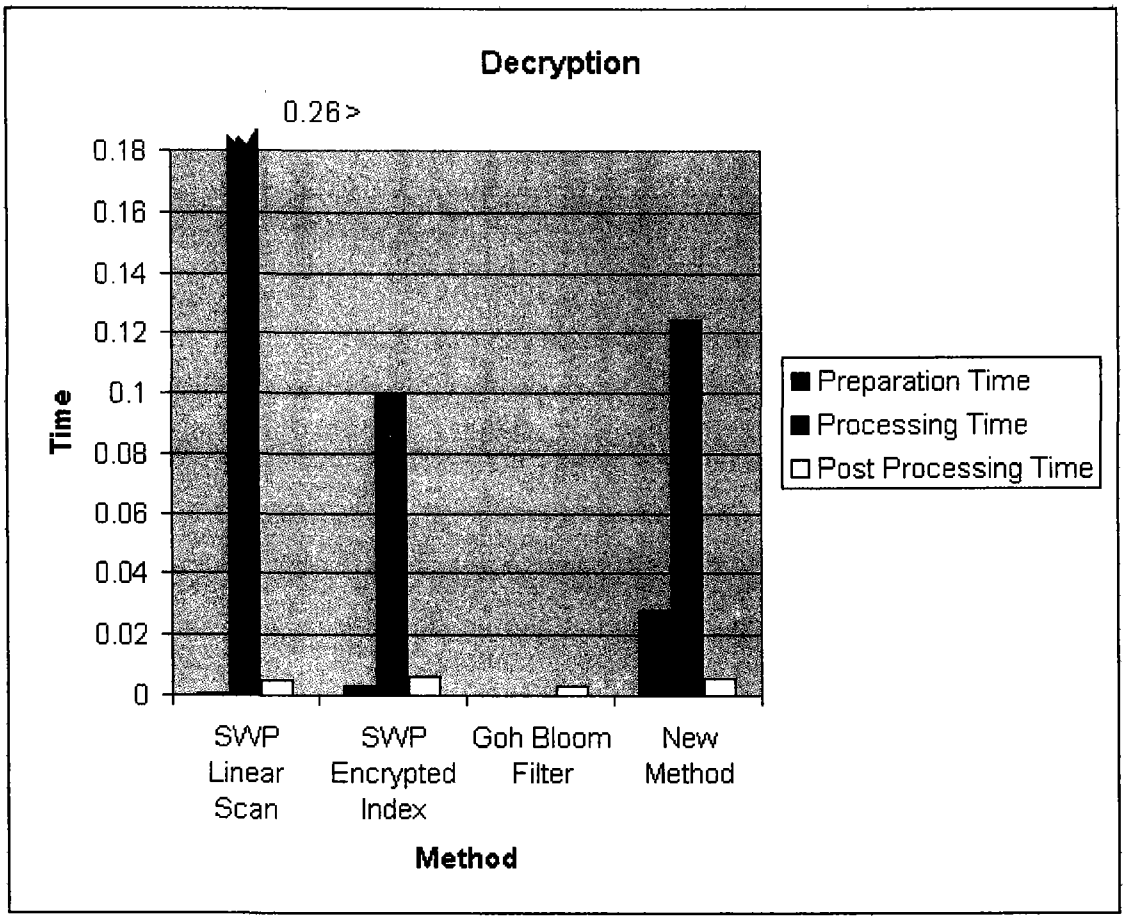


	No of Words = 375 Encryption(seconds)	No of Words = 375 Decryption(seconds)	Search(seconds)
Prep Time	0.00325021243783013	0.0284156517716004	0.00025845086170335 8
Post Time	0.0366460161159733	0.00567701893048541	
Processing Time	0.183642046	0.1246849	0.017614753
Average Time (per word)	0.000494992	0.00033607789639980 80	

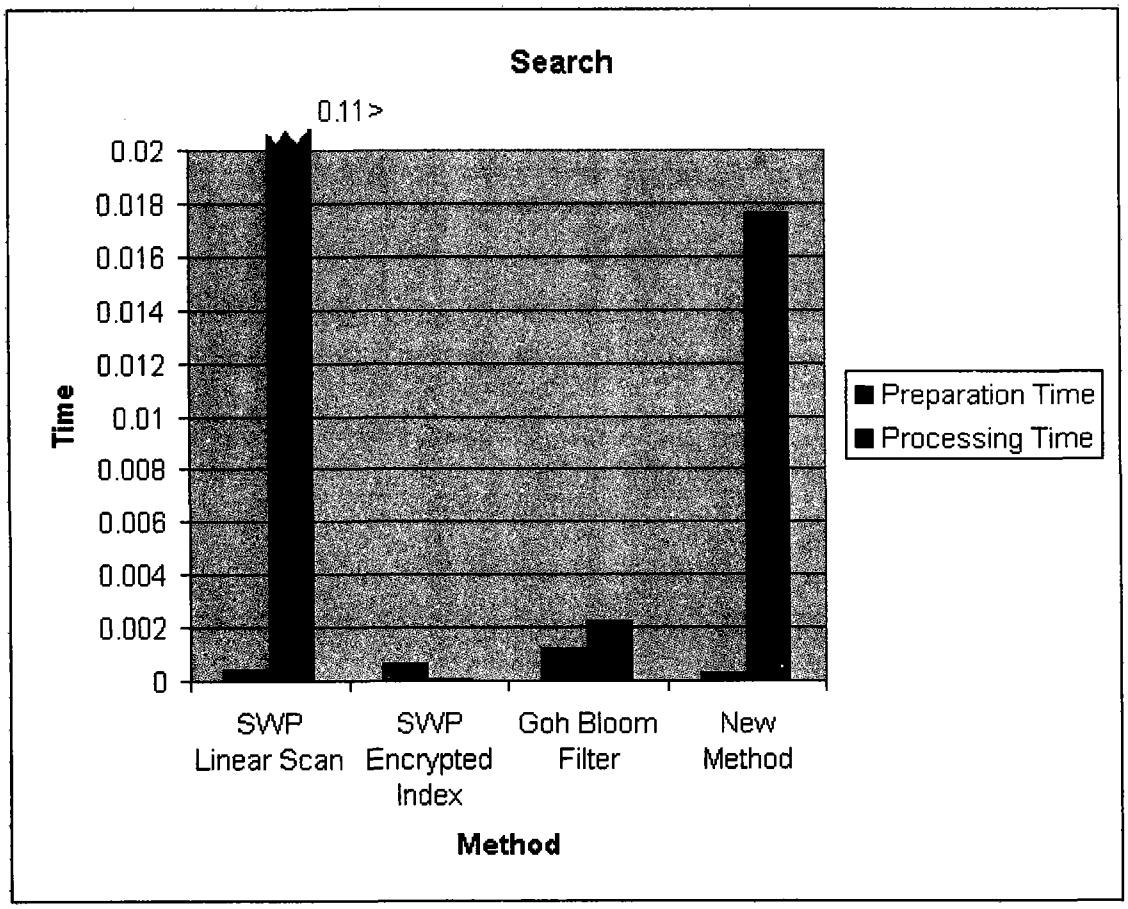


Total Time	Encryption(seconds)	Decryption(seconds)	Search(seconds)
SWP Linear Scan	0.51961	0.27389	0.11689
SWP Encrypted Index	0.08623	0.10943	0.00068
Goh Bloom Filter	14.57684	0.00315	0.00344
Scheme 1	0.22354	0.15878	0.01787

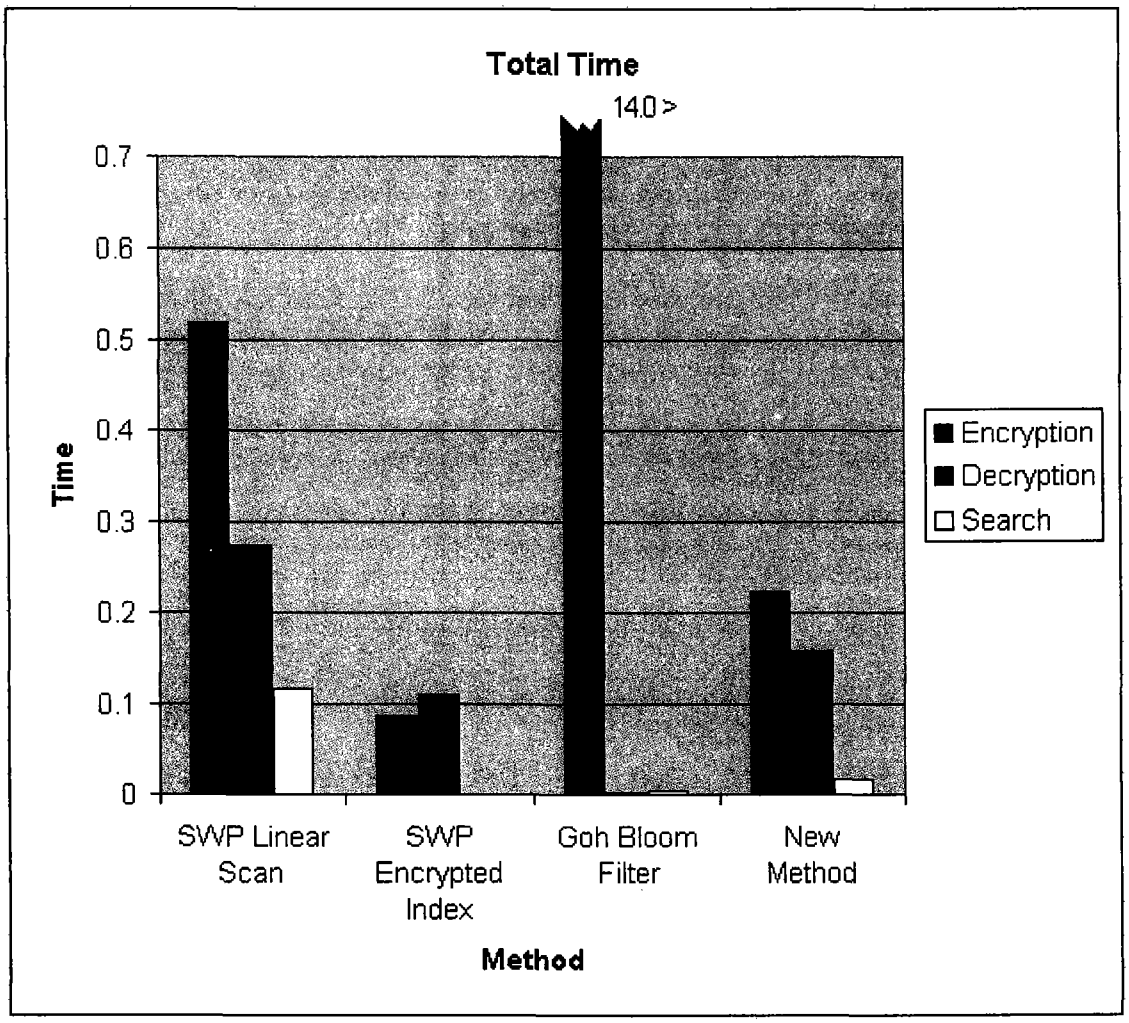

Encryption
14.0 >
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
Time
SWP Linear Scan
SWP Encrypted Index
Goh Bloom Filter
New Method
Method
Preparation Time
Processing Time
Post Processing Time


Decryption
0.26 >
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
Time
SWP Linear Scan
SWP Encrypted Index
Goh Bloom Filter
New Method
Method
Preparation Time
Processing Time
Post Processing Time

Appendix S *Comparison of Methods (Search)*



Appendix T *Comparison of Methods (Total Time)*



PUBLISHED PAPERS

PUBLISHED PAPERS

Searchable Public Key Encryption

Lai Kien Mun and Azman Samsudin
Computer Science School, University Science Malaysia,
11800, Penang, Malaysia.
P:(+60)04 6533888 x3617, F:(+60)04 6573335
Email: azman@cs.usm.my

Abstract

Nowadays it is normal for users to store their confidential information in data storage server such as email and file servers. These data storage server must be fully trusted for not revealing the content without authorization. To ensure the integrity of the stored data, the data is encrypted before placing it on data storage server. With file encrypted on data storage server, it is difficult for user to retrieve certain file based on their content. Using the traditional methods, searching on the encrypted files require decrypting every files. Our proposed solution is to form a mechanism which allows data storage server to search for specific keyword in the encrypted data and return only relevant information to the user, while ensuring that data storage know nothing about the encrypted content.

1. Introduction

Nowadays, email and file servers play major roles as data storage servers for organizations and individuals. To increase information integrity stored on the data storage servers, the information stored is normally encrypted. Therefore, only the authorized user can recovered the encrypted information. However, this makes it harder for the legitimate user to retrieve certain information, especially if the information is stored on a different server. Traditionally, searching for a piece of information will require decrypting all related files. For example, suppose Bob encrypts his email to Alice using Alice's public key and then send the encrypted email to Alice. Assume Alice wants her email gateway to route only those emails with the keyword 'urgent' to her first for priority reading. Since all the emails are encrypted, the email gateway is not able to search for the keyword and therefore unable to make the necessary routing decision. As a result, Alice need to download all her emails to her local machine, decrypt them, and then search for the specific keyword herself.

This method is inefficient and inconvenient because it requires decrypting all the emails, regardless of the keyword Alice is looking for. Our ideal solution is to construct a mechanism which allows the email gateway or data storage searches for specific keyword in the encrypted email or encrypted data and return only relevant ones to the user, while ensures the server learns nothing about the encrypted content.

In our approach we are focusing on email server where Bob will encrypt his email using Alice's public key before sending the email to the email server. Bob will first encrypts his email using a session key that is randomly generated, $E_{sessionkey}(M)$. The session key will then be encrypted using Alice's public key, $E_{Apub}(sessionkey)$. Alice is the only person who is able to decrypt the encrypted session key using her private key during decryption later. Then, each keyword for the subject of the email will be encrypted using our searchable public key encryption (SPKE) approach. The resulting cipher text will be appended to the encrypted message M, $E_{sessionkey}(M)$. For example, to send a message M with keywords $W_1, W_2, ..., W_m$, Bob sends

$$E_{sessionkey}(M) \quad || \quad E_{Apub}(sessionkey) \quad || \quad SPKE(A_{pub}, W_1) \; || \; ... \; || \; SPKE(A_{pub}, W_m)$$

Where A_{pub} is Alice's public key. Let assume Alice wishes to search for the keyword W, she send to the email gateway a Trapdoor information, $T_w(W', K_s)$, where K_s refer to temporary session key. Trapdoor information enables the gateway to identify her encrypted email through one of the keywords associated with the message by checking if one of the keywords is equal to the word W of Alice's choice. Given a searchable encryption $W_{SKPE} = SPKE(A_{pub}, W_m)$

and T_w, the gateway can test to identify whether $W_{SKPE} = E_{Ks}(W')$. If $W_{SKPE} \neq E_{Ks}(W')$, the gateway learns nothing about W_m. This is a noninteractive searchable public key encryption mechanism where no communication between Alice and Bob is required for the entire process. All that is needed is for Bob to generate the searchable encryption for W_m with the given Alice's public key.

2. Related Work

There are some research works that had been done to solve the problem on searching encrypted data (or emails) in an untrusted data storage servers [1,2,3]. The details on some of the solutions are discussed in this section.

2.1 Practical Techniques for Searches on Encrypted Data (SSKE)

Song [2], studies the problem of searching data which initially encrypted with a symmetric key setting. He proposed an idea on how to support searching functionality on encrypted data without losing data confidentially by using, *sequential scan* on the entire encrypted documents. A search for a specific keyword W will return all the positions where W occurred in the plaintext, as well as possibly some other erroneous positions [2]. His technique provides secrecy for encryption and controlled searching where every keyword searched by untrusted server needs to get user's authorization. Song's technique used pseudorandom function to generate a sequence of pseudo random values $S_1, \ldots, S_m$ using stream cipher.

There are four main operations in SSKE mechanism as shown as below:

Key Generation: Pseudorandom function is used to generate a sequence of pseudorandom values $S_1, \ldots, S_t$ using some stream ciphers S_t. Alice takes the pseudorandom bits S_i from the n-bits word, W_i which appear in position i. To encrypt the word W, Alice sets $T_i := \{S_i, F_k(S_i)\}$ and the result is cipher text $C_i := W_i \oplus T_i$. Nobody can decrypt the cipher text except Alice because she is the only person who is able to generate the pseudorandom stream $T_1, \ldots, T_t$.

Encryption: In Song's technique, each document is divided into 'words', which depends on the application domain of interest. In this case, Song assumes that these 'words' have the same length. This can be done by either pad the shorter 'words' or split the longer 'words' to get a same equal length for each 'words'.

Let assume Alice would like to search for a keyword W but she is not willing to reveal W to the untrusted server. So, she will encrypt a document which contains the sequence of words $W_1, \ldots, W_t$. Alice encrypts W_i using a deterministic encryption algorithm E with her secret key (key a) to get a result, $E_a(W_i)$. After the encryption phase, Alice has a sequence of E-encryption words, $E_a(W_1), \ldots, E_a(W_t)$. $E_a(W_i)$ which then split into two parts, $E_a(W_i) = \{L_i, R_i\}$, where L_i denotes the first $n-m$ bits of $E_a(W_i)$. R_i denotes the last m bits of $E_a(W_i)$.

A standard Pseudo Random Function (PRF), F is used to derive key K_1 with a key b and L_i, $PRF_{key\ b}(L_i) = K_i$. Another pseudorandom bits S_i is generated using a standard Pseudo Random Generator (PRG) based on the location of W_i, where S_i is $n - m$ bits long, same as L_i. Use a PRF with key K_i which generated from $PRF_{key\text{-}b}(L_1)$ to pad with S_i to produce stream cipher SC_i. Operate bitwise exclusive-or (XOR) between $E(W_i)$ and SC_i to produce the cipher text of W_i.

Generate Trapdoor: Consider Alice wishes to search for a specific keyword KW, she sends to the untrusted server a certain trapdoor information, T_{kw}, that enables the server to test whether each encrypted documents contains the keyword KW without knowing anything else. The trapdoor information of Song's technique is set as $T_{kw} = \{E_a(KW), K_{kw}\}$.

Test: Alice send the K_{kw} and $E_a(KW)$ to the untrusted server. Server performs sequential scan on the entire encrypted keywords of the documents, $C_1, \ldots, C_t$. If $C_i = E_a(KW)$, then $V' = F_{kkw}(S_i)$. The server will test if $F_{kkw}(S_i) == V^*$, the output 'yes' if true and 'no' otherwise.

2.2. Secure Indexes for Efficient Searching on Encrypted Compressed Data

Goh's paper [3] focuses on building secret keyword indexes using Bloom filters [4] and pseudorandom function that allow efficient search on encrypted compressed documents. Refer to Goh's definition, a bloom filters represents a set of $S = \{s_1, \ldots, s_n\}$ of n elements and is represented by an array of m bits. The filter uses r independent hash function $f_1, \ldots, f_r$, where $f_i : \{0,1\}^* \rightarrow [1,m]$ for $1 \leq i \leq r$. For each element $s \in S$, the array bits at positions $f_1(s), \ldots, f_r(s)$ are set to 1. A location can be set to 1 multiple times, but only the first is noted.

2.3. Searchable Public Key Encryption (SPKE)

Boneh [1] have studied the problems of searching on data that is encrypted using public key encryption system. They had implemented a Searchable Public Key Encryption mechanism based on the Identity-

Based Encryption scheme. Boneh's scheme is used as reference and foundation to devise our own Searchable Public Key Encryption model.

Searchable Public Key Encryption [1] is defined as searching for a specific keyword on encrypted data using a public key system, but learns nothing about the data and searching keyword. The public key refers to the fact that most people encrypted their sensitive documents using public key before sending it to untrusted server. The motivation of Boneh's idea was to allow an email gateway to prioritize encrypted email by certain keywords. Consider user Bob encrypted his email before send to Alice with keywords $W_1, \ldots, W_k$ and using Alice's public key, A_{pub}. Assume sender name and all words in email subject line will be used as keywords. Bob sends the following message:

$$E_{Apub}(M) \parallel SPKE(A_{pub}, W_1) \parallel \ldots \parallel SPKE(A_{pub}, W_m)$$

Where M is the email body, and SPKE is an algorithm. The SPKE values do not reveal the secrecy of the encrypted message, but enable searching for specific keywords [1].

Boneh and his team defined four polynomial time randomized algorithm of a noninteractive public key searchable encryption schemes. First, Alice generates her public/private key pair (A_{pub}, A_{priv}) by executing the KeyGen algorithm using a security parameter, s. Many people know her public key and at the same time she has to keep her private key confidential. Each word W in the email subject line that send to Alice will be encrypted using Alice's public key A_{pub} to produce a searchable encryption of W' when running using SPKE algorithm. The resulting W' then will be appended to the entire email content which encrypted with standard encryption algorithm in advance before sending to email server. To search for any encrypted keyword W, Alice uses Trapdoor algorithm to generate trapdoor T_w. The email server uses the resulting trapdoor as input to the Test algorithm to locate all encrypted emails which contains one of the keywords W specified by Alice.

We call this scheme as noninteractive public key searchable encryption because there is no communication or information exchange between the sender and receiver during the email encryption process and keyword searching process. Alice is able to ask the untrusted server to locate all her encrypted emails that sent by others, which contains specific keywords by sending a short secret key T_w. The server then sends the relevant emails to Alice, without knowing the content of her email.

SPKE is consider secure where attacker is unable to obtain any information about W or search for a keyword W' without going through trapdoor process. Searchable Public Key Encryption is related to Identity Based Encryption (IBE) [5]. Boneh and his team had given three constructions for Searchable Public Key Encryption (SPKE) based on recent Identity Based Encryption (IBE). They are using different idea (bilinear maps, Jacobi symbols and trapdoor permutation) to construct each of their SPKE model.

3. Methodology

3.1. System Architecture

Our proposed scheme solves the problem of searching on data (in our case, we refer to email) that is encrypted using a public key encryption system. Assume Bob wishes to send a sensitive email to Alice. Using proposed scheme the searching for the keyword only focus on each keyword for the subject of the email. Bob will encrypt his email using a session key which generated using AES key generator function. Then, the session key is encrypted using standard public key encryption with Alice's public key, A_y. To send a document M with subject keywords W_1, W_2, ..., W_m, the following structure is sent to the server.

$$E_{sessionkey}(M) \parallel E_{A1y}(sessionkey) \parallel SPKE(A2_y, W_1) \parallel SPKE(A2_y, W_2) \parallel \ldots \parallel SPKE(A2_y, W_m)$$

In our scheme, there are two pairs of public-private keys involved. First pair of public-private key ($A1_y$, $A1_x$) is used to encrypt and decrypt email content, while second pair of public-private key ($A2_y$, $A2_x$) is used for SPKE keywords in email subject field. Our proposed scheme will use both declarative and procedural programming. Declarative programming requires no instruction and it consists only integer, facts, rules or relationship where the need to consider when and how the information (such as integers, facts, rules and relationship) is being applied do not arise. Database is an example of declarative programming. As for procedural programming, instructions are used to extract and use the data from the database. There are four main algorithms in our SPKE scheme: Key Generation, SPKE Encryption, Trapdoor Generation and Test for Specific Keyword.

3.2. Key Generation

There are 2 pairs of public-private keys involved. First pair of public-private key ($A1_y$, $A1_x$) is used to encrypt and decrypt email content, while the second

pair key ($A2_y$, $A2_x$) is used for SPKE keywords in email subject field. First pair of public-private key can easily be generated by using RSA algorithm. To generate the second pair of public-private key, Alice selects a large prime number p and an element q, which $2 <= q <= p - 2$, that generates a cyclic or subgroup of large order. These values are initially determined and known by other users. Bob will encrypt sensitive document using session key that randomly generated and then session key is encrypted using Alice public key, $A1_y$. Alice is the only person who can decrypt the session key using her private key, $A1_x$. We use Diffie-Hellman algorithm to generate Alice's another public-private key pair ($A2_y$, $A2_x$) which both key are real number. Alice chooses her private key $A1_x$, at random from the set $\{1, \ldots, p\text{-}2\}$. Alice's public $A2_y$ is computed $A2_y = q^{A2x} \bmod p$. Private key $A1_x$ and $A2_x$ only known by Alice and this key will be used during decryption.

3.3. SPKE Encryption

Suppose Bob wishes to send email which contain M to Alice with the subject keywords $W_1, W_2, .. W_n$. Our SPKE encryption process goes through the following procedures to produce a searchable encrypted document;

1. First, a session key, $key_{session}$ is generated using AES key generator function.
2. Email contents, M are encrypted using session key, $E_{keysession}(M)$.
3. Encrypt session key, $key_{session}$ with Alice's public key, E_{A1y} ($key_{session}$).
4. Each subject keyword $W_1, W_2, .. W_n$ is hashed using MD5 hash function. MD5 take an arbitrarily sized block of data as input, W_i (in our case is word), and produce a 128-bit (16-bytes) message digest. Each byte of the resulting messages digest will *XOR*ed one another to produce 8-bit output W'_i. Then, $W'_1, W'_2, .. W'_n$ will send to SPKE engine to produce searchable cipher text, $C_{W1}, C_{W2}, .. C_{Wn}$ where $C_{wi} = (A2_y)^{Wi'} \bmod p$
5. The resulting output from procedure 1, 2 and 4 is padded. To send message with subject keyword $W_1, W_2, .. W_n$, Bob send
   E_{A1y} ($key_{session}$) || $E_{keysession}(M)$ || C_{W1} || C_{W2} || ... || C_{Wn}

3.4. Trapdoor Generation

If Alice wishes to search for a specific keyword, N from her encrypted message which store on server. To enable server locates all related messages to her, Trapdoor Generation is used to generate a trapdoor T_N for keyword N. The detail of the process showed as below.

1. Keyword N is hashed using MD5. An output 128-bit (16-bytes) message digest is produced. Each byte of the resulting messages digest will be *XOR*ed one another to produce 8-bit output N'.
2. Compute for $K = (q)^{N'} \bmod p$ (p and q value is used in Key Generation)
3. Select a session key, s
4. Encrypt output K with session key, $E_s [(K)^{A2x} \bmod p]$
5. Send $E_s [(K)^{A2x} \bmod p]$ and session key, s as the trapdoor of keyword N to server. $T_N = [E_s [(K)^{A2x} \bmod p], s]$

3.5. Test for Specific Keyword

After receiving trapdoor for keyword $T_N = [E_s [(K)^{A2x} \bmod p], s]$, server will perform searching process. Let $T_N = [A, B]$ where $A = E_s [(K)^{A2x} \bmod p]$, $B = s$.

1. First, email agent on the server retrieved all encrypted message, n which sent to Alice.
2. If result is positive ($n > 0$), each cipher text C_{Wi} from the particular encrypted message is selected and then encrypted with B (session key s).
3. Test if $E_B (C_{Wi}) = A$. If so, the output is 'yes' and relevant document will be downloaded to Alice; else get the next cipher text C_{Wi+1} and encrypted with B (session key s). Test if $E_B (C_{Wi+1}) = A$.
4. The process is repeated until all encrypted message that sent to Alice had been scanned through.

4. Result and Discussion

SPKE encryption time refer to the total time that we used to encrypt email which contains x words in body of the email and SPKE process for y keywords in email subject field. Performance of SPKE encryption time can be tested in two ways, first on how SPKE encryption time changes reflect to the increasing of total words in subject field and second how SPKE encryption time changes reflect to the increasing of total words in email contents. For first situation, we design a test case where we send an email which contains 100 words but with increasing total keywords in subject field, from 10, 20, 30, until 80 words (see Figure 1.).

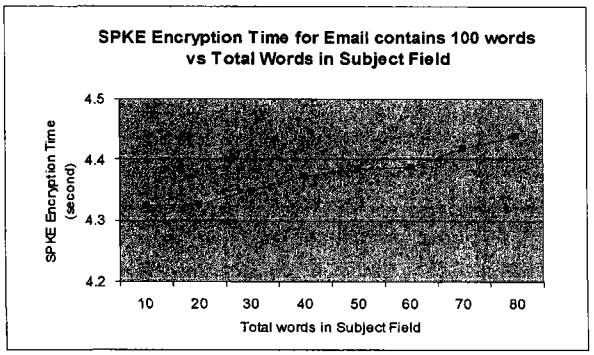


Figure 1. SPKE Encryption Time for Email That Contains 100 Words vs Total Words in Subject Field

Figure 1 shows that for 10 words in Subject filed, SPKE encryption time is about 4.33 seconds. From 10 words to 80 words in Subject field, only a small time variance, less than 0.2 second was observed. The above data proved that our implementation scheme performs an efficient SPKE encryption, where it took less than 5 seconds for email contains 100 words and time is not increasing drastically although total words in Subject field have been increased.

For second situation, we design a test case where we collect SPKE encryption time by increasing the total words in email contents from 50 words to 600 words. We run on 3 different sest of data with 50, 60 and 70 words in Subject field (see Figure 2).

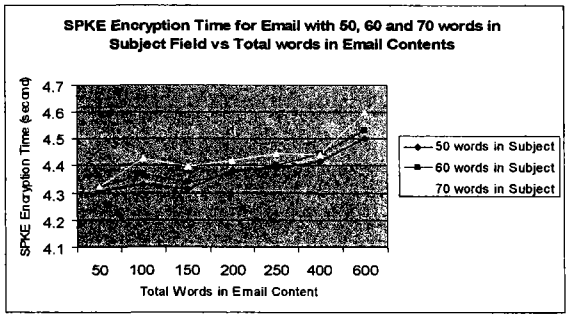


Figure 2. SPKE Encryption Time for Email with 50, 60 and 70 Words in Subject Field vs Total Words in Email Contents

From the above results in graph, we found that our proposed method which is SPKE Encryption has a good performance in email encryption time (see Figure 1 and Figure 2) and encrypted keyword searching time (see Figure 3 and 4). Refer to the results that we obtained, total time to encrypt an email which contains 600 words with 70 words in Subject field took less than 5 seconds (see Figure 2.).

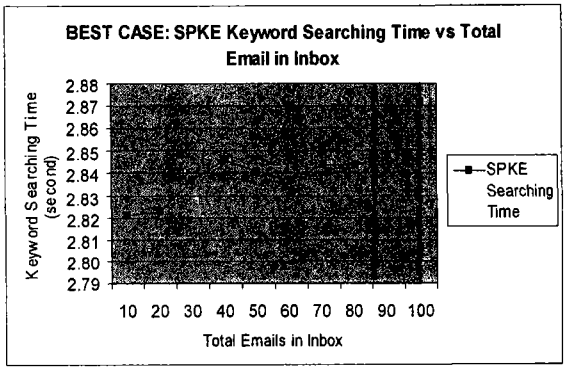


Figure 3. Best Case: SPKE Keyword Searching Time vs Total Email in Inbox

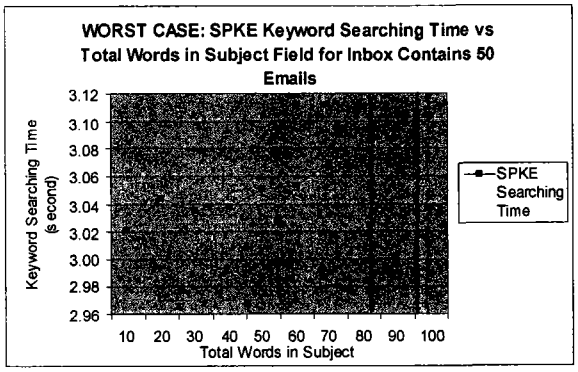


Figure 4. Worst Case: SPKE Keyword Searching Time vs Total Words in Subject Field for Inbox Contains 50 Emails

In term of keywords searching time, we design a best case where keyword that we search appears at first position in e-mail's subject field by increasing total emails in Inbox. This design will show the best performance of searching time. To obtain the worst searching time, we prepare 50 emails in inbox with the search keyword appears at the last position in subject field. Results show that searching time increase slowly when total words increase in subject field. The result is as shown on Figure 3 and Figure 4.

5. Conclusion

From the observation, we found that our SPKE scheme has a good performance in encryption time, keyword searching time and decryption time. In our scheme, we used Diffie-Hellman as our foundation primitive in trapdoor generation for particular

searching keyword. Our trapdoor generation involves few basic mathematic calculations (power and modulus) and a hash function which use to create message digest. Simple mathematic calculation ensures our SPKE scheme has a better performance time.

Although our scheme has a good performance time, but Statistical Attack can still happen. If multiple samples with same key can be collected by attacker, the encrypted keyword can be analyzed after some operations. However, Statistical Attack only able to take out some information of encrypted keyword in e-email's subject field. The contents of email is still secure because it was encrypted using AES encryption standard. Receiver is the only person who can decrypt the email contents by using her/his private key. We conclude that our searchable public key encryption provides an easy and powerful way for processing encrypted email which useful in our daily life.

References

[1] Boneh, D., Crescenzo, G. D., Ostrovsky, R., and Persiano, G. (2003). Searchable Public Key

[2] Song D., Wagner D., and Perrig, A. (2000). Practical Techniques for Searches on Encrypted Data, in Proc. Of the 2000 IEEE Symposium on Security and Privacy.

[3] Goh, E. J. (2003). Secure Indexes for Efficient Searching on Encrypted Compressed Data. Cryptology ePrint Aechive, Report 2003/216.

[4] Bloom, B. (1970). Space/Time Trade-Offs in Hash Coding With Allowable Errors. Communication of ACM, 13(7), 422-426.

[5] Shamir, A. Identity-based Cryptosystems and Signature Schemes. CRYPTO 84.

[6] Boneh, D. and Franklin, M. (2003). Identity-based Encryption from the Weil Pairing. SIAM J. of Computing, Vol. 32, No. 3, pp. 586-615.

Secure Hashing of the NEMO Mobile Router Communications

Tat Kin Tan and Azman Samsudin
School of Computer Science, Universiti Sains Malaysia
Penang, Malaysia
{tatkin,azman}@cs.usm.my

Abstract—**Mobile Router, being part of a network-in-motion (NEMO), played an integral role in ensuring the continuous connectivity of communications and services offered to the entire mobility of the network. The Mobile Router, on top of being a subset to most of the functionalities of the MIPv6 Mobile Host, offers extra NEMO specific protocols such as the modification in Binding Updates, Binding Acknowledgement, Home Agent Discovery Request and Reply messages. While it is understood that the implementation of IPSec as a standard security measurement may offer good protection on the communication path of between Mobile Node and Correspondent Node, such security dimension is insufficient to protect the signaling in particularly between the Mobile Router's Home Agent Discovery Messages within NEMO protocol.**

Keywords—**Hash Message Authentication Codes (HMAC), Message Digest Algorithm 5 (MD5), Network Mobility (NEMO), Secure Hash Algorithm (SHA1)**

I. Introduction

In the practice of Mobile IPv6 (MIPv6), the communications between a Mobile Node (MN) and its Correspondent Node (CN) often involved the participation of Home Agent (HA) and Access Router (AR) [1].

The MN that is also termed as Mobile Host, will have 2 or more addresses in particular when the mobility-play comes into the picture. It is typical that the Mobile Host being assigned a unique Home Address (HoA) and Care of Address (CoA) that changes when the MN is roaming from network to network. Per standard specification [1][2] when the MN roamed to a foreign network, the Binding Updates (BU) plays a vital role in keeping the HA in sync of the changes of CoA to a particular MN[1]. Figure 1 below signifies a typical communication scenario between Mobile Hosts.

When a Mobile Node (termed as MN in MIPv6, and as Mobile Network Node – MNN in NEMO), leaves its home link, it maintained communication sessions by firstly sending BU back to its home link addressing to its Home Agent. This is done for the fact that, once the MNN reached another foreign network, the node is given a CoA by the foreign network service. The node will have to inform HA that now it has now been given a second address and that the BU serves the purpose of binding these Home Address and Care of Address together. The CN on the other hand, has no transparency on the physical location of the MNN. The CN will still address to MN's primary address, ie: HoA. When the packets destined to MNN's HoA arrived to the home link, the Home Agent will look at the binding cache list, mapped that the HoA with newly bounded CoA, and routed the packet now destined to CoA [1].

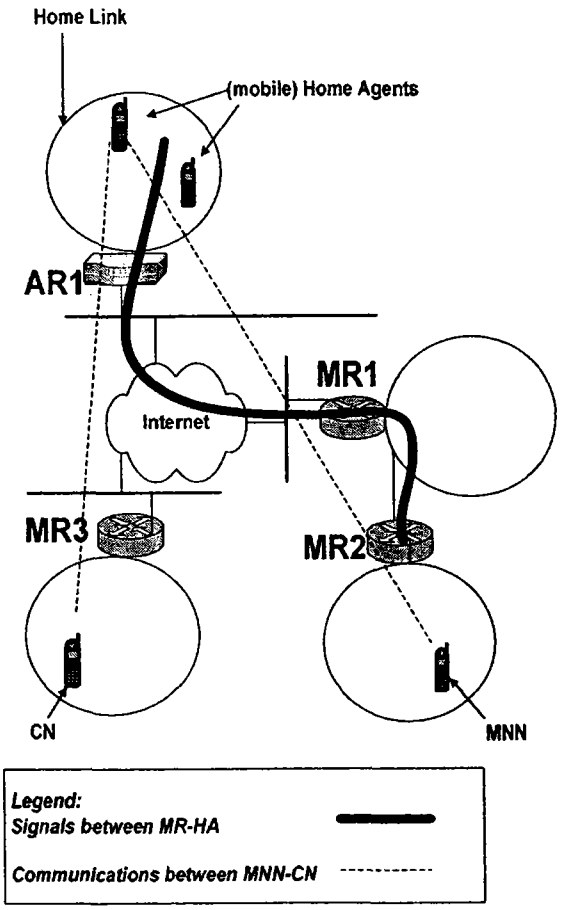


Fig. 1. Typical MR-HA and MNN-CN Interaction

Mobile Routers as an integral feature of NEMO, played an important role in ensuring the mobility of the network as these mobile hosts sit between the communication path of MN and CN. Like in MIPv6 environment, a MR which is in termed a mobile host, in order to enjoy the full freedom of roaming, has its own unique HoA so that the Home Agent will be able to participate the high mobility of the MR. This

This work was supported by Universiti Sains Malaysia.

also mean that the MR will have CoA being assigned when it roamed into foreign links [2]. It is important to recognize one of the main functionalities of MR is as a router that allocates Care of Addresses to mobile nodes that intended to attach into it.

In terms of the implementation of MR in NEMO formation, certain protocol enhancements were announced such as the changes to the Binding Updates and Acknowledgement, the transformation of Prefix Table configuration and the modification of Dynamic Home Agent Discovery [2]. It is required for instance, the introduction of Mobile Router Flag, ie: the "R" bit into the communication messages of MR for HA to easily distinguishable of whether the particular message is from a MR or a MN [2], as both entities is also a subset of Mobile Host.

While the current Basic NEMO specification [2] indicating that the authentication between MR and HA must adopt IPSec implementation [3], it is also highlighted that the incorporation of IPSec for communications between MR and HA is insufficient. This is particularly true due to the fact that the IPSec as a security deployment 1) occurred in transportation level of NEMO; and 2) the IPSec protection offered in tunnel mode for NEMO specific traffics tunneling between CN and MN. An interesting security threads analysis has been done in [4].

II. PROBLEM STATEMENT

It is also outlined that the IPSec can protect the NEMO signaling such as the Binding Updates and Binding Ackowledgement via the use of Authentication Header (AH) and Encapsulation Security Protocol (ESP). As outlined in [4], these are done at transportation communications between MR and HA. Here in this paper, we present a problem in the scenarios of how malicious MR that sits between communications can modify communication messages with the loophole in IPSec and AH.

There are tremendous security concerns being highlighted when IPSec is being deployed, and in particular when MR is in search for Home Agent, that is the Dynamic Home Agent Discovery process [1] (refer to Figure 2 for the protocol layout). Note that Dynamic Home Agent Discovery process proposed in MIPv6 proposed stacks has been modified in NEMO specification to provide router mobility support. As shown in Figure 2 below, the "R" bit has been proposed so that the Dynamic Home Agent Discovery messages can be recognized differently as to whether this message is for a typical mobile node, or a mobile router.

Typically, in a perfectly possible real-life example, a MN, or rather in this case a MR, that roamed to a foreign link and is in search for Home Agent, will send out signals to search for mobility support. In the NEMO environment, the MR will engage in the Dynamic Home Agent Discovery (DHAD) process that, the MR will send a DHAD-Request by setting a Mobile Router Flag ("R" bit set to 1). This is to indicate that the MR wanted to discover Home Agents that support Mobile Routers [2]. This scenario is possible in reboot or setup stage of a typical mobile host.

Dynamic Home Agent Discovery Messages

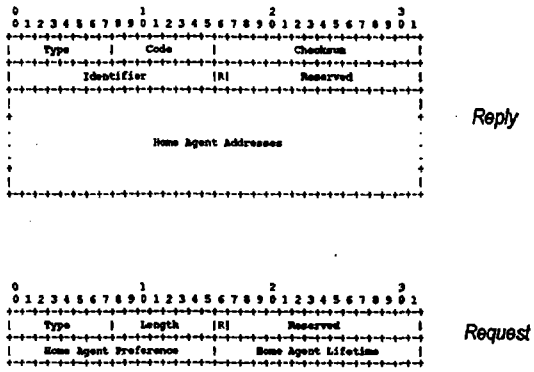


Fig. 2. Dynamic Home Agent Discovery option defined in NEMO-MIPv6

If a Home Agent receives this DHAD-Request message that is with the Mobile Router support flag ("R" bit) set to enable (value = 1), the HA will have to reply by sending a DHAD-Reply message [1]. Depending on environmental setup in the Home Link, this DHAD-Reply message that is sent by Home Agent could contain a list of Home Agent Addresses that support Mobile Routers. If the Home Agent does not support Mobile Router, the Mobile Router support flag (the "R" bit) will be set to 0. This message is then sent back to the MR that is in quest of router mobility support.

Whereas, in NEMO Threats Analysis draft [4], it is believed that there are significant flaws in DHAD message exchange between MR and HA. We see the issue as in a bigger problem and can be applied to a more general message exchange. In simple we think this is a more serious threat that not only applicable in DHAD process but also applicable to more message exchange processes in NEMO. We will use DHAD as the example for demonstration.

A. Attack of the DHAD-Request path

As shown in Figure 3, if a malicious attacker is this scenario assuming as a MR1, listened to the communications between MR2 and HA (refer to Figure 1 as the possible setup scenario) intercepted the DHAD-Request message sent by a legitimate MR2, and modified the Mobile Router support flag "R" bit from the original request of 1 to 0, the malicious MR1 then forward the modified message to HA. When the DHAD-Request message is being received by a HA, the HA will look at the message, and see the "R" bit value as disabled (value = 0) and not know that this message is actually asked for a DHAD-Reply!

Since the HA does not see the R-bit flag being enabled, the HA will not response to the request according to the correct behavior. In this example, the HA will interpret that the message is sent from a mobile host, probably a mobile node and not from a mobile router. This is understood that it is the MR whom will be setting the "R" bit. The HA will respond to MR2 in a way that in HA's reply, HA will not enable the "R" bit to indicate the acknowledgement of mobile router support.

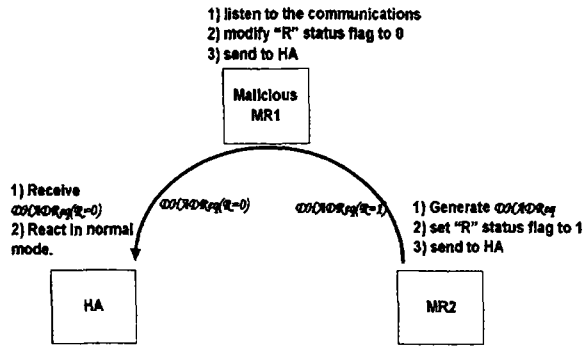


Fig. 3. Attacks on DHAD Request

On the other hand, the legitimate MR2 that was requesting the MR-supports, will be waiting for replies and when it sees the DHAD reply message responded by HA did not have the "R" bit enabled, will eventually conclude that there is no Home Agent that provides Mobile Router supports.

B. *Attack of the DHAD-Reply path*

Yet another possible scenario of attack by malicious MR1 can be imagined from another way around. Imagine that MR2 did not attack at the DHAD-Request path but strike at the DHAD-Reply. The scenario can be demonstrated as in Figure 4, that MR2 sent DHAD-Request message, and being received by a Home Agent. The HA realized that the Mobile Router support flag "R" bit is set in the message, and thus replied back to MR2 with DHAD-Reply message that, depend on situation, could contain a list of Home Agents that support Mobile Routers. In this situation, let us supposed that the HA maintained a list of Home Agents Addresses that enabled Mobile Router support. The HA will then reply a DHAD-Reply message and then enable the Mobile Router support flag of "R" bit and set to value 1.

Now, imagine the malicious MR1 sits between the communication path of the DHAD-Reply, the malicious MR1 can attack by intercepting the DHAD-Reply message, and set the Mobile Router support flag of "R" bit into 0, and resent to the legitimate MR2 as shown in Figure 4.

Refer to Figure 1 that such setup is possible and is practical because MR2 is actually a nested network within MR1. The impact of doing so is obvious that, when MR2 received the DHAD-Reply message, the MR2 will decipher in a way that, a DHAD-Reply by HA has indicated that the "R" bit has been disabled (with value = 0) and translated as though that there is no Home Agent that provide support to Mobile Router!

III. SOLUTION

To ensure the integrity of packet header (or even to a greater extent, to assumed that the entire packet itself) is not corrupted or being compromised in the middle of communication in this NEMO DHAD scenario in particular, we recommend the NEMO protocol may be revised.

The current NEMO specification has overly rely on IPSec as the primary security standards and made assumption that the usage of IPSec with Authentication Header [1] would provide authentication for the packets. However and despite the strength of IPSec with AH, there are still security threats as discussed in [4].

As such, to counter measure the problem outlined in the attack of DHAD message as an example, we propose to introduce to a new implementation of hashing algorithm, targeting into hashing the DHAD protocol in NEMO situation. It is also worth mention that, the idea of hashing can be extended beyond the packet header, which is to hash the entire packet. It is not limiting to the possibility of applying the same hashing idea into other NEMO communication area.

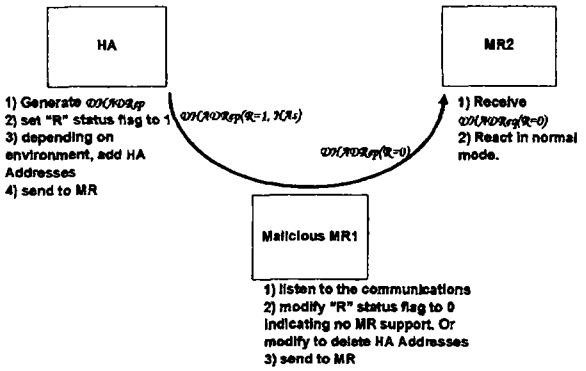


Fig. 4. Attacks on DHAD Reply

By establishing hashing mechanism, the entire communicated message can be hashed to produce a digest. The usage of this hashed digest is that, the digest will then be appended into the DHAD protocol to provide new dimension of authentication purposes. If the message has been altered, or corrupted in between a communication, for instance in DHAD example, the receiver will always be able to determine that the communication has been compromised via the re-calculation of hash algorithm. Note that the hash calculation is different from what has been provided as packet checksum in Figure 2.

A. *Introducing Hashing Mechanisms*

It will make no sense to have the data exchanged between MR and HA ciphered because both parties needed to know the source and destination address. Based on this fundamental conception, we propose to use Hashing Message Authenticate Codes (HMAC) [5] as the alleviation to the problem as described above.

The motivation of voting for HMAC is simple. This is due to the fact that HMAC is a simple hashing protocol that takes any length of data, and a shared secret key that only the parties-in-communication know, calculate and capable of producing a hash value, and at the same time the HMAC do not encrypt the message.

The idea of including hash codes at the end of the message as a method to authenticate the message integrity require minimum changes on the current underlining NEMO protocol. Our solution only required additional trailing bytes to be added in the message and the size of extra bytes are depending on the selection of hashing algorithm.

B. Hashing Implementation on MR

It is not unbearable to alter the existing NEMO basic etiquette for our proposal. Based on the basic rules in NEMO of engagement, when DHADs are in progress both MR and HA have to set the "R" bit to 1, as the endorsement of Mobile Router [2].

As shown in Figure 5, base on our propose circumstances, when MR2 generated the DHAD Request, assuming the implementation of HMAC hashing algorithm, the MR will need to append a 16-byte hashed output into the packet. 16 bytes is needed because of MD5-HMAC. The hash output can be calculated by taking the entire and original DHAD Request, termed as *DHADReq* for simplicity in our example, apply HMAC hashing algorithm together with a Secret Key – *scK* and the hash output, *h1*, can be constructed.

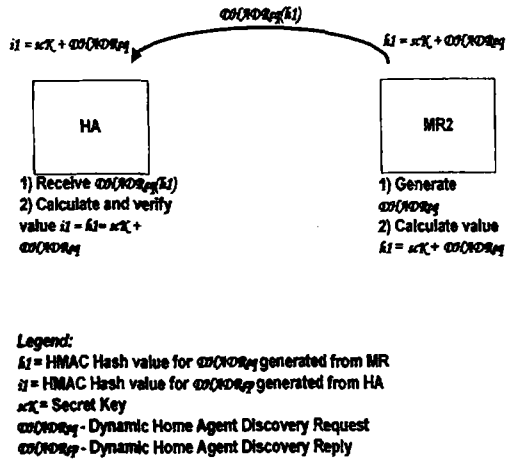


Fig. 5. Proposed New Dynamic Home Agent Discovery Request

The MR2 will then send *DHADReq(h1)* message in clear text format out for communications. We recommend the *h1* to be embedded as part of the DHADReq message. This can easily be done by introducing an extra field, or to extend the existing checksum field. HA will detect the DHAD signal as though in normal communications as in Figure 1, this message will then be sent to its home link and recognized as a DHAD Request.

HA on the other hand upon receiving the signal, must be able to authenticate the integrity of the DHAD Request message.

The process will be done by enhancing the current NEMO protocol that, when HA detected the DHAD Request, HA will extract out the *DHADReq* message, and then apply HMAC hashing calculation together with the Secret Key, *scK* that is only known between HA and MR2. The resulting hash codes by HA, for clarity in this paper, will be named as *i1*. HA will then use this *i1* to compare and verify with the hash code being attached with the DHAD Request message which is *h1*. The resulting comparison must be that these 2 sets of hash codes are exactly equivalent.

In any case that these 2 sets of hash codes *h1* and *i1* are dissimilar, the HA will know that the DHAD Request is either corrupted or being compromised in the middle of the transmission. Depending on implementation, user may configure the HA to either alarm MR on the problem and take necessary reaction, or decided to drop the request silently.

C. Hashing Implementation on HA

As for the HA, when HA has validated, verified and is convinced that the authenticity of the DHAD Request, per the Basic NEMO specification, the HA realized that an "R" bit was set and hence interpret that the request is asking for Mobile Router supports. HA will need to react accordingly by creating the DHAD Reply message [1,2]. Depending on the situation and environmental setup, the DHAD Reply message may or may not contain multiple Home Agent Addresses. Nevertheless, the HA will enable the "R" status flag within the reply message with the intention so that when MR2 receives the reply, MR2 will be able to understand that the reply is indeed indicating for Mobile Router support.

Using the HMAC hashing algorithm again, HA will compute a hash output, in this case we named as *i2*, by applying the Secret Key, *scK*, that is already shared and only known amongst MR2 and HA, together with the DHAD Reply message, *DHADRep*. HA will then send the DHADRep*(i2)* message back to MR2 in clear text format as can be seen in Figure 6.

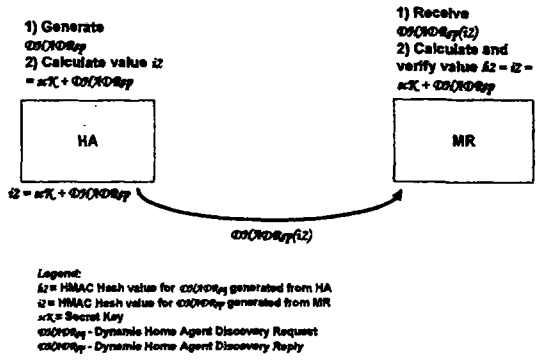


Fig. 6. Proposed New Dynamic Home Agent Discovery Reply

Upon receiving the reply, MR2 will extract the *DHADRep* message, and compute another hash codes, we named as *h2*, by applying HMAC hashing algorithm with the Secret Key, *scK* and *DHADRep* message that it received from HA. MR will then verify that the 2 sets of hash codes *i2* and *h2* are similar and concluded the authenticity of *DHADRep* message.

In any case that these 2 sets of hash codes are dissimilar, the MR will then know that the DHAD Reply message has either be modified or corrupted in between the transmission. Depending on implementation, user may configure the MR to alarm HA on the problem and necessary reaction can be taken, or choose to drop the packet silently.

D. Hashing Implementation on HA

We proposed to choose MD5-HMAC as the hashing algorithm to resolving the DHAD problem. We recommended MD5-HMAC in the preference of the superior and critical performance of MD5 [5], compared to other algorithms that offered similar functionalities such as SHA1.

Although SHA1-HMAC appears to be a stronger and more robust algorithm in terms of cryptographic terminology, MD5-HMAC offers 16 bytes of hash output data as the result of MD5-HMAC, while SHA1-HMAC uses 20 bytes of space in the packet.

It is however, up to the implementer to decide which deployment to be implemented with respect to performance of MD5-HMAC versus the robustness of SHA1-HMAC.

E. Secret Key Exchange Method

It is assumed that both MR and HA had already obtained a shared Secret Key prior to engaging to the DHAD process. This can be done by enforcing the implementation of Diffie-Hellman Key Exchange.

Basically in the IPv6 world, the IPv6 subnet prefix allocation can be managed by ISP or particular organization [8]. As such, it is not difficult for related enforcement party to setup authentication server, key server or some kind of public-key infrastructure. In this case, a Diffie-Hellman Key Exchange algorithm can be supported between nodes and organization's key server. The technical implementation of Diffie-Hellman Key Exchange can be obtained at [7].

The processes and detailed procedures of how MR and HA built a shared Secret Key via Diffie-Hellman algorithm is outside the scope of this paper.

IV. BENEFITS – CRYPTO PROCESSING

Many researchers have been looking for alternative solutions to replace IPSec as the main security feature implementations for MIPv6, or in this case, NEMO. In NEMO Threats draft [4] it had clearly indicated that although IPSec offer authentication and with Authentication Header Mechanisms [6], it could not protect the attacks between MR and HA and example given in DHAD process. As such, the idea of providing Hashing Algorithm together with Diffie-Hellman implementation can be extended to not only the DHAD process but also addressable to all other communication messages within NEMO.

The current drafts or proposals published especially in NEMO; do not promote other security implementations because it is already assumed that IPSec is good enough. Hence, our proposal not only provides a different dimension but also supply a substitution method for IPSec.

V. DIFFICULTIES – STANDARD WEAKNESSES

Common security threads that applicable in HMAC hashing algorithm [5,6] as well as NEMO (or rather internet) are applicable in this solutions. Imagine that, even with the implementation of HMAC, a malicious MR can still intercept the packets and attempt to modify or corrupt the packet content. Such act will result the 2 hash codes being dissimilar. Either MR or HA will conclude that the packet has been corrupted or being attacked, resulting packet being dropped. Such act can still be considered as another form of Denial of Service (DoS) attack because the mobile hosts may no longer be able to communicate properly. This problem however, is still widely regarded as the common implementation problems for most message exchange protocol in the internet world.

To counter measure the above mentioned difficulty such as DoS attack especially causing packets to be dropped silently, we propose a simple solution for ratification. This can be done, in the event that either mobile host (HA or MR) receive similar requests/replies over a repeatable number of time scale, either mobile host will be able to conclude that an attack has occurred and instead of droping the packet silently, certain actions can be taken. For instance reactions can be taken by either informing enforcement parties such as Internet Service Provider (ISP), or notify each other that an attack indeed happened. Sessions can then be terminated or MR can choose to leave the problematic domain/network.

Another potential difficulty for this proposal will be that it involved the chances of rewriting of NEMO protocol as a whole. Extra precautious must be taken since our proposed solution may influence new ideas or developments on NEMO in particular the Binding Updates authentications in both NEMO and MIPv6 protocols.

VI. CONCLUSION

As a conclusion, the existing usage of IPSec together with Authentication Header (AH) onto Network Mobility for security measurement undoubtedly could establish extensive security stronghold against various attacks.

However it has the weaknesses as well. The typical example will be as highlighted in this paper, the loophole in Dynamic Host Agent Discovery Messages between Mobile Routers and Home Agents. This is particularly true when the IPSec is not protecting on the transportation layer of such protocol.

As such, in our solution we have proposed to use Hash Message Authentication Codes as an authentication protocol. In our example, we opted to use MD5-HMAC as our solution.

The solution is considered simple to be implemented because of only needed to enforce extra message which is the hashed message, to be carried together with the Dynamic Host Agent Discovery Messages.

REFERENCES

[1] D. Johnson, C. Perkins, J. Arkko, "Mobility Support in IPv6 , draft-ietf-mobileip-ipv6-24, IETF, June 2003

[2] V. Devarapalli, R. Wakikawa, A. Petrescu. and P. Thubert, "Nemo Basic Support Protocol" (work in progress). Internet Draft, IETF. draft-ietf-nemo-basic-support-02.txt. December

[3] J. Arkko, V. Devarapalli and F. Dupont. "Using IPsec to Protect Mobile IPv6 Signaling between Mobile Nodes and Home Agents". Internet Draft, IETF. draft-ietf-mobileip-mipv6-ha-ipsec-06.txt (work in progress). June 2003

[4] A. Petrescu, A. Olivereau, C. Janneteau. and H. Y. Lach, "Threats for Basic Network Mobility Support (NEMO threats)" Internet Draft, IETF. Draft-petrescy-nemo-threats-01.txt. January 2004

[5] H. Krawczyk, M. Bellare and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997

[6] E. Rescorla, "A Survey of Authentication Mechanisms". Internet Draft, IETF. Draft-iab-auth-mech-03.txt. March 2004

[7] W. Diffie, M. Hellman "Multiuser Cryptography Techniques". IEEE Transactions on Information Theory November 1976

[8] R. Hinden, S. Deering, "IP Version 6 Addressing Architecture". RFC2373, July 1998

PKI and Secret Key Cryptography Implementation for NEMO Security

Tat Kin Tan and Azman Samsudin
School of Computer Science, Universiti Sains Malaysia,
11800, Penang, Malaysia.
Email:{tatkin, azman}@cs.usm.my

Abstract

Conventionally, the security design in MIPv6 and NEMO had been relying on IPSec that offer Authentication Header and Security Associations profiling as the main design. The focuses were particularly on how to fence off the unauthenticated or malicious bindings of Binding Updates that open the doors for various attacks. There also exist some circulations of internet drafts and specifications in today's industry for PKI implementation within the protocol of MIPv6 and NEMO, with or without the existence of IPSec because there were numerous security design inefficiencies been found to date. In our paper, we will firstly analyst and concentrated on the communications between Mobile Node, Home Agent and Correspondent Node MN-HA-CN on the path of Return Routability (RR) and Route Optimization (RO), from both MIPv6 and NEMO point of view. We will then provide our own solutions of implementing the secure communication of PKI together with the distribution of Secret Key methodology into the NEMO environment.

1. Introduction

The Internet Protocol (IP) has evolved into the recent IPv6, as well as the support of the Mobility of IPv6 (MIPv6). The evolution of IP had enabled devices to enjoy mobility without loosing communications. Access Routers (AR) were needed to provide routing capability for packets during IPv4 generation. As in the MIPv6 protocols, the AR plays a part of the access points for devices such as Mobile Network Node (MNN), Home Agent (HA) and Correspondent Node (CN). The AR routes packets between the MNN and CN. The Home Agent (HA), support the MIPv6 protocol as the agent that keep tracks on the where about of the MNN when MNN is roaming out from its home.

One key technology breakthrough is the introduction of "Networks that mobile", hence Network Mobility (NEMO). The term Mobile Routers (MR), had since been seen gaining more popularity amongst its peer (the AR in MIPv6), as the MR supplements the traditional bulky access point. At present, the MIPv6 supports protocol recommended the use of IPSec and the associated Encapsulation Security Payload (ESP), Security Associations (SA) and Authentication Header (AH) as the anchor features that achieve reasonable security design and fence off majority security threats [1]. The NEMO protocol, being a derivative from MIPv6 support, too, recommended and inherited the similar security design of using IPSec [2]. Because of the inheritance, common security design problems that the IPSec is facing, would also be seen in NEMO. As such, effective and efficient authentication methods in NEMO and the protection of communications has become one of the hot topics, and so as the motivation of this paper.

To have used IPSec as the core security design into both MIPv6 and NEMO, numerous questions had surfaced by either having doubts on the efficiency of implementing IPSec, or problems being exposed after employing IPSec, as well as the ease of maintaining and exchanging keys for cryptographic processing [7][8].

We intended to provide some analysis on the current security implementation of MIPv6 and NEMO, and to also provide alternative solutions to the problems by introducing Public Key Infrastructure (PKI) together with Secret Key encryption system. We will also look into some existing works on conference papers that provide some excellent design of making the Binding Updates (BU) and Return Optimization (RO) operations more robust against redirection attacks [1][7][8]. One may argue the introduction of trusted server, also knows as Certificate Authority (CA), which

is a key component of the success of PKI could be complicating to the existing protocols. However, we think otherwise since the trusted servers that distribute and manage Public Keys for a particular node can in fact be the Internet Service Provider (ISP) itself.

2. Overview of MIPv6 and NEMO

For MIPv6: In MIPv6 specification [1], a method knows as Return Routability (RR) is used in associate with Binding Updates (BU). As illustrated in the MIPv6 specification, the BUs carry important information such as the Care of Address (CoA) given to a MNN when the MNN is not at home, or to provide prefix subnet delegation onto Mobile Routers of NEMO. The MNN sent BU to HA to register the current addressable CoA. HA will send Binding Acknowledgement to MNN informing the status of the binding. The RR on the other hand is a process used to authorize, authenticate, exchange and establish cryptographic token or keys being used between MNN and CN [1].

Figure 1 depicts the topology of RR protocol. As defined in MIPv6 specification [1] and narrated in [7][8], when MNN roamed out of its own home network, MNN will need to engage with RR process. Home Test Init (HoTI) and CareOf Test Init (CoTI) are the 2 messages being sent to HA and CN respectively almost simultaneously. When HA receives the HoTI message, it then routes the message to CN, whereas the CoTI is sent to CN directly. Within the HoTI and CoTI, both messages contain the MNN's Home Address (HoA), CareOf Address (CoA), Random Number for HoTI (R_{HoTI}) and Random Number for CoTI (R_{CoTI}) respectively and addressed to CN. Both R_{HoTI} and R_{CoTI} are being returned by Home Test (HoT) and CareOf Test (CoT) to assure that MNN gets back the random values that it sent earlier on.

Upon receiving the HoTI request from MNN, the CN will use the source address within the HoTI, which is HoA, together with CN's secret key and index of nonce ($Nonce_x$) to calculate a hash value C_{HoTI} and C_{CoTI} (also termed as cookie) [1]. The key hash values are calculated per specified in [10]. The index of Nonce, $_x$ and $_y$, will be used to speed up index searching at the later stage of the RR protocol. We can see from the RR protocol, some kind of authentication methods are introduced that, i.e.: random numbers generated by MNN, are being passed via 2 different channels. These random values are then later being authenticated of the existing of the channels and the most importantly the routes, by MNN receiving back the CN's reponses.

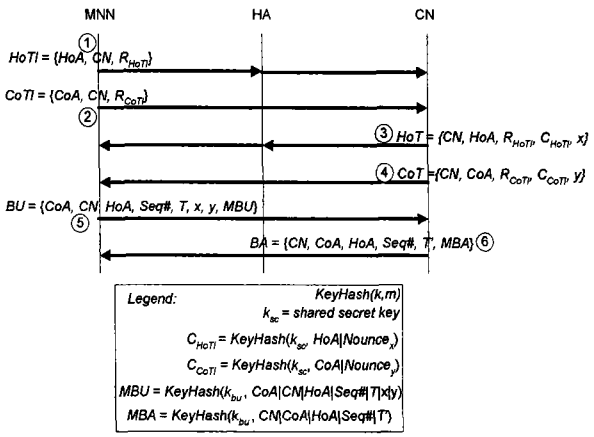


Figure 1. The Return Routability Protocol.

Leading to the completion of the RR protocol, MNN is now armed with the necessary information to construct a Binding Update. This Binding Update will serve to update CN on the new CoA that the MNN has been assigned. The BU will also consist of a Sequence number to protect from replay attacks, Time Stamp that indicates the life span of the BU, index $_x$ and $_y$, and a Message Authentication Code (MAC) that determined to protect the session of the k_{BU} of the BU. The k_{BU} on the other hand is the resulted production of a hashing on C_{HoTI} and C_{CoTI}. When CN received the BU, CN will need to authenticate the BU by validating the data such as $_x$ and $_y$ that informs CN which nonce value in index to be used, within the BU message and recomputed the C_{HoTI} and C_{CoTI}. In return of the acknowledging the positive result, CN will return a Binding Acknowledgement (BA) to MNN. The CN will form a binding cache of the new CoA to associate with MNN, so that from this point onwards the MNN is addressable at the new CoA.

For NEMO: The NEMO Basic Support Protocol has identified numerous changes to support the fundamental building blocks for network mobility. Network Mobility (NEMO) by definition, is a segment of network or subnet that has mobility and can move the point of physical attachment, to also achieve routing capability to the nodes that associate with the movement. The motivation for NEMO is also to provide continuous communications as well as session connectivity for the mobile nodes that attached with the Mobile Router (MR), even when the MR has changed to its physical attachment point [2]. The MR is also essentially a MNN that has routing capability between its point of attachment, and to also provide subnet prefixes for connectivity. Since the MR is also essentially a mobile host with MIPv6 and routing

capability, the MR in NEMO will implement Binding Updates protocol with a bi-directional IP tunneling between the MR and its HA. RR protocol is thus applicable in NEMO protocol between MR-HA, when the MR has changed its point of attachment and being assigned a new CoA. This RR protocol served the similar functionality as to update HA of the location of the MR, and that HA will be able to route the packets destines to prefix that belongs to MR accurately from other nodes. The messages sent by MR will include a "R" bit to flag to HA that the MR is functioning as a Mobile Router [2].

For our discussions we will assume MR will function as a router rather than Mobile Host. Nested Mobility [12] and other NEMO home network models [11] are not the scope of this paper. As shown in Figure 2, within the BU sent by MR to HA to updates the CoA newly acquired by MR, a flag indicated as "R" will be set to inform HA that the MR would like to perform as a router. Once BU is succeeded, the HA will have the aggregation of network prefixes advertised by MR.

When CN wants to communicate to a node in the mobile network, the packets are routed to HA, of which this HA will have the binding cache of the particular MR prefix. When HA received packets that are destined to a certain mobile prefix, the HA will know which MR to forward to. HA will then encapsulate the packets and tunnel to the MR which is addressable at the updated CoA. MR on the other hand, will need to de-capsulate the packet that it received from HA and route the packet to the node in conventional routing protocol. From the MNN's perspective, as shown in Figure 3, the tunnel in between MR-HA is transparent to the MNN, which also means MNN's MIPv6 protocol has not been changing. The only changes is the MR that will need to build up multiple IP Encapsulation tunnelings (IP-to-IP encapsulation) for multiple communication path [2][3].

3. Problem Statement

The solutions proposed in [7][8] such as the Extended Certificate-Based Binding Update protocol (ECBU) are focus on MIPv6 solutions, in suggesting a Certificate-based BU implementations to cover the communication paths between CN-MN, CN-HA and HA-MN scenario. It also suggested some authentication mechanisms with the likes of Key Exchange methods within RR. The focus of these papers are on exposing the security threads of MIPv6 between MN-CN and made assumption that HA/MR are dumb and bulky device that sits in a server room with good processing power. Without considering the improved technology that made Network Mobility, these assumptions needed to be enhanced and redesigned of security system as a whole is imminent.

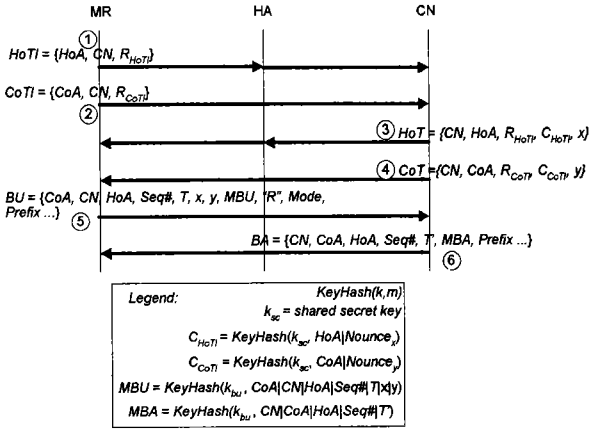


Figure 2. RR and Binding Updates of NEMO

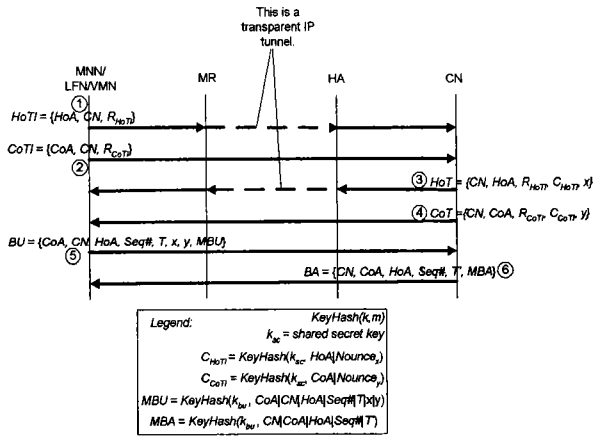


Figure 3. IP Tunneling between MR-HA

On the other hand in NEMO Threats draft [4] it had clearly indicating that although IPSec offer authentication and with various Authentication Header Mechanisms [6], it could not protect the attacks between MR and HA. We would also like to point out that, while protecting the data integrity can be done by introducing hashing mechanism such as the method of HMAC [5] introduced in MIPv6 [1], HMAC can only protect the data from being modified but not offering the sheltering to secrecy and sensitivity of the data.

The introduction of IPSec, which means the added overheads of Security Association (SA) for IPSec

users, would consume processing resources on devices such as PDA. We see the IPSec implementation [3] would introduce processing overheads due to the complicated protocol levels and especially when nested networks in NEMO that would introduce multiple levels of IP-in-IP encapsulation/de-capsulation that could even stressing the performance.

We recognized that there is a danger that the BU data is being compromised with the absent of clear probably encryption standards being defined in [1][2]. When the BU is communicated in clear text and below is one of the possible examples and the similar idea had also been pointed out in [7][8]. Assuming a situation of NEMO that 2 MRs are in active communications with the same CN as shown in Figure 4, an intruder (for simplicity, here we termed as i-MR) i-MR can listen and monitor the communications of MRs-CN as well as the MRs-HAs-CN path. Since the RR and BU messages can be sent in clear text format, i-MR can easily extract the information embedded within RR and BU. As such, the cookies, random numbers and index are exposed.

When the i-MR obtained C_{HoTI}, the i-MR can initiate a fake CoTI on behalf of MR1 to CN with source as MR2's CoA and the CN will reply with CoT message consisting C_{CoTI}. The i-MR can then obtain the C_{CoTI}. The sessions key k_{BU} computations that required both C_{HoTI} and C_{CoTI} can be easily calculated. Once the session key is exposed, the resulted hashing message of MBU and MBA can easily be altered. i-MR can now manipulate BU initiation on behalf of the MR1 and the impact is catastrophic. Important data such as the routing prefixes can be altered and resulting packets being routed to wrong MR and impacting entire routing network, or resetting or disabling the "R" bit flag and thus denying mobile host to function as a mobile router etc.

Another simple attack would be on the nonce index message on RR test. Since the BUs would likely to be in clear text, a malicious node can simply monitor and modify some bits in the packet which stored the information of nonce indexing, of all messages exchanged. This attack is possible since the nonce index is not part of the input parameter in hash value at the very beginning. Refer to Figure 1, step 3 and 4 where the messages are sent in clear text and the values x and y are not yet a part of the hash input. This attack will be a form of Denial of Service (DoS) and will lead to CN discarding the packet when the CN search its own index database using wrong value and could not produce a correct hash for verification. We must also highlight that the RR is not protected. In MIPv6 it is assumed that the channel between MNN and HA are protected by IPSec. We will introduce a secret key encryption which we believe provide better performance. Based on the assumptions above, we intended to provide a more practical way of implementing PKI with the combination of Secret Key Encryption.

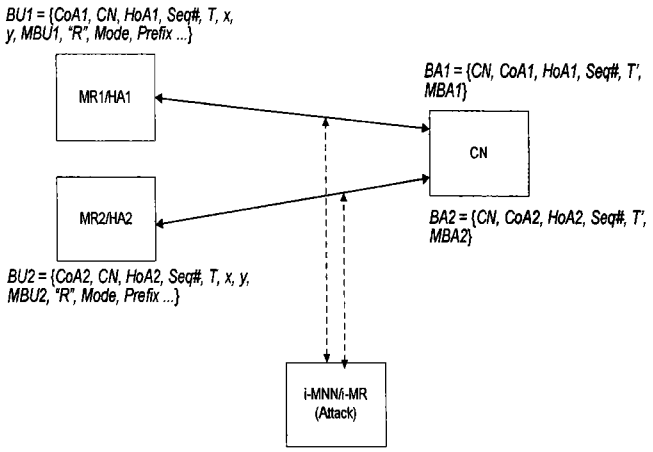


Figure 4. Scenario Where Malicious MR Attack

4. Solution

We propose to implement Public Key Infrastructure (PKI) into NEMO system supports. While one may argue that the intensity of cryptographic public key calculation may be deferring and dampen the processing power and resource of the typical portable devices such as PDA, we think otherwise. There already exist in the marketplace some commercial products processors (such as Intel IXP425) that can offload cryptographic processing to other segment of the co-processor to enhance system performance.

Our thoughts on PKI based on the fact that the ease of implementation, and in considering of using the Internet Service Provider (ISP) as the trusted Certificate Authority (CA). No Mobile Host, be it HA/MNN/MR/CN can exist without registered to ISP for subscription of internet services; hence the problem of getting a trustable CA can easily be ironed out. Our recommendation of security design will consist of two phases. The first phase will be the PKI keys exchange to establish communications between MR and CN. The second phase will be using a common secret key for encryptions and decryptions.

We recommend that, when the Mobile Hosts (in this case the MRs) boot up, the MHs should have subscribed to its own ISP and when this action happened, the ISP acts as CA, certifying and established trust relationship. The CA will play a role

as Public Key Authority (PKA) that stores, exchanges and distributes Public-Private Keys of individual MH. MR (as the Mobile Host in our example) will generate a pairs of Public Key (K_{Pb}) and Private Key (K_{Pv}), and update both keys to the CA. We also recommend a timestamp of the entry being provided to ensure new pairs of keys to be updated once the lifetime of existing key pairs has expired. How to manage the keys between the ISP-PKA-MH is out of the scope of this paper.

As shown in Figure 5, supposed that the MR in the active mode and already established the trust relationship with its CA (ISP) and initiated Public-Key-Request to CA to obtain CN's public key. The request contains the transactions of Request and Time1. This request is sent in plain text and when the PKA responded, the message returned by PKA to MR will be encrypted by the PKA's private key. The PKA will insert CN's public key, K_{PbCN}, together with the received Request and Time1 messages, and encrypt this return message with PKA's private key, and send this message to MR as shown in step 2. To decrypt the message, the MR will need to apply with CA's public key, which already exists in the MR's cache. MR can now be sure that the message is originated from PKA since only applying PKA's public key can the message be deciphered.

In Step 3 the MR will then send a message to CN that contains the identification of MR and a random number, RN1, in the attempt to establish communication. When CN receive the message sent by MR, CN decrypt the message by applying its Private Key, K_{PvCN}. CN then checks its database and realized that CN does not already have the public key of MR. Similar to how MR obtains CN's public key, as in step 4, CN initiates the process to obtain MR's public key by sending a Public-Key-Request message to PKA, associated with a time stamp, Time2. The PKA will reply to the request the same way it responded to MR's request, by forming a message that containing MR's public key, the Request and the Time2 and encrypts the message with PKA's private key, as depicts in step 5. CN generates a Cookie, CK2, together with the RN1 random number received, and formulate in a message to be replied to MR. This reply message will then be encrypted with the newly obtained Public Key of MR, K_{PbMR} as shown in step 6.

When MR received the encrypted message from CN, MR decodes the message by applying MR's own Private Key. MR will verify that the Random Number RN1 being sent earlier has indeed being returned. MR will also acquire the Cookie CK2 being sent by CN and store within MR's cache. The first phase of PKI exchange has now been completed and both MR and CN have verified the existence of individual. As depicted in Figure 5, the MR will formulate the second phase of communication by preparing a message that included with CK2 that the MR newly attained from CN, and generate a Secret Key, K_{Sc}. This K_{Sc} will be used for secret key encryption and decryption at later stage. As in step 7, MR will encrypt this message with its own Private Key on the message to ensure that this message is indeed originated from MR and can be verified by applying decryption key of MR's public key. MR then further encrypt this already encrypted message with CN's public key, to ensure that only CN can read it when CN apply CN's own private key for decryption. The formula is adopted and modified from [9] and as shown as below:

$$E_{KPb(CN)}\ (E_{KPv(MR)}[CK2,\ K_{Sc}])$$

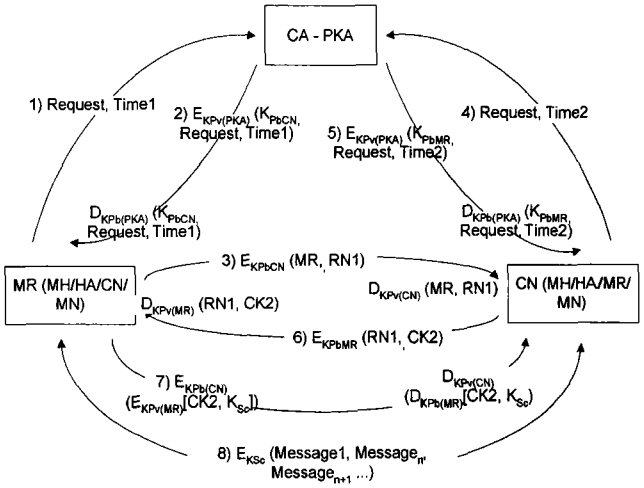


Figure 5. Keys exchange and communications between MR-CA-CN

When CN received the message, CN first decrypts the outer layer of the message with CN's private key. Only CN can decrypt this message successfully since only CN owns the private key. CN then further decrypts the inner layer of the message by using MR's public key to ensure that this inner layer is indeed originated from MR. Both the authenticity (CN verifying the CK2 that it sent earlier on) and data integrity of message is then verified. Both MR and CN have a shared secret key that only known by both of them. The decryption formula is adopted and modified from [9] and as shown as below:

$$D_{KPv(CN)}\ (D_{KPb(MR)}[CK2,\ K_{Sc})$$

Once both MR and CN agreed upon a shared secret key, both will switch to adopt secret key encryption/decryption methodology.

5. Advantage and Disadvantage

The proposed design provides an alternative solution to the IPSec as the main security architectures as suggested in [1] and [2] because our solution eliminates the hassle of setting up and agreeing SA, AH and ESP needed in IPSec [3]. Beside that, our solution does not introduce IP header protocol overhead since no IP-to-IP tunneling is required. The solution shows cheaper in processing as compared to Secure RRH in NEMO draft [13] because it uses a waterfall model of encryptions when in nested mobility. Furthermore, it should show significant benefits when implemented in the nested mobile router scenarios simply because the flexibility of secret key encryption is applicable at any level of the communications of BU or even RR. Our proposed solution even make the BUs as a cipher text therefore, there is no way for the integrity of the packets being compromised, unless the secret keys already known. This is do-able since the encryptions only happened on payload of the packets and much of the packets header structure will remain unchanged.

However, the initial setup that involved PKI keys exchange will need six processing steps and this could potentially an issue when considering the mobility of Mobile Host. Nevertheless, once MR and CN authenticated each other, the six steps required by PKI will not be needed, until either the next system reboot, or the fresh communications of MR to new Mobile Host, the PKI will not be needed again. Secret Key encryption will take over as subsequent exchange of communications. Also the implementations of PKI and Secret Key mechanism may introduce some degree of structural changes on the MIPv6 and NEMO and even the possibility of rewriting the protocols.

6. Conclusion

Our proposed security designs making use of two different encryption methodologies, namely the PKI and Secret Key Encryption. When these two methodologies are used together, the common security threats will be reduced since the PKI methodology provides solid authentications and as for Secret Key Encryption will ensure faster and simpler processing time and yet concretely protect data integrity.

7. References

[1] Johnson, D., Perkins, C., Arkko, J., "Mobility Support in IPv6", IETF RFC3775, June 2004

[2] Devarapalli, V., Wakikawa, R., Petrescu, A. and Thubert, P., "Network Mobility (NEMO) Basic Support Protocol". IETF RFC3963. January 2005.

[3] J. Arkko, V. Devarapalli and F. Dupont. Using IPsec to Protect Mobile IPv6 Signaling between Mobile Nodes and Home Agents. Internet Draft, IETF. draft-ietf-mobileip-mipv6-ha-ipsec-06.txt (work in progress). June 2003.

[4] Petrescu, A., Olivereau, A., Janneteau, C. and Lach H.-Y., "Threats for Basic Network Mpbility Support (NEMO threats)" Internet Draft, IETF. Draft-petrescu-nemo=threats-01.txt. January 2004.

[5] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[6] Rescorla, E., "A Survey of Authentication Mechanisms". Internet Draft, IETF. Draft-iab-auth-mech-03.txt. March 2004

[7] Deng, R. H., Zhou, J., Bao, F., "Defending Against Redirection Attacks in Mobile IP" ACM CC @ '02 November 2002

[8] Qiu, Y,. Zhou, J., Bao, F., "Protecting All Traffics Channels in Mobile IPv6 Network" WCNC 2004 @ '02 November 2002

[9] Stallings, W., "Cryptography and Network Security", Third Edition, Prentice Hall

[10] H. Krawczyk, M. Bellare and R. Canetti, "HMAC: Keyed-Hashing for Messaging Authentication", *IETF RFC 2104*, February 1997.

[11] P. Thubert., R. Wakikawa., V. Devarapalli., "NEMO Home Network Models". Internet Draft, IETF. Draft-ietf-nemo-home-networl-models-02.txt. February 2005.

[12] T. Ernst., H-Y. Latch., "Network Mobility Support Terminology". Internet Draft, IETF. Draft-ietf-nemo-terminology-02.txt. October 2004.

[13] Zhao, F., Wu, F.S., Jung, S. and Kim, H., "Secure Reverse Routing Header Solution in NEMO", Internet Draft, IETF. Draft-zhao-nemo-rrh-security-00.txt, July 2004.

A NEW PROPOSED PROTOCOL OF ROUTER'S CA CERTIFICATE

Wafaa A-H Al-Salihy and Azman Samsudin
School of Computer Science
University Sciences Malaysia
11800 Penang
Malaysia.
wafaa@nrg.cs.usm.my, azman@cs.usm.my

Abstract- **The CA certificate is the certificate that issued by a Certificate Authority (CA) to an entity that is also allowable to issue a sub-certificates to the end nodes. In this paper we present a protocol of CA certificate that issued to particular routers in the Internet, which in turn, the router is allowable to issue sub-certificates to the connected nodes. The format of the CA certificate and the sub-certificate is extension to what had presented in user's digital certificate. However the protocol for issuing, renewing and revoking the router's CA certificate and the node's sub-certificates is new and first time presented in the literature. Analysis of the security consideration of each part of this protocol is also presented. By the introduction of this protocol we are able to avoid the replay attack, man in the middle attack and denial of service attack (flooding type). denial of service attack (blocking type) is not considered as this type of attack is not associated to particular protocol.**

I. INTRODUCTION

One of the most interesting solutions in providing repudiation and authentication for users is digital certificates [1] and [2]. In the last years digital certificates have become more and more popular for securing Internet connection due to the simplicity in providing authentication and also integrity of data. For instance, we can find the use of digital certificates in many applications such as emails [3], e-Commerce and web browser [4] and [5]. However, in most cases the digital certificates are issued only to users. With the advance of the new technology that we are experiencing today and the rapid development of digital devices and digital homes, with IPv6 [6] and Mobile IPv6 protocol [7] deployment, all the devices are expected to be reached by their IP addresses. This phenomenon in the future involves a new security issues which make a new security protocols based on the concept of certificates is a need.

Two major types of certificates exist: end-entity certificates and *CA* certificates. End–entity certificates are issued by a Certificate Authority to an entity that does not in turn issue certificates to another entity. While *CA* certificates are issued by a Certification Authority to an entity that is also a Certificate Authority and so may issue end- entity certificates. Certificates can be issued to users as well as routers and nodes. The Certificate for users is described and used in many ways. However, the certificates of routers and nodes still remain as a new research field

This paper proposes a new protocol using *CA* certificates concept to be designed for routers. This protocol allows a router to obtain certificates from the *CA* and generate an end entity certificate "sub-certificate" to the nodes connected to the router. This protocol is particularly adaptable in the context of the IPv6 and Mobile IPv6 but can also be applied in other cases such as authenticating and securing routing protocols or any other protocols used in router layers.

II. ROUTER'S CERTIFICATE

In our proposal, we made few assumptions. First, the Certificate Authority (*CA*) that issue certificates to users should also issue certificate to routers, more detail description about the creation of certificates and the format are explained in the next sections. Second assumption is the *CA* should integrate with the Internet Service Provider (*ISP*). *ISP* usually assigns a block of IP addresses to routers and it will be more suitable if the certificates to routers can be issued from the same source. Third assumption is the *CA* should have world wide coverage and the fourth assumption is routers are high end routers in term of speed and memory.

A router certificate issued by a *CA* is required to be installed on the particular router. A certificate authenticated by a *CA* gives the ability to a router to generate sub-certificates for authorized nodes. These sub-certificates are then used when a node communicate with its peers to proof that the node is a trusted node.

A. Router Certificate Creation

The administrator of a router who wants to obtain a certificate will request the certificate from a valid *CA*. The *CA* must verify physically the information given by the administrator to be sure that the router is not a fake router. Following the verification, the *CA* will issue a certificate for the router.

To use a certificate, first the administrator must install the certificate on a particular router. Then specifies which *MAC* addresses of nodes are allowed to ask for a sub-certificate in an ingress list of the router. This ingress list serves to restrain the number of nodes able to obtain a sub-certificate. The *MAC* address is used because of two reasons: first it is more difficult to spoof a *MAC* address and second a *MAC* address will not change in the case of the movement of mobile node from a network to another. Figure 1 shows the router information at this stage. The

administrator also has to install pre-shared secret keys *SK* on the router and on the nodes.

The *MAC* address and the pre-shared secret keys are used to authenticate the request of sub-certificate from a node to the router. This is explained later in this paper in the sub-certificate generation section.

Router Certificate Router Private Key	
Secret key 1 Secret key n	MAC address 1 MAC address n

Figure 1: Router Information

As mentioned earlier, the information or the attributes included in the certificate is an extension to what had been proposed in the user digital certificate, so instead of the field of *User Id* the certificate should include the *Router Id.* It's preferable to add another attribute to show the task of the particular router, for example in Mobile IPv6 the particular router can work as a Home Agent. Other attributes like the issuer of the certificate, the version, serial number and etc will remain the same.

- *Security Consideration of Router Certificate Creation*

There is no security problem during the certificate generation because all the operations are manually done. The only consideration is to be careful in storing the pre-shared secret and the private key on the router and on the node. Additional care need to be considered to avoid an attacker having a physical access to the storage area. This is also true for the table containing the authorized *MAC* address on the router. However, in general, access to routers should be secured by extra encryption. However the encryption is not part of the proposed protocol.

B. Router Certificate Renewing

The renewing of a router certificate is automatic. It needs a few exchanges of messages between the *CA* and the router as shown in Figure 2. The renewing has to be done at least before the expiration of the router certificate because after this date, the certificate is considered invalid by the *CA*. This is done to avoid a sub-certificate, to stay valid after the router's *CA* certificate has expired. A router verifies a certificate expiry date by sending regular request to the *CA*.

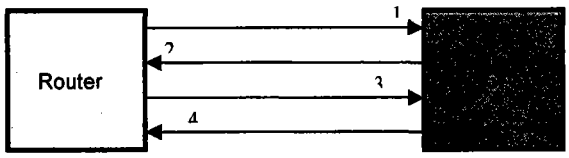


Figure 2: Router Certificate Renewing

Below are the detail descriptions of each message found in Figure2:

[1] Rtr → CA: $SIGN_{Rtr}$ [new Rtr PK, new Rtr PK Id, old Rtr PK Id, n1] + Rtr Cert
[2] CA → Rtr: $SIGN_{CA}$ [new Rtr Cert, n1]
[3] Rtr → CA: $SIGN_{Rtr}$ [n1]
[4] CA → Rtr: $SIGN_{CA}$ [n1]

Message 1: The router sends a certificate renewing request to the *CA*. This message includes the router *PK* and the *PK Id* of the new certificate. The message is signed with the old *PV* corresponding to the *PK* of the old certificate of the router and contains the *Id* of this *PK*. This message also contained a random value n1 to avoid replay attacks on the *CA* response.

Message 2: Firstly, the *CA* verifies if the previous certificate of the router has not been revoked or if the time-validity window has not been expired. Secondly, the *CA* verifies the request message by using the *PK* contains in the old certificate of its database, identified by the *Id* sent. If the message is correct, the *CA* generates a new certificate for the router with the new *PK* and the new *PK Id.* The new certificate is send to the router in the response with the random value previously received, and the message is signed by the *CA*.

Message 3: The router verifies the *CA* signature of the response message by using the *CA's PK.* It also tests if the random value sent and the random value received is the same. If the response is correct, the router saves its new certificate and sends an acknowledgment containing the random value. This message is signed with the *PV* corresponding to the new certificate.

Message 4: The *CA* verifies the signature and the random value, and activates the certificate when it receives the acknowledgment from the router. The *CA* sends a signed message to the router to confirm the activation of the certificate. When the router receives the message, the router can start using the new certificate.

- *Security Consideration of Router Certificate Renewing*

All the messages are signed and the data contained in the messages are not confidential. Man in the middle attack can't be mounted on this section of the protocol because if the attacker uses his own *PV* then there will be a conflict with the key that is associated with the router certificate in Message 1 shown in Figure 2. Replay attacks are avoided by the random value sent with the messages. The attacker cannot possess the certificate because the certificate will never be active as the attacker cannot sign Message 3. With this we can conclude that this part of the protocol is secure.

C. Router Certificate Revocation

If the router's private key is corrupted, the network administrator must notify this situation to the *CA*. After the revocation, the router needs to obtain a new certificate. If the router possesses another valid certificate, it can send a certificate renewing request

to the *CA*. Otherwise, the network administrator has to communicate physically with the *CA*.

All the sub-certificates issued with a revoked *PK* are considered invalid. All the nodes using such sub-certificate are obliged to renew the sub-certificate when they receive an invalid certificate error message.

III. NODE'S SUB-CERTIFICATE

Node's sub-certificate is the certificate that is issued by a router to a node upon the node request and authorization. As mentioned earlier the sub-certificate is a certificate used by a node to authenticate itself to other nodes in other networks. The attributes of the certificate can be similar to the attributes of the router's *CA* certificate but the value of these attributes will be different e.g. the *Node Id*, the issuer and the time of validity. The *Node Id* must be unique for all nodes which belong to the same router. This *Id* is generated by the router.

The *Router Id* and *Node Id*, coupled together will uniquely identify a node. The period of validity and the create time are set by the router and can be verified only by the router so a router does not need to be synchronized with other nodes.

A. Node Sub-Certificate Generation

A node which wants to obtain a sub-certificate must follow the following protocol, as shown in Figure 3. Before sending the first message, the node must possess a private key and the corresponding public key. The way to obtain these keys is not described in this section. They could be generated offline or pre-installed on the node. The first key is used to sign the messages and the second key is used to verify the signature.

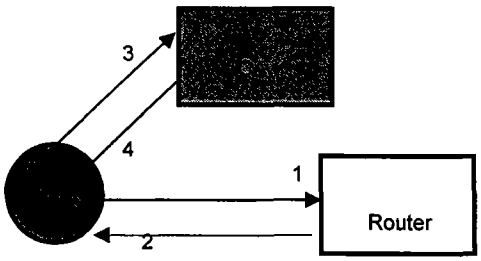


Figure 3: Node Sub-Certificate Generation

Below are the detail descriptions of the messages showed in Figure 3

[1] Node→ Rtr: HASHSK [PKNode + MACNode]
[2] Rtr→ Node: Rtr Cert + Node Cert
[3] Node→ CA: Rtr Cert +n2
[4] CA→ Node: $SIGN_{CA}$ [CA Id, Router Prefix, Router PK Id, n2]

Message 1: The node requests the router to generate a sub-certificate. The message carries the Public Key (PK_{Node}) and the *MAC* address (MAC_{Node}) of the node. These pieces of information are concatenated and encrypted using the pre-shared key.

Message 2: On the router side, the router decrypts the message using the pre-shared key and verify in its ingress list if the *MAC* address of the node is allowed to obtain a sub-certificate. If the verification is positive, the router sends the response to the node which is the router certificate (*Rtr Cert*) and the node sub-certificate (*Node Cert*), otherwise the router will send an error message.

Message 3: The node required the *CA* to know the validity of the router certificate. It sends the entire router certificate (*Rtr Cert*) to the *CA* and a random value n2 to avoid replay attacks on the response of the *CA*.

Message 4: On the *CA* side, the *CA* will check the validity of the router certificate. The *CA* verifies with its *PK* if the router certificate is issued by this particular *CA*. Then the *CA* checks if the time-validity window of the router certificate allows the router to issue a sub-certificate for the nodes. This implies that the period of validity of a sub-certificate can't exceed the time-validity window of the router certificate. A router certificate is also not valid if it has been revoked.

The *CA* replies with an error message if the certificate is not valid. Otherwise the *CA* will positively replies with a message that contain the identification of the certificate, and the random value *n2* that was carried in the request of validity from the node. The response is signed by the *CA* and send to the node.

The node receives and verifies the *CA* response. The node will verify if the identification of the certificate and the random value *n2* correspond to the request of validity and if also correspond to the same router certificate sent in Message 2. If the verifications fail, the node sends an error message to the router. Otherwise, the exchange is complete and the node can begin to use its sub-certificate.

- *Security Consideration of the Node Sub-Certificate Generation*

Message 1: The first message is encrypted to avoid attacker from knowing the content of the message. Moreover, an attacker is not able to send a request to the router to obtain a certificate because of the ingress list. An attacker who able to spoofs the *MAC* address of an authorized node is not able to send a request because the attacker does not posses the *SK* to encrypt the message.

Message 2: A fake router can generate a message using its *PK* as the PK_{CA} but the response of the true *CA* in Message 4 will not be readable and this will cause the procedure to be aborted. A fake node who wants to use the node certificate will also fail.

Message 3: In this case, an attacker can either replaces the router certificate by another certificate or changes the value of *n2*. The modified certificate

must be obligatorily emitted by the *CA* because otherwise, the checking of the *CA*'s signature will fail and the *CA* will return an error message. If the modified certificate is a certificate issued by the *CA*, the identification received by the node will be different from what had been send. The receiving node will know that there is a problem with the certificate. On the other hand if the attacker modifies the random value *n2*, the node will not accept the response message that holds different *n2*.

Message 4: The message is protected by the signature of the *CA* thus the message cannot be generated by an attacker. It is not possible to replay an old response of the *CA* because the random value contained in the reply must be the same with one in the request.

It is possible for an attacker to block the process; however this type of attack is also possible in all communication protocols where the attacker is able to block the transmission of messages. The most important security measure which the proposed protocol has is that an attacker can never obtain a valid sub-certificate which can use for illegal authentication.

B. Node Sub-Certificate Verification

After a node obtains a certificate, the node can use the certificate to be authenticated by the correspondent node as shown in Figure 4:

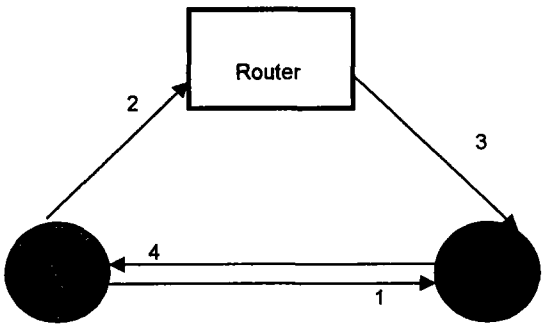


Figure 4: Node Sub-Certificate Verification

Below are the detail descriptions of the messages used in Figure 4

[1] Node→ Cn: Node Cert
[2] Node→ Rtr: HASHSK [initiate verification +n3 +t]
[3] Rtr→ Cn: SIGNRtr [t] +Rtr Cert + Node Cert + HASHPKNode [n3]
[4] Cn→ Node: Response of verification+ HASHPKNode [n3]

Message 1: The node sends a node sub-certificate to its correspondent *Cn* when the node wants to authenticate itself to that *Cn*. At the same time the node sends Message 2 to initiate the router to sends the information needed by *Cn* for the authentication purpose.

Message 2: The node request the router to initiate the verification process on *Cn*. The message has random value *n3* which has time-window of t. This message is encrypted using the pre-shared key for confidentiality.

Message 3: The router sends the node sub-certificate with its router certificate, and the random value *n3*. The message encrypted with the node's *PK*. This message has time-window of *t* and is signed by the router.

Message 4: On the *Cn* side, first, *Cn* verifies the time-window if the time-window has expired or not, and at the same time check if this router *PK* is the same as the router certificate's *PK*. If all the checks are valid then the process will proceed else *Cn* will send an error message to the node indicating the error. Secondly, *Cn* verifies the router certificate by using PK_{CA}. If the certificate is valid, the router uses the router's *PK* to verify the node sub-certificate, and at the same time compare the certificate with the received certificate from the node. If the comparison is successful then *Cn* will send Message 4 as positive response with the encrypted *n3*, otherwise the *Cn* will sends an error message to the node.

Error message can also be send if the validity period of the node sub-certificate is not valid anymore. Also an error message can be send if the router's *PK* cannot open the node sub-certificate or the time-validity window of the message received has been expired.

After this phase, the *Cn* obtains enough insurance that the node is a trusted node. All messages send by the node can be signed and authenticated by *Cn*.

- *Security Consideration of the Node Sub-Certificate Verification*

Message 1: There is no potential attack on this message. The node can send its certificate to any other node. An attacker can change the certificate but then the verification done by *Cn* will produce an error message.

Message 2: This message is encrypted with the pre-shared key. Because of the encryption an attacker cannot decrypt the message or generate another message. What an attacker can potentially do is mounting a replay attack by replaying the message before time *t* expires. However the value of *n3*, in this case, will be different from what the node should receive in Message 4. Moreover the attacker is not able to mount the denial of service attack (flooding) by continue sending this message to the router because the message will expire soon.

Message 3: In this case, there are two possibilities. The attacker can either replaces one of the information such as the router certificate or the node sub-certificate or the encrypted *n3* or replaces all the information by replaying an old message from the router. For the first possibility, the modified router certificate must be obligatorily emitted by the *CA* because otherwise, the checking of *CA*'s signature

will fail and the *CA* will return an error message. If the modified certificate is a certificate issued by the *CA*, the Public Key of the router will not be able to open the node sub-certificate, hence the verification will fail. Similar for the node sub-certificate, either the certificate is not similar to the one sent from the node or the certificate can't be verified with router *PK*. For the encrypted *n3*, an attacker is not able to generate the value. However an attacker can replay old message but then the old message has a different value of *n3*.

For the second possibility, the attacker can replay the whole message but then the node sub-certificate is not similar to the one received from the node in Message 1. If an attacker manages to replay Message 1 and 3, still the attacker is not able to authenticate itself to other nodes.

Message 4: There should be no potential attack on this message because *n3* should be similar to the one sent before, otherwise an error will be detected.

C. *Renewing and Revocation of Node Sub-Certificate*

A sub-certificate must be renewed if the period of validity has expired. The renewing should be activated by the router that issued the node sub-certificate. The process of renewing a certificate is exactly the same as the process of generating a new certificate. The only difference is that Message 1 is being added, which is sent by the router when the renewing need to be activated. Figure 5 shows the message activated by the router.

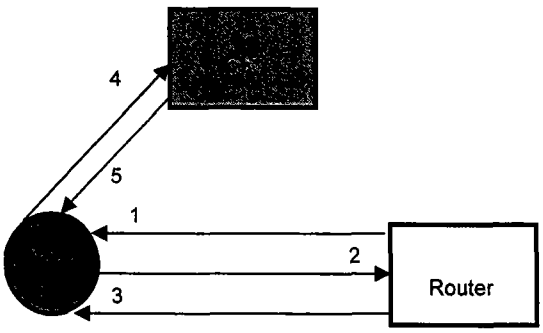


Figure 5: Node Sub-certificate Renewing

The renewing process is as follows

[1] Rtr→ Node: Initiate renew + HASHSK [n4 + t]
[2] Node→ Rtr: HASHSK [new PKNode + old PKNode + MACNode +n4]
[3] Rtr→ Node: Rtr Cert + Node Cert
[4] Node→ CA: Rtr Cert +n5
[5] CA→ Node: SIGNCA [CA Id, Router Prefix, Router PK Id, n5]

Message 1: This message initiates the renewing process of the node sub-certificate. The message carries random value *n4* initiated by a router and encrypted with the pre-shared key *SK*.

Message 2: The node replies the router to generate a sub-certificate. The message carries the new Public Key (new PK_{Node}), the old *PK* node, the *MAC* address (MAC_{Node}) of the node and random value *n4*. This information is concatenated and encrypted using the pre-shared key.

Message 3: On the router side, the router decrypted the message using the pre-shared key then verify in its ingress list if the *MAC* address of the node is allowed to obtain a sub-certificate and whether *n4* is similar to what has been received in Message 2. The router sends a reply to the node with a message that contains router certificate (*Rtr Cert*) and the node sub-certificate (*Node Cert*); otherwise the router will send an error message.

Message 4: The node requests the *CA* to check the validity of the router certificate. The node sends the entire router certificate (Rtr Cert) to the *CA* and a random value *n5*. The value of *n5* is used to avoid replay attack on the response of the *CA*.

Message 5: On the *CA* side, *CA* checks the validity of the router certificate. The *CA* verifies with its PK if the router certificate is issued by the *CA*. Secondly, it checks if the time-window of router certificate's validity allows issuing a sub-cert for nodes. The *CA* considers the router is not allowed to issue a node sub-certificate with a time-window of validity exceed the time-window of validity of the router certificate itself. A router certificate is also not valid if it has been revoked.

CA replies with an error message if the router certificate is not valid or the *CA* replies with a message that contains the identification of the certificate, which are *CA Id*, the Router Prefix, the Router *PK Id* and the random value *n5*. The random value *n5* should be similar with the one sent in Message 4. The message is signed by the *CA*.

The receiving node firstly verifies if the *CA Id*, Router Prefix, the Router *PK Id* and the random value *n5* correspond to the request of validity. Secondly, the node checks the signature of the router certificate using PK_{CA}. If the verifications fail, the node sends an error message to the router. Otherwise, the exchange is complete and the node can begin to use its new sub-certificate. If the router's private key was corrupted, the network administrator must acknowledge the situation to the *CA*. After the revocation, the router needs to obtain a new certificate.

- *Security Consideration of Node Sub-Certificate Renewing*

Message 1: An attacker cannot generate this message to the node. What the attacker can do is to replay the message before time *t* expires. But when the router receives a reply from the node, the router will ignore the message because the router does not send any activation for renewing. The attacker is not able to mount the denial of service attack (flooding) by continue sending this message to the node because the time-window of this message will expire soon.

Message 2: The message is encrypted to avoid an attacker from knowing the content of the message. Moreover, an attacker is not able to send a request to the router to obtain a certificate because of the ingress list and the value of *n4*. An attacker who spoofs the *MAC* address of an authorized node is not able to send a request because the attacker doesn't have the pre-shared key to encrypt the message.

Message 3: A fake router can generate the message using its *PK* as the PK_{CA} but the response of the true *CA* in Message 4 will not be readable and this will cause the process to be aborted. A fake node who wants to use the node certificate will also fail the certificate verification process.

Message 4: On this message an attacker can either replace the router certificate by another certificate or change the value of *n5*. The modified certificate must be obligatorily emitted by the *CA* because otherwise, the checking of its signature will fail and the *CA* will return an error message. If the modified certificate is a certificate issued by the *CA*, the identification received by the node will be different from what was sent and the node will detect an error. On the other hand if the attacker modifies the random value *n5*, the node will not accept the response message that holds different value of *n5*.

Message 5: The message is protected by the signature of the *CA* thus the message cannot be generated by an attacker. It is not possible to replay an old response of the *CA* because the random value contained in the reply message must be the same with one in the request message. Regarding the revocation, the same process can be executed as found on the router certificate revocation process.

IV. CONCLUSION

In this paper we designed a new protocol for router's *CA* certificate that can be issued by a known *CA* which in turn the trusted router is allowed to issue the node sub-certificate. Our new proposed protocol requires a few messages and provides strong authentication and security.

The protocol is securing against man-in-the-middle, IP spoofing, replay attacks, and the denial of service type of flooding attack. We do not discuss in this protocol other attacks such as denial of service type of blocking, because such attack not only specific to our protocol. Moreover, our protocol does not transgress the rule of layers violation because the protocol uses only information from the network layer (*IP* layer) which is the layer implementing by routers. The only requirement of this protocol is more deployment of existing *CA* to provide the required services by the router's certificate. *CA* is preferred to be deployed and taken by ISP, which has all the information of the routers in each area. This evolution seems to be logical and the required services are completely feasible.

REFERENCES

[1] R. Housley, W. Ford, W. Polk, and D. Solo, Internet X.509 Public Key Infrastructure Certificate and CRL Profile, *RFC 3280*, April 2002, http://www.ietf.org/rfc/rfc2459.txt

[2] A. Nash, W. Duane, C. Joseph, D. Brink, "PKI: Implementing and Managing E-Security" (California, U.S.A: McGraw-Hill, 2001).

[3] D. Atkins, W. Stallings, and P. Zimmermann, " PGP Message Exchange Formats", *RFC 1991*, August 1996, http://www.ietf.org/rfc/rfc1991.txt.

[4] A. O. Freier, P. Karlton, and P. C. Kocher," The SSL Protocol Version 3.0", *draft-freier-ssl-version3-02.txt*, November 1996.

[5] T. Dierks and C. Allen, "The TLS Protocol Version 1.0", *RFC 2246*, January 1999, http://www.ietf.org/rfc/rfc2246.txt

[6] S. Deering, and R. Hinden," Internet Protocol, Version 6 (IPv6) Specification", *RFC 2460*, December 1998, http://www.ietf.org/rfc/rfc2460.txt

[7] D. Johnson, C. Perkins,& J. Arkko, Mobility Support in IPv6,IETF, *RFC 3775*, June 2004.

Efficient Search on Encrypted Data

Lee Thian Aun Joseph, Azman Samsudin, and Bahari Belaton
School of Computer Science, Universiti Sains Malaysia
Penang, Malaysia
{joseph,azman,bahari}@cs.usm.my

Abstract—**Encrypted data are being kept in remote server for purposes like backup and space savings. In order to retrieve these encrypted data, efficient search methods were proposed that enable the retrieval of the dataset without leaking too much information thus ensuring better security and less information leakage. An improved method is proposed in this paper for an efficient search on encrypted data which implements a keyword list in a hash table for each encrypted document. The keyword is encrypted in such a way that by providing the file server with required search information known as "a capability for a certain keyword", searches can be performed without leaking any information.**

Keywords—**Search, Encrypted Data, Bloom Filter, Linear Scan, Encrypted Index.**

I. INTRODUCTION

As we advance into the digital age, more and more information are stored in computers. These data are becoming much increasingly important as it consists either personal details, money account or technology researches. To thwart people from reading the contents of the information stored, encryption is introduced where the owner have the 'key' that allows the accessing of the information. These encrypted data is stored in a database for safekeeping. In order to retrieve the information, the owner will have to select the correct file and decrypt it. As the amount of documents grow, it would not be feasible to decrypt all documents to find the needed document. Furthermore, if the encrypted data is kept in an untrusted storage on a different location, it would be unwise to decrypt the data. Therefore a search method is needed to find the needed document without decrypting first to ensure better security and less information leakage[1-6].

Due to this an efficient search method in getting the correct encrypted document based on certain keyword by the user is needed. This saves time involving in decrypting the documents and does not leak any information on the untrusted storage area.

II. LITERATURE SURVEY

Song, Wagner and Perrig (SWP) [1] presented two methods of searching encrypted data, which are Linear Scan and Encrypted Index method. The first method will be known in this paper as SWP Linear Scan while the later as SWP Encrypted Index. Other than SWP methods, Goh[2] presented a method that uses Bloom Filter[3,4] to search on encrypted data. This method will be known in this paper as Goh Bloom Filter.

This work was supported by Universiti Sains Malaysia.

A. SWP Linear Scan

In the paper by SWP, four schemes were introduced as proof of concept for SWP Linear Scan. The four schemes; Basic Scheme, Controlled Searching, Hidden Searches, Final Scheme will be visited in detail below.

In this scheme, we have Alice, owner of a set of documents D represented as d_1, d_2,..., d_t where t represents the number of documents and Bob the owner of the file server FS where the documents are to be kept. Before Alice gives the document to Bob, the document is arranged in a sequence of words W_1, W_2, ..., W_l where l is the number of words in a document. Each of this word will be allocated a fixed length of n bits. The encrypted document that is sent to Bob is derived from an XOR function of each word W_i with another fixed length random bit array T_i for every position i in the document. The result of the XOR function will be the cipher text $C_i = W_i \oplus T_i$.

In order to generate the T_i inputs for the XOR function, a stream of pseudorandom bits S_1, S_2, ..., S_l where l is the number of words in the document will be generated from a pseudorandom number generator G_i with a secret seed. These pseudorandom bits are n-m bits long. The pseudorandom bit S_i will act as an input for a function F with a key k_i to generate the rest of the m bits. The key k_i used here can be same or different for all i position. Both the combined pseudorandom n-m bits S_i and the generated m bits $F_{ki}(S_i)$ will serve as the input of the XOR function $T_i := \langle S_i, F_{ki}(S_i) \rangle$.

With the generated input T_i and word W_i, Alice can now XOR every word in the document and sent the list of cipher text C_i to Bob for safekeeping. The same process is done to every document in set D.

B. Basic Scheme Searching Technique

Now that the documents are with Bob, Alice would like to find documents that contain a certain word W_i that Alice wants. Here, Alice will send the key k_i with the word W_i where Bob can do an XOR function to get the value of $T_i = W_i \oplus C_i$ where $T_i := \langle L_i, R_i \rangle$ and $L_i = S_i$; $R_i = F_{ki}(S_i)$. By using the key k_i with the function F, Bob will be able to check whether $F_{ki}(S_i)$ equals to R_i, the m bits of the cipher text block (Figure 1).

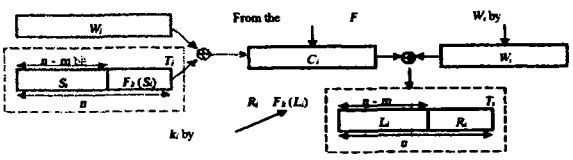


Fig. 1. Basic Scheme Searching Technique

C. Controlled Searching Scheme

SWP made improvement by introducing controlled searching. Here, another function f is used to generate the value of k_i. The word W_i will be applied to the function resulting a newly generated key $k_i = f_k(W_i)$ (Figure 2). The value of k' is kept secret from Bob, and only the generated value k_i is given to Bob. By doing this, the value k_i is independent on the position of words and thus Alice does not need to know the location of the word prior to the search. The search is performed identical to the Basic Scheme Searching Technique. The values of $f_k(W)$ is dependent on word W and this allow Bob to reveal all the position i where W occurs but not other position where $W_i \neq W$ (Figure 3).

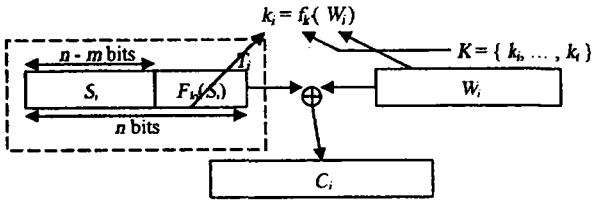


Fig. 2. Controlled Searching Scheme

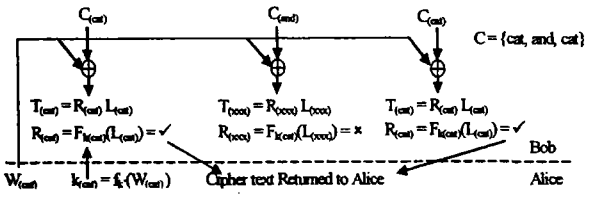


Fig. 3. Controlled Searching Scheme Example

D. Hidden Search Scheme

Both the Basic Scheme and Controlled Search Scheme allows Bob to know what word W that Alice is searching. To prevent this, hidden searches method is introduced where the W is first encrypted using a deterministic algorithm $E_{k''}$. The prerequisite in this method is that the encryption method E is not allowed to use any randomness and must rely on W only without the knowledge of position i of the word. An implementation of this scheme is to use Electronic Codebook mode (*ECB*) on the word W. For a longer document, the Cipher Block Chaining Mode (*CBC*) can be used where word W is encrypted using a constant initialization vector (*IV*) but must be same for every position.

Now Alice will take every word in the document W_1, W_2, ..., W_t and encrypt it using the function E with a key k''. This will result in the cipher text X_1, X_2, ..., X_t where $X = E_{k''}(W)$. The input for the XOR operation, T_i is generated with a change where encrypted word X_i is used with pseudorandom bits S_i, resulting $T_i := \langle S_i , F_{k'}(S_i) \rangle$, $k_i = f_k(X_i)$.

E. Final Scheme

The Final Scheme presented by SWP allows the returned cipher text of the document to be decrypted. The words other than what Alice searched for cannot be decrypted because Alice is unable to determine the value of $k_i = f_k(E_{k''}(W_i))$ to generate the R_i m bits of T_i. Alice will not know the encrypted word $E_{k''}(W_i)$ for every position i of the document (Figure 4).

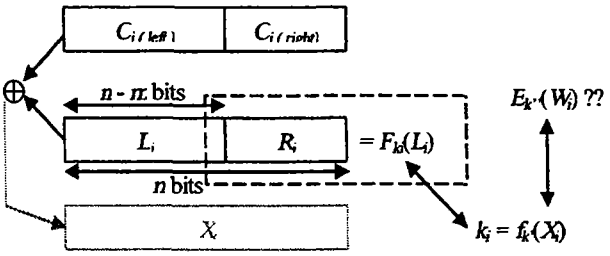


Fig. 4. Final Scheme

F. SWP Encrypted Index

SWP proposed the use of an index to speed up the search for document based on keywords. In this method each keyword W_i is attached to a list of document pointer P where each pointer in the list p_i points to a document d_i, $p_i \rightarrow d_i$. The keywords and pointers form the rows of the index i (Figure 5).

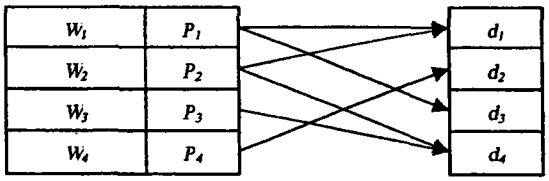


Fig. 5. SWP Encrypted Index

The keyword and the document pointers in each list in the index are first encrypted. Alice will send the encrypted word $E(W_i)$ and $E(P_i)$ to Bob for safe keeping. When Alice wants to retrieve the documents, Alice will send the encrypted word $E(W_i)$ and get the returned encrypted list of pointers $E(P_i)$. With this, Alice can decrypt the encrypted list and send another request for the documents. As noted, this method will take two trips.

To save a trip, Alice can encrypt the list of document pointer in the index $E_{kp}(P_i)$ using key $kp = F_{kw}(E(W))$ related to the encrypted word. Searching can be done when Alice reveal $\{E(W), kp\}$. Bob will be able to decrypt the encrypted pointer list $E_{kp}(P_i)$ and perform another search for the document on behalf of Alice (Figure 6).

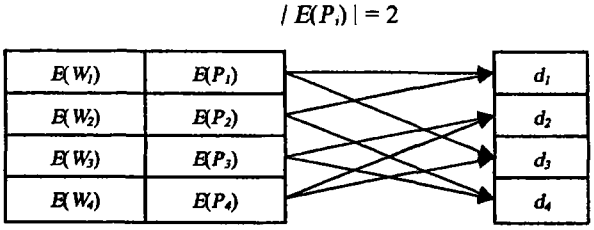


Fig. 6. SWP Encrypted Index with Encryption

Bob can be prevented from doing a statistical analysis on the index if the list of pointers is kept in fixed size list where infrequent keywords are padded up to fixed size with false

documents (document that does not contain the keyword). Common words are split into few where several search queries have to be merged and done in parallel.

G. Goh Bloom Filter

Goh introduced the method that uses the bloom filter hash coding by Bloom. In this method the document D are represented with a set of words $S = \{s_1, s_2, \ldots, s_n\}$ where n is the number of words chosen by Alice. Each elements of set S represents an array of m bit. The conversion of the words in set S is done by applying r independent hash function h_1 to h_r where $h_i : \{0,1\}^* \rightarrow [1,m]$ for $1 \leq i \leq r$. For each element in S the array bits are hashed $h_1(s_i), \ldots, h_r(s_i)$. The location of each distinct bit of the hashed value will set the bit address in the hash area to 1. Bit addresses with multiple set are not changed and remain the value of 1.

To determine membership of a word s_i in set S the hash value of the generated word $h_1(s_i), \ldots, h_r(s_i)$ must all have the value of 1 in the hash area. Bloom filter sacrifices space and time for allowable error [3]. These allowable errors are known as false positives. False positives are words s_i that are not a member of the set S but proven by bloom filter checks as member. This is due to the bits set by a collection of other words in set S (Figure 7).

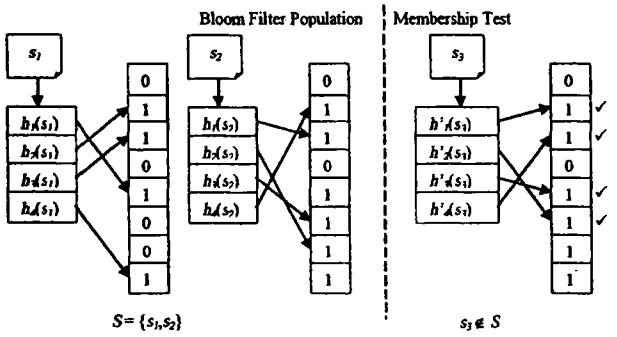


Fig. 7. Goh Bloom Filter

III. METHODOLOGY

Based on the search methods analyzed earlier, a hybrid method of all three methods will be proposed. The motivations behind the creation of this method are:

1) Have a method that allows the owner of the data to find the required data from a remote and untrusted storage
2) Supports any types of data
3) Allows Alice to choose just the required keywords describing the data
4) Preserve the keywords where the keywords can be retrieved if needed
5) Time complexity of O(1) to search for a keyword
6) Easy integration with any existing indexing scheme
7) Good performance time in terms of encryption, decryption and search

With these objectives in mind, the new search method is described as below. It also maintains the needed securities from the methods analyzed earlier.

A. Scheme 1

When it comes to having a fast and efficient search, methods like hash tables and trees are deployed to reduce the time needed. The common architecture is that each of them has to build a kind of index representing the data which can be accessed based on a certain function [7]. This results in an O(1) time complexity search time for the best case while the worst case is O(n) time complexity. Scheme 1 will incorporate indexing.

1) Setup / Encryption Phase

In this method, the keywords $W_1, W_2, \ldots, W_t$ where t is the number of keywords belonging to a document D will be organized into a hash table known as HT. The keywords are allocated to different location of the hash table with the use of a hash function $H: \{0,1\}_m \rightarrow \{0,1\}_n$ where m represents number of binary of the word to be hashed and n represent the number of binary digit for the allocated cells in the hash table HT.

It would be tempting to just insert the encrypted word into the location defined by the hash function and thus creating a complete encrypted index. However, this can be dangerous as the single encrypted word is prone to analysis attack where the same encrypted word will record the same value in different document within the hash table.

Due to this, it would be better to insert a different value in the hash table. However the value should allow the keyword to still be searchable. This brings Scheme 1 to utilize the SWP idea that generates a different value for each encrypted word. In SWP method, the random number generator allows this attribute to work. Therefore the creation of the cipher text C_i is done through the XOR product of the encrypted word $E_{k'}(W)$ with the random number block T_i. With $Loc(W_i)$ determining the location in the hash table HT, the value C_i can be stored (Figure 8).

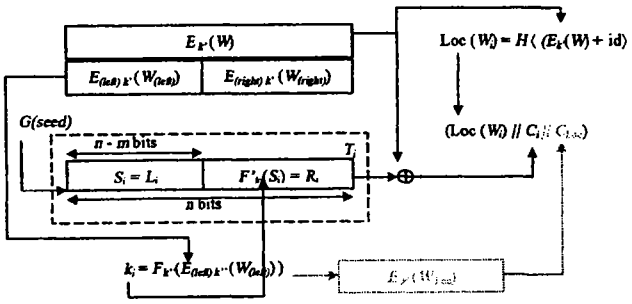


Fig. 8. Generating different cipher text for storing

At this point, the cipher text C_i can also be a candidate for determining the location instead of the encrypted word $E_{k'}(W)$. The reason cipher text C_i is not used is due to the search phase where it can skip the process of recreating cipher text C_i just to find the location of the encrypted word.

2) Search Phase

Although the setup phase and decryption phase consist of quite a number of steps, the search phase is still quite simple. The server will just require either 3 or 4 value depending on the search mode.

a) Single Document Searching Mode

For searching on a single document, the server would require the document number and location of the cipher text in the hash table to perform a direct search of O(1) time complexity (Figure 9).

To check whether the word exists on the server, Bob will need to do an XOR operation of the encrypted word and cipher text, generating the other half of S_i with function F' and key k_i. A comparison of the generated portion and the existing portion will check if the encrypted word is the one that is being searched.

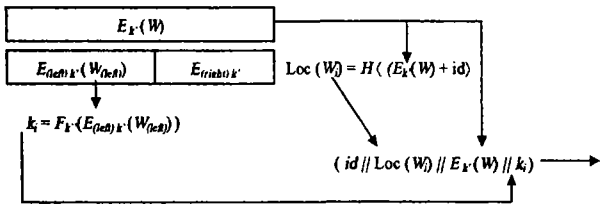


Fig. 9. Single Mode Search

b) Multiple Documents Search Mode

This search is performed when there is a need to find a certain word in multiple documents or the document number is unknown. Without the document number, search can still be performed. This is possible as the hash function just required the encrypted word value to enable the hash function H to find *Loc*. This allows Bob to do the hash function H on behalf of Alice. The only information needed by Bob would only be the encrypted word $E_{k''}(W)$ and key k_i. Bob would need to find the possible location of the word by doing the hash function H on the given value $E_k(W)$ with the document id for all hash tables (Figure 10).

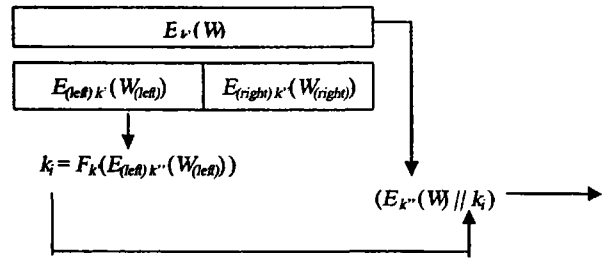


Fig. 10. Multi Mode Search

3) Decryption Phase

With two public values known for every cipher text C_i, Alice would need to be able to decrypt the whole keyword list. The location for each C_i now plays an important part here as the value is used to generate the S_i for each C_i to be XOR resulting $E(W_i)$. Without *Loc*, cipher text C_i cannot be decrypted. The decryption process is similar to SWP method where half of the encrypted word will be derived from the C_i allowing the other half to be derived next. Both portion of the encrypted word would allow decrypting of the word possible (Figure 11). If the word location W_{Loc} is available it can be decrypted with the key dependent word k_i.

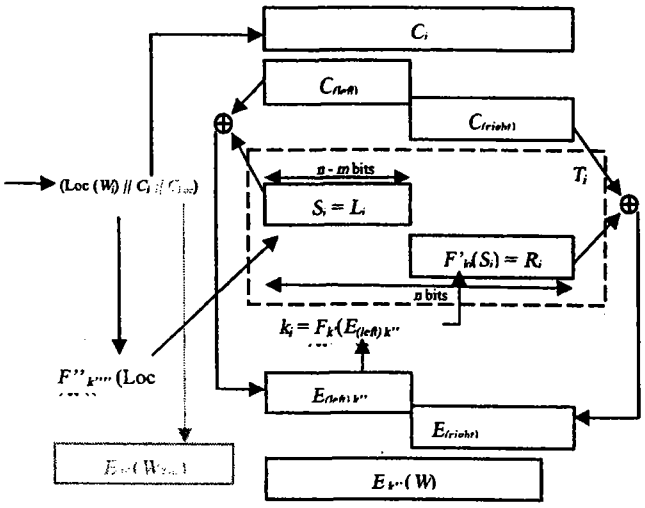


Fig. 11. Decrypting the keywords

IV. Discussion

The discussion will revolve around the three main methods studies earlier, which are SWP Linear Scan, SWP Encrypted Index and Goh Bloom Filter in comparison with the new proposed method (Scheme 1).

TABLE 1: SUMMARY OF PROPERTIES

	SWP		Goh	Improved
	Linear Scan	Encrypted Index	Bloom Filter	New Method
Exact Location	✓	x	x	x/✓
Controlled Search	✓	✓	✓	✓
Variable Keyword Length	x/✓	✓	✓	x
Boolean Queries	✓	✓	✓	✓
Proximity Search	✓	x	x	x/✓
Regular Expression	✓	✓	✓	✓
Occurrences	✓	✓	✓	✓
Data Type	Text	Any	Any	Any
Key Management (no[subkeys])	3	2	1[4]	1[3]
Time/Work Cost (Setup) (per word / per doc)	5r O(n) / O(nm)	4r O(n) / O(nm)	4r O(n) / O(nm)	6r / 8r O(n) / O(nm)
Time/Work Cost (Deletion) (per word / per doc)	1r ----- / O(m)	2r ----- / O(nm)	1r ----- / O(m)	1r ----- / O(m)
Time/Work Cost (Search) (per word / per doc)	2r O(n) / O(nm)	2r O(1) / O(1) O(n) / O(n)	1-2r O(1) / O(m)	1-2r O(1) : O(m) O(n) : O(nm)
Space Cost (per word / per doc)	O(n) / O(nm)	O(1) / O(1+m)	O(1) / O(m)	O(1) : O(m) O(n) / O(nm)
Encryption Method	Any	Any	Any	Any
Decryption	✓	✓	x	✓
Precision	x	x	x	x/✓

Aspects that will be discussed are search properties, data types supported, space cost; time/work cost, key management, encryption methods, decryption and precision. The below table shows a list of the properties for all four search method analyzed in this paper (Table 1).

V. Prototype and Result

The four methods SWP Linear Scan, SWP Encrypted Index, Goh Bloom Method and Scheme 1 will be studied in

detail. The prototype for the three methods are coded and ran on the below computer specifications.

Processor	:	Intel(r) Pentium(r) 4 2.60GHz
Memory	:	496 DDR RAM
Operating System	:	Windows Server 2003 Standard Edition
Programming Language	:	C# .NET
Encryption Algorithm	:	AES 128 bit key, 128 bit block, 128 bit iv, CBC mode, PKCS7 padding
Cryptographic Hash Algorithm	:	HMAC-SHA1

The tables below (Table 2, 3, 4, 5) show the result of executing all the four methods.

A. Method Comparison Discussion

Below is the total processing time of each method where the preparation time, post processing time and processing time is added up (Table 6). A comparison of time for all method can be seen from Appendix P, Q, R, S and T.

TABLE 2: SWP LINEAR SCAN

	No of Words = 980	No of Words = 980	No of Words = 980
	Encryption(seconds)	**Decryption(seconds)**	**Search(seconds)**
Prep Time	0.207021562246983	0.0007169371939700004	0.60043308478235725
Post Time	0.00137466322242356	0.00530519335312488	
Processing Time	0.311217135	0.267862917	0.11645776
Average Time (per word)	0.000319525	0.000275013	0.0001195664886620310

TABLE 3: SWP ENCRYPTED INDEX

	No of Words = 375	No of Words = 375	
	Encryption(seconds)	**Decryption(seconds)**	**Search(seconds)**
Prep Time	0.00134303890195474	0.00293470331958206	0.000616595597023557
Post Time	0.0146203045841848	0.00652851756178432	
Processing Time	0.07026838	0.099963504	0.000063353510429116 7
Average Time (per word)	0.000190429	0.0002709038045386810	

TABLE 4: GOH BLOOM METHOD

	No of Words = 375		
	Encryption(seconds)	**Decryption(seconds)**	**Search(seconds)**
Prep Time	0.00222837799282878		0.00120500443382041
Post Time	14.136435927054	0.00314664301962447	
Processing Time	0.438177299		0.00223311
Average Time (per word)	0.001190699		

TABLE 5: SCHEME 1 METHOD

	No of Words = 375	No of Words = 375	
	Encryption(seconds)	**Decryption(seconds)**	**Search(seconds)**
Prep Time	0.00325021243783013	0.0284156517716004	0.000258450861703358
Post Time	0.0366460161159733	0.00567701893048541	
Processing Time	0.183642046	0.1246849	0.017614753
Average Time (per word)	0.000494992	0.0003360778963998080	

1) SWP Linear Scan

Has a long processing time where each word of the file is encrypted and preserved. The search time increases as with the number of word making this method not suitable for files with many words

2) SWP Encrypted Index

This method has the fastest processing time. However this method is not feasible where a single master index manages all the documents. The reason for this is that any changes to the documents in the file server whether adding, removing or editing a single document will affect the whole index which promotes information leakage. This will also require a pool of keywords to be maintained.

3) Goh Bloom Filter

Goh Bloom Filter has very high security where the keywords are hashed and thus irretrievable thru any means. With each document having a single bloom filter which acts as an index for searching, this allows a fast search time without leaking much information. However bloom filter does not allow preservation of keywords and may prove to be a problem if there is a need for keyword retrieval.

Usage of bloom filter has a disadvantage where it requires a large index size to ensure that the false positive percentage is of acceptable level [2]. With the increase of index size, it requires a longer processing time for creation of file buckets and management to hold the index information. This disadvantage makes this method not suitable for active file server where changes to document occur frequently.

TABLE 6: TOTAL PROCESSING TIME

Total Time	**Encryption(seconds)**	**Decryption(s econds)**	**Search(seco nds)**
SWP Linear Scan	0.51961	0.27389	0.11689
SWP Encrypted Index	0.08623	0.10943	0.00068
Goh Bloom Filter	14.57684	0.00315	0.00344
Scheme 1	0.22354	0.15878	0.01787

B. Scheme 1

Scheme 1 which is a hybrid of Goh Bloom Filter, SWP Encrypted Index and SWP Linear Scan allow this method to inherit good properties/attributes from these methods which gives an average processing time for setup, decryption and search.

In term of setup, it only keeps meaningful searchable keyword like Goh Bloom Filter and SWP Encrypted Index which gives a much better performance time as well as support for different file types. This method follows the same setup model as SWP Linear Scan during its encryption thus preserving the keywords for retrieval if needed.

Using a single index to document model like Goh Bloom Filter allows changes to a document does not affect the security of other files.

In term of searching, indexing allows a good search time however not as fast as Goh Bloom Filter and SWP Encrypted Index. With an average time, this method is also suitable for active file server where changes to document occur frequently.

VI. SUMMARY

Three different encrypted search methods which are SWP Linear Scan, SWP Encrypted Index and Goh Bloom Filter were analyzed in detail and evaluated on. From the studies we find that an efficient search method on encrypted data has the following attributes: Controlled Search, Variable Keyword Length, Boolean Queries, Proximity Search, Regular Expression, Data Type, Key Management, Space Cost, and Search Cost. Focus has been put into the new proposed method to incorporating the good attributes listed above. With the specification of the new proposed method outlined, the attributes that are incorporated into the new method are: Controlled search (Able to search on a particular encrypted data based on given data ID), Boolean Queries (Process multiple queries and results merge based on Boolean command), Variable Keyword Length (Partial support by splitting long words), Regular Expression (Wildcards in queries are preprocessed as multiple queries), Data Type (A separate search index created where searches are performed on the index without any dependencies on the actual data), Key Management (Utilization of pseudorandom number generator to create the required sub keys from a single master key), Space Cost (Each document has an index with a document) and Search Cost (Uses hash table for a constant O(1) time complexity access at best).

REFERENCES

[1] Dawn Xiodong Song, David Wagner, Adrian Perrig. Practical Technique for Searches on Encrypted Data. In proceedings of IEEE symposium on Security and Privacy, IEEE, 2000

[2] Eu-Jin Goh. How to Search Efficiently on Encrypted Data. October 7, 2003

[3] Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. Communication of the ACM, Vol 13 / Number 7 / pg 422-426, July 1970

[4] Sarang Dharmapurikar. CSE 535: Lecture 5 – String Matching with Bloom Filters. Washington University, Fall 2003

[5] Dan Boneh, Rafial Ostrovsky, Giovanni Di Crescenzo, Giuseppe Persiano. Public Key Encryption with Keyword Search. In proceedings of Eurocrypt 2004, LNCS 3027, pp. 506-522, 2004

[6] Benny Chor, Oded Goldreich, Eyal Kushilevitz, Madhu Sudan. Private Information Retrieval, April 21 1998

[7] Peter K. Pearson. Fast Hashing of Variable-Length Text Strings. Communications of the ACM, Vol 33 / Number 6. June 1990

Heuristic Cryptanalysis of Classical and Modern Ciphers

Ho Yean Li, Azman Samsudin, and Bahari Belaton
School of Computer Science, Universiti Sains Malaysia
Penang, Malaysia
{yeanli,azman,bahari}@cs.usm.my

***Abstract*—Block cipher algorithms are commonly used to secure confidential information in everyday user applications However, it is quite common for ignorant users to use familiar dictionary words as their personal passwords. This research will examine the effects of weakly chosen password-keys on the security of block ciphers. A new hybrid optimization heuristic cryptanalytic attack (Tabu Search and Genetic Algorithm) is used to conduct an intelligent key-search attack on classical ciphers and modern ciphers. The algorithm chosen to represent modern block ciphers is the Advanced Encryption Standard (AES) algorithm. AES is an algebraic product cipher which combines elements of substitution and transposition. Therefore, the primarily aims of this paper is to study the effects of an optimization heuristic cryptanalytic attack on block cipher.**

***Keywords*—Cryptographic Ciphers, Cryptanalysis, and Heuristic Search.**

I. INTRODUCTION

IN today's K-Economy where knowledge means power, cryptology is an integral part of the study of securing information and preventing confidential data from falling into the wrong hands. There are two main types of cryptographic algorithms: symmetric-key and asymmetric algorithms. Symmetric-key algorithms can be divided into two categories: block ciphers and stream ciphers. Figure 1 illustrates the different classifications of Cryptographic ciphers.

This study is aimed at examining the application of a hybrid optimization heuristic cryptanalytic attack on weakly chosen keys in block ciphers; namely classical ciphers and modern ciphers. A block cipher is a symmetric-key cryptographic cipher which uses the same key to encrypt as well as decrypt fixed blocks of a secret message. There are two categories of classical ciphers: substitution ciphers and transposition ciphers. Most modern ciphers are product ciphers, which are extensions of classical ciphers. A product cipher is a block cipher which is an amalgam of components made of substitution and transposition ciphers [1].

The Advanced Encryption Standard (AES) [3] is chosen to represent the group of modern ciphers because it is a relatively new product cipher. Among all the transposition ciphers, the Columnar Transposition Cipher which is most similar to the ShiftRows step of the AES algorithm will be studied as well. Since most modern ciphers combine polygraphic substitution ciphers with a transposition cipher, the Hill Cipher is chosen to represent the substitution ciphers in this study. Furthermore, the Hill Cipher is also quite similar to the SubBytes step of the AES algorithm.

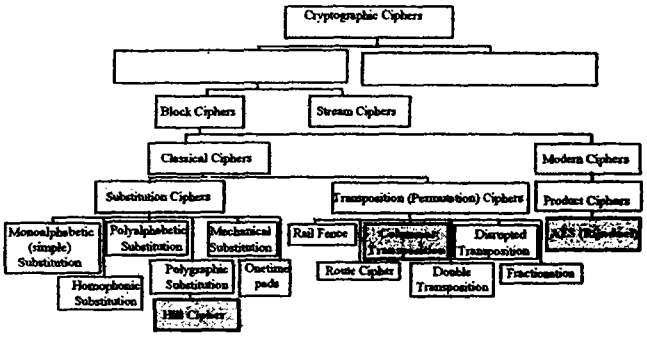


Fig.1. Schematic representation of cryptographic cipher classification (adapted from [1, 2]).

II. RELATED WORK

A The Substitution Cipher (The Polygraphic Hill Cipher)

A substitution cipher maintains the original position of a plaintext character in the ciphertext but substitutes the value of a plaintext with another value [1]. A polygraphic Cipher substitutes blocks of characters in groups; usually pairs of characters known as bigrams. The Hill cipher is a polygraphic substitution which combines and substitutes groups of letters in a block matrix using linear algebra. According to Stallings [4], the frequency distribution of bigrams is more evenly spread in ciphers like the Hill Cipher as compared to the frequency distribution of individual letters in a monoalphabetic cipher. This makes polygraphic more difficult to break the ciphertext. It is difficult to break the Hill Cipher based on known-ciphertext only. However the linearity of the Hill Cipher makes it vulnerable to known-plaintext attacks. Hence, it is usually combined with a permutation (transposition) component as found in Modern Ciphers like Feistel Ciphers.

B. The Transposition Cipher (The Columnar Transposition Cipher)

The transposition cipher rearranges the positions of the plaintext characters in a different and complex order but "leaves the value of a character or character string unaltered

This work was supported by Universiti Sains Malaysia.

when transforming plaintext into Ciphertext" [1]. The Columnar Transposition Cipher arranges the plaintext in a square matrix from left to right and from top to bottom. It depends on the key to determine the number of columns for the letters in the square. Each character in the key becomes a column header followed by the plaintext message in successive rows beneath those headers. Spaces are ignored or replaced with a "null" value. Finally, the encrypted message is written in groups according to columns.

The transposition cipher basically rearranges the content according to a regular pattern. This could be made more complex by additional shuffling the positions of the characters.

C. The Modern Cipher (The AES Algorithm)

AES was designed to overcome the weaknesses and flaws discovered in the design of the Data Encryption Standard (DES). This improved algorithm was meant to be a replacement for DES or triple DES. In addition, the designers of AES claim that the common means of modern cipher cryptanalytic attacks are ineffective against AES due to its design structure. "However, compared to the analysis of DES, the amount of time and the number of cryptographers devoted to analyzing AES are quite limited"[4].

Although AES is an algebraic algorithm with a simple mathematical structure, it does not necessarily mean that it would be easy to break. Up till now, there are only two main directions explored in the cryptanalytic efforts on AES: the algebraic attacks on the S-box and the "Square Attack" on the key schedule.

The main trend of cryptanalytic attacks on AES is based on the Square Attack, which is considered the best known approach to attacking AES so far. The Square Attack and its subsequent variants (the Collision Attack, the Partial Sums Attack and the Related-Key Attack) are unable to break a full version of AES. The attacks only successful in reducing the complexity for about 60 – 70% of the number of rounds required for a complete AES algorithm [8]. The designers of AES have foreseen the possibility of an attack on a few rounds of AES using the Square Attack and set a very high minimum limit for the number of encryption rounds required for each key length to safeguard the security of the algorithm. The best results obtained so far are for 7 out of 10 rounds for 128-bit keys, 8 out of 12 rounds for 192-bit keys and 9 out of 14 rounds for 256-bit keys [8]. This mysterious factor (which limits the number of rounds applicable to the attacks – about 60-70%) is holding the security of AES at the moment.

So far, all the cryptanalytic attacks surveyed are impractical and insufficient to reduce the complexity of an attack on a full version of AES. Most of the attacks focus on the key or the key schedule (with the exception of the XL and XSL attacks which focus on the S-box). Although the designers of AES have made sure that an exhaustive key search on AES would be impractical, the results from the variants of Square attacks show that the complexity is significantly decreased (244 as compared to 272 for a 6-round attack on all key lengths) if combined with an intelligent key search attack.

D. Optimization Heuristic Attacks

Some intelligent cryptanalytic brute-force attacks have been conducted on cipher keys using artificial intelligent methods like simulated annealing, Genetic Algorithm and Tabu Search [1, 11-16]. So far, these search algorithms have only been attempted on classical ciphers like substitution ciphers [1, 13, 15] and transposition ciphers [1, 16] as separate entities. Nevertheless, there has yet to be an attempt to apply these algorithms to a product cipher. A modern cipher is a product cipher which is a combination of both the substitution and transposition cipher. Therefore, the results would be different because of the combination of dispersion and confusion factors involved. However, based on the statistics observed [1, 11-16], there is a good chance that there would be a general improvement in terms of search complexity as compared to an exhaustive-key search if these algorithms were applied as intelligent key-search.

According to [6], Genetic Algorithm and Tabu Search out-performed simulated annealing with positive results. The results presented in [16] also show that Genetic Algorithm and Tabu Search perform better against transposition ciphers (although the authors claim that simulated annealing is more powerful). Hence, this experiment will be conducted using a new optimization heuristic approach. The Genetic Algorithm introduces diversity into the solution pool whereas the Tabu Search prevents the same solution from being re-evaluated too soon.

Genetic Algorithms were first introduced by Holland [17] to solve problems based on the evolutionary process of gene reproduction. Figure 2 shows a general overview of the algorithm which is adapted from its biological counterpart. The Genetic Algorithm begins with a pool of pre-computed solutions (gene pool). Two solutions (parent chromosomes) with the best fitness are selected from the pool to go through the reproduction process, where specific alleles in both parents are swapped randomly to produce two new children with a combination of genes from both parents. Each child is then evaluated to determine its fitness value. The fittest child is selected for the next phase known as mutation. During the mutation phase, specific locations (loci) in the chosen individual are replaced with randomly chosen values to produce an individual with a better fitness value. The new individual is returned to the solution pool and the cycle repeats itself for the successive generations.

The Tabu Search [18] algorithm maintains several Tabu Lists to represents taboo moves in short-term and long-term memory. This algorithm is usually problem-specific. An initial solution is generated and updated with a better solution at each consecutive iteration. Long-term Tabu Lists store frequency values while short-term Tabu Lists store regency values. Aspiration criteria allow taboo moves to be executed if the overall solution is an improvement.

The structure of an English language word consists of unigrams, bigrams and trigrams. Studies have been done to determine the probability of occurrence for characters in the English language [19]. The study reveals the general order of frequency for the occurrence of each character and common bigrams and trigrams. These frequency statistics can be used to determine the probability of occurrence for

each unigram, bigram and trigram in a potential password key. Generally, vowels are the most frequently used character in the English language. A heuristic function could be created to piece together some of these elements to form a word, which could ultimately be the correct password used to form the possible key solution.

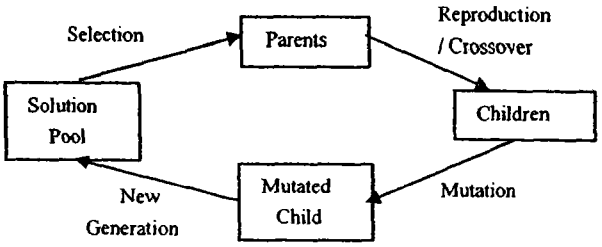


Fig. 2. The Evolutionary Process.

III. Proposed Design Methodology and Framework

A. Overall Framework of the Proposed Solution

A known plaintext will be encrypted by the chosen cipher using a randomly chosen key of reduced length. The possible key-solution generated by the heuristic function will be used to decrypt the known-ciphertext. The resulting plaintext is compared to the original. The fitness value for the solution is obtained by decrypting the known-ciphertext and calculating the percentage of character-location matches in the original plaintext and the decrypted ciphertext. The intelligent search for the correct key combination will continue until a solution match has been found or the closest match has been found within the constraints of the test environment.

For uniformity, a general structure of the proposed methodology was applied on the Hill Cipher, the Columnar Transposition Cipher and the AES. Following sections briefly illustrate a general outline of the proposed methodology for the Tabu Search Algorithm and the Genetic Algorithm. Each series of tests will consist of three trial runs of the full test cycle (one full round) to obtain the average search results of that particular test series.

In order to observe the unique properties and to allow unbiased comparisons between the three different types of cipher algorithms, a uniform structure and environment was used to conduct the tests. The following criteria of the cipher algorithms were adjusted to prepare a suitable uniform environment for testing in the limited time frame given:

1) A uniform intelligent known plaintext-known ciphertext key-search attack using Tabu Search and Genetic Algorithm was conducted on all three types of cipher algorithms.
2) The continuous tests were conducted on Pentium IV 1.50 GHz Computer with 256MB RAM running on a Linux C platform.
3) Only character-location matches will be considered. Upper hex matches and lower hex matches will not be considered for uniformity among cipher algorithms.
4) The plaintext message used for testing is limited to a standard of 16 bytes (128 bit).
5) The encryption and decryption key will be limited to a fixed maximum 8-byte English dictionary word.
6) The symmetric key will only contain English syllables and common dictionary words.
7) Only ASCII characters will be considered. This will reduce the complexity of the attack to a maximum of 56^{16} encryptions for an exhaustive key search. (This would take a maximum of approximately 2.97×10^{8} years of brute-force attack provided 1 million encryptions are done every microsecond).
8) The Tabu Search Algorithm will search randomly for possible key solutions from a pool of known words in the English language. The length of these keywords can be from 1 character to a maximum of 8 characters.
9) For the Tabu Search test run, an assumption is made that the plaintext message is encrypted with a commonly known weak password included in the pool of passwords.
10) For the purpose of comparison, the Tabu Search test will be conducted on two separate pools on different occasions. The first pool contains 2,275 common passwords (8 characters or less) in upper case, lower case and title case. The second pool contains 72,504 common dictionary words (8 characters or less) in lower case.
11) The Genetic Algorithm will be constrained to search randomly for possible 8-character key solutions from a pool of known syllables in the English language. This pool consists of 27 unigrams, 30 bigrams and 12 trigrams.
12) For the purpose of comparison and uniformity with the Tabu Search test, the Genetic Algorithm test will be conducted on two separate pools on different occasions even though the pool size has no bearing on the ultimate results. The first pool contains 320 common passwords (exactly 8 characters) in upper case, lower case and title case. The second pool contains 27,020 common dictionary words (exactly 8 characters) in lower case.

B. Proposed Tabu Search Algorithm Framework

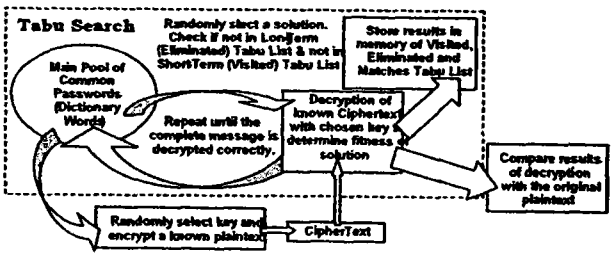


Fig. 3. One Full Test Round of *Tabu Search Algorithm*

For the purpose of uniformity, consider the encryption and decryption process as a black box. A description of a series of test rounds is as follows (summarized by Figure 3):

1) Run steps 2-5 for the Hill Cipher, the Columnar Transposition Cipher and the AES Cipher.
2) Initialize two Long-term memory Tabu Lists: "Eliminated" and "Matches". Initialize one short-term memory Tabu List "Visited" of length n/2, where n = number of solutions attempted from the solution space.

Randomly select an encryption key from the main solution pool and encrypt the known plaintext message.

3) Randomly select a solution (keyword) from the main solution pool and evaluate the fitness of the solution. Calculate the fitness value for the solution by decrypting the known-ciphertext and calculating the percentage of character-location matches in the original plaintext with the decrypted ciphertext. Store the fitness value of the current solution. If the fitness value is zero, store the solution in the "Eliminated" Long-term memory Tabu List. If the fitness value is > 0, store the matched character-location value in the "Matches" List and store the solution in the "Visited" Short-term Tabu List.
4) Repeat step 3 and compare the fitness value of the new solution with the old solution. Repeat steps 3-4 until an exact match has been found. Identify the total number of decryptions required to decrypt the full message correctly.
5) Repeat steps 2-4 twice to produce a test series of 3 test rounds. Obtain the average number of search keys required to decrypt the full message correctly.

C. *Proposed Genetic Algorithm Framework*

For the purpose of uniformity, consider the encryption and decryption process as a black box. A description of a series of test rounds is as follows (summarized by Figure 4):

1) Run steps 2-11 for the Hill Cipher, the Columnar Transposition Cipher and the AES Cipher.
2) Randomly select an encryption key from the main solution pool and encrypt the known plaintext message.
3) Create a new solution pool from the pool of common syllables and calculate the fitness value for all the solutions in the new solution pool by decrypting the known-ciphertext with each solution key by calculating the percentage of character-location matches in the plaintext and the decrypted ciphertext.

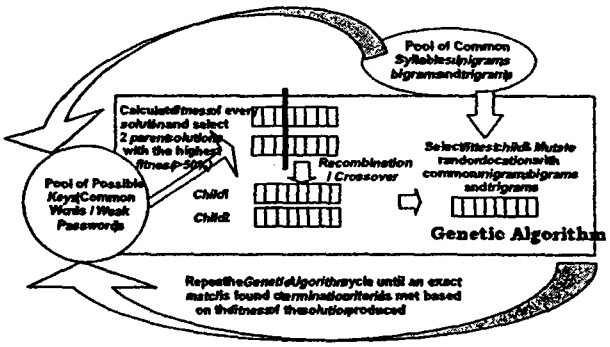


Fig. 4. One Full Test Round of Genetic Algorithm

4) Choose two solutions with the best fitness value. Each solution should minimally be able to recover at least 50% of the original plaintext message.
5) Randomly select a "crossover" point and swap the contents between the two solution key arrays.
6) Evaluate the fitness for each new child (solution key) by decrypting the known-ciphertext with each "child" key and calculating the percentage of character-location matches in the plaintext and the decrypted ciphertext.
7) Choose the "child" solution with the highest fitness value.
8) Randomly select locations and mutate the selected locations with arbitrarily chosen unigrams, bigrams and trigrams from the pool of common syllables.
9) Evaluate the fitness of the solution. If the fitness value is better than the current fitness value, update the current fitness value and the best solution variables.
10) Repeat steps 8-9 until the fitness value is 100% or there is no change in best fitness for a predetermined number of iterations.
11) Repeat steps 2-10 twice to produce a test series of 3 test rounds. Obtain the average number of search keys required to decrypt the full message correctly.

IV. Implementation and Results

A. *Implementation Problems*

The total amount of time needed to get the results for intelligent key-search attack depends on three major factors: the probability of random selection, the weakness of the keyword chosen and the strength of the cipher structure against a heuristic attack. The Genetic Algorithm proved to be most efficient on transposition cipher (the Columnar Transposition Cipher). However, it was also observed that the Genetic Algorithm produced weak results for the substitution cipher (the Hill Cipher) and the modern cipher (the AES product cipher).

After many trial runs, it was discovered that the processing power of the test environment was insufficient to completely recover the full plaintext message from these two ciphers (the Hill Cipher and the AES product cipher). Nevertheless, the attacks were successfully conducted on the full-cycle versions of all the cipher algorithms to produce measurable results.

B. *General Findings: Results of Implementation of Proposed* Tabu Search *Algorithm Framework*

Figure 5 and Figure 6 summarize the results obtained from 21 test runs (seven series) of the Tabu Search algorithm on the three types of ciphers (AES, Hill and Columnar Transposition). Result from Figure 5 was based on a pool of 2,275 possible keywords and result for Figure 6 was based on a pool of 72,504 possible keywords. Figure 7 to Figure 9 shows the effectiveness of Tabu Search on the ciphers based on the pool size comparison.

C. *General Findings: Results of Implementation of Proposed* Genetic Algorithm *Framework*

For uniformity, the Genetic Algorithm is tested using an encryption key from two pools. However, the pools of encryption keys only contain keywords which are exactly 8 characters long. The sizes of the two pools are 320 keys and 27,020 keys respectively. Overall, the Genetic Algorithm produced results from the Columnar Transposition Cipher fast and efficiently. In fact, the performance against this cipher was better than the Tabu Search. However, the Genetic Algorithm generally did not perform well on the other two ciphers, namely the Hill Cipher and AES. In most of the cases, the Genetic Algorithm could not produce any significant positive result from these two ciphers at all. After one month of continuous test runs, it was discovered that these two ciphers have a consistent pattern: One test run

cycle can last up till 8-12 hours before the computer fails and crashes in the midst of building the initial solution pool. Consequently, an important point to note is that when attempts were made to use the Genetic Algorithm on the Hill Cipher or the AES Cipher, the process almost never goes beyond the first step of initializing the solution pool and obtaining two parent key solutions with a minimum fitness of 50% or more.

Figure 10 summarizes the average results of conducting 10 series of test runs (total of 30 test runs) of the Genetic Algorithm on the Columnar Transposition Cipher.

Figure 11 illustrates a pattern, showing the relationship between the total numbers of generations of key solutions required to be tested before an optimal solution is found vs. the initial pool size required to obtain two parents with a minimum fitness of 50%.

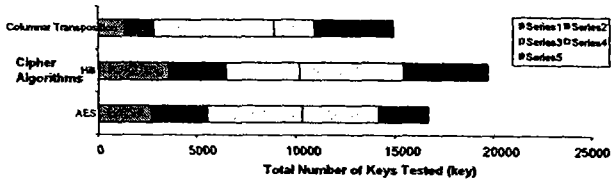


Fig. 5. Comparison of Effectiveness of Tabu Search on Cipher Algorithms: 15 rounds of Tabu Search (Tabulation Based on Total Search Keys Required in Pool Size of 2,275 words)

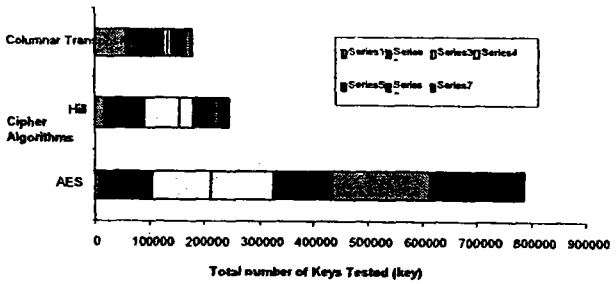


Fig. 6. Comparison of Effectiveness of Tabu Search on Cipher Algorithms: 21 Test Rounds of Tabu Search (Tabulation Based on Total Search Keys Required in Pool Size of 72,504 words)

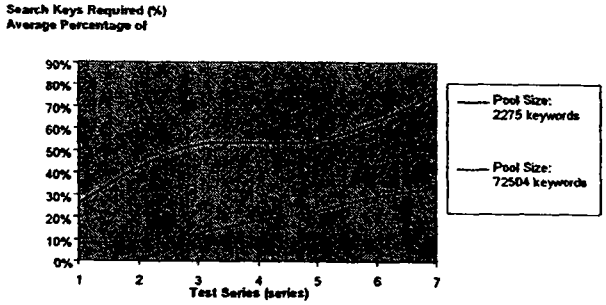


Fig. 7. Effectiveness of Tabu Search on Hill Cipher: A Comparison Based on the Keyword Pool Size

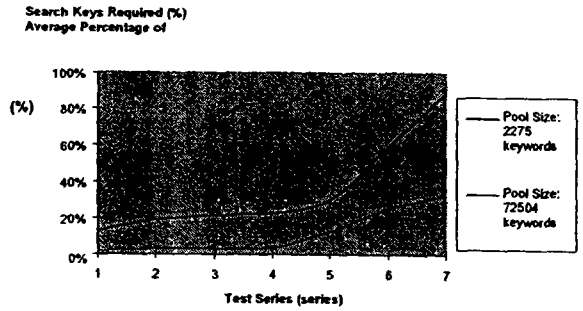


Fig. 8. Effectiveness of Tabu Search on Columnar Transposition Cipher: A Comparison Based on the Keyword Pool Size

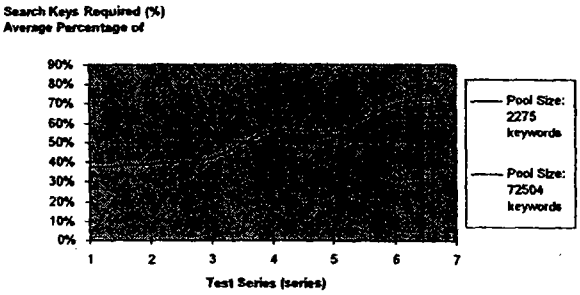


Fig. 9. Effectiveness of Tabu Search on AES Cipher : A Comparison Based on the Keyword Pool Size

D. *Discussion of Results*

Proposed Tabu Search Algorithm Framework

A trend was observed from the results of this research that regardless of the strength of the cipher algorithm, the performance of the Tabu Search attack is generally improved if the attacker uses a larger pool of known potential weak password. However, contrary to the characteristics of the two classical ciphers the security of the AES cipher proved to be relatively stable and did not vary too much with the change of Tabu Search keyword pool size. Although there is a very slight improvement in the performance of the Tabu Search attack on the AES Cipher with the increase of the potential keyword solution pool size, the changes are very minor and almost negligible.

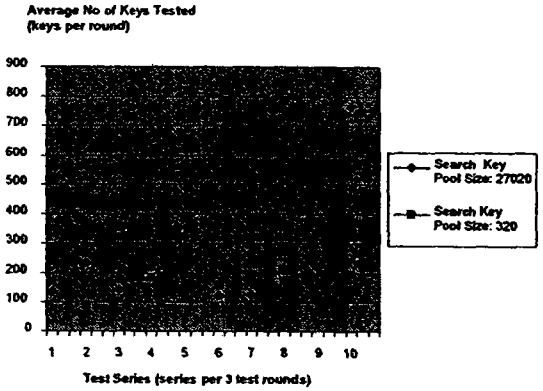


Fig. 10. Effectiveness of Key Search Using Genetic Algorithm on Columnar Transposition Cipher (10 series – 30 test rounds)

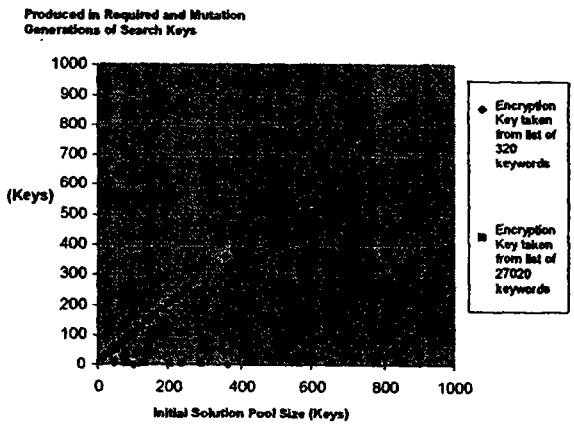


Fig. 11. Genetic Algorithm on Columnar Transposition Cipher: Relationship between the Total Generations of Solutions Required and the Size of the Initial Solution Pool Generated

TABLE 2
SUMMARIZED AVERAGE RESULTS FOR THE APPLICATION OF *GENETIC ALGORITHM* ON THE *AES CIPHER*

	Fitness (Percentage of Plaintext Recovered)	Initial Pool Size (keys)	Generations (keys)	Estimated Average Time Span to Obtain Best Parent Solution (Hours)
Best Case	25.0%	18580088	0	4.5
Average	12.5%	385592	0	2.5

This is surprising considering the AES cipher is a product cipher which should contain the properties of both the substitution and the transposition ciphers. It appears as if the product cipher has inherited more of the strengths of both types of classical ciphers but very little of the weaknesses. However, this proves that although the strength of the AES product cipher is affected by the strength of the key to a certain degree, the cipher's security is relatively stable because it does not fully depend on the security of the key alone.

Proposed Genetic Algorithm Framework

Generally, the Genetic Algorithm attack proved to be most efficient against the transposition cipher. The attack succeeded in recovering the original plaintext message in less than an hour for each trial run. Nonetheless, it was also observed that the Genetic Algorithm attack produced weak results for the substitution cipher (the Hill Cipher) and the modern cipher (the AES product cipher).

This is due to the fact that the original plaintext message may be recovered by using an alternative key with similar properties as the original encryption key on the transposition cipher, but never on the substitution cipher or on the product cipher. This is because of the confusion property inherent in both the substitution cipher and the product cipher. Generally, the results suggest that a parallel implementation of the Genetic Algorithm would produce better results than the serial implementation done here.

V. CONCLUSION

The results have shown that the transposition cipher (Columnar Transposition Cipher) is most susceptible to the Tabu Search and Genetic Algorithm attacks on weak passwords. This is followed by the Polygraphic Substitution Cipher (Hill Cipher), which is also vulnerable to the Tabu Search attack as well as the Genetic Algorithm attack, but at a greater time cost (provided the encryption key is a weakly chosen password). The product cipher (the AES cipher) is the most secure among the three. Unlike the other two, the product cipher is rather stable in terms of its vulnerability towards the optimization heuristic attacks. Nevertheless, the product cipher is still susceptible to weak password attacks by the average hacker or script kiddie using a basic personal computer system. This is especially obvious from the average 52% - 53% key search efficiency using the Tabu Search algorithm. In short, regardless of the strength and security of a cryptographic cipher, all categories of cipher algorithms are vulnerable to optimization heuristic attacks by a basic personal computer if the encryption key is a weakly chosen password.

REFERENCES

[1] Gründlingh, Werner R. and Van Vuuren, Jan H. *Using genetic Algorithms to Break a Simple Cryptographic Cipher.* http://www.apprendre-en-ligne.net/bibliotheque/genetic.ps

[2] Schneier, Bruce. 1996. *Applied Cryptography - Protocols, Algorithms, and Source Code in C.* Second Edition. Canada: John Wiley & Sons.

[3] National Institute of Standards and Technology, U.S. department of Commerce. November 26, 2001. *Advanced Encryption.*

[4] Stallings, William. 2003. *Cryptography and Network Security Principles and Practices.* Third Edition. New Jersey: Prentice Hall. pp.29,37-40,653.

[5] *Standard(AES).* FIPS PUB 197. http://csrc.nist.gov/publications

[6] Courtois, N.T. and Pierprzy, J. Dec 2002. *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations.* Asiacrypt 2002.

[7] Moh, T. September 18, 2002. *On the Courtois-Pieprzk's Attack on Rijndael.* http://www.usdsi.com/aes.html

[8] Murphy, S. and Robshaw, M.J.B. 2002. *Essential Algebraic Structure within the AES.* Crypto 2002. http://www.isg.rhul.ac.uk/~mrobshaw/rijndael/aes-crypto.pdf

[9] Ferguson, Niels; Kelsey, John; Lucks, Stefan; Schneier, Bruce; Stay, Mike; Wagner, David and Whiting, Doug. 2000. *Improved Cryptanalysis of Rijndael.* Springer-Verlag. http://www.macfergus.com/pub/ icrijndael.pdf

[10] Babbage, Steve. November 11, 2002. *Rijndael and other block ciphers.* NESSIE Discussion Forum. http://www.cosic.esat.kuleuven.ac.be/nessie/forum/read.php?f=1&i=82&t=82

[11] Dlanielyan, Edgar. February 2001.*AES: Advanced Encryption Standard is Coming.* ; login:, the magazine of USENIX and SAGE 26(1): 62.

[12] Kolodziejczyk, Joanna. 1997. *The Application of Genetic Algorithm in Cryptanalysis of Knapsack Cipher.* Proceeding of European School on Genetic Algorithms. Eurogen '97. http://ingenet.ulpgc.es/functional/eurogenxx/ eurogen97/contributed/kolodziejczyk/ht/kolodziejczyk.htm

[13] Spillman, Richard. October 1993. *Cryptanalysis of Knapsack Ciphers using Genetic Algorithms.* Cryptologia XVII (4). pp. 367-377. http://www.plu.edu/~janssema/ga_solve.zip

[14] Clark, Andrew and Dawson, Ed. 1998. *Optimisation Heuristics for the Automated Cryptanalysis of Classical Ciphers.* Journal of Combinatorial Mathematics & Combinational Computing. Vol 28. pp.63-86. http://sky.fit.qut.edu.au/~clarka/papers/jcmcc1998.pdf

[15] Lebedko, O. and Topchy, A. 1998. *On Efficiency of Cryptanalysis for Knapsack Ciphers.* Poster Preoceddings of ACDM'98 PEDC. http://www.msu.edu/~topchyal/acdm98.ps

[16] Dimovski, A. and Gligoroski, D. October 2003. *Attacks on the Transposition Ciphers UsingOptimization Heuristics.* Proceedings of ICEST 2003. http://www.pmf.ukim.edu.mk/~danilo/ResearchPaper/Crypto/ AttackTranspositionICEST2003.pdf

[17] Dimovski, A. and Gligoroski, D. March 2003. *Attack On the Polyalphabetic Substitution Cipher Using Genetic Algorithm.* Technical Report, Swiss-Macedonian scientific cooperation trought SCOPES project. http://www.pmf.ukim.edu.mk/~danilo/ResearchPapers/Crypto/ AttackPolyalphabeticSCOPES2003.pdf

[18] Holland, J. 1975. *Adaptation in Natural and Artificial Systems.* Ann Arbor, Michigan: University of Michigan Press.

[19] Glover, Fred; Taillard, Eric; and de Werra, Dominique. 1993. *A user's guide to tabu search.* Annals of Operations Research, 41. pp. 3-28.

[20] Crypto'04. 2004. *Most Common Letters, Digrams, and Trigrams in the English Language.* http://academic.regis.edu/jsejbert/Crypto04/Frequency.pdf