Journal of Engineering Sciences, Assiut University, Vol. 38, No. 3, pp.749-761, May 2010.

EFFICIENT PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH

B. MORGAN[1], M. HAMADA[2], AND G. ABDEFADEL[3]

[1] *Bassant.Morgan@live.com*

[2] *Assistance Professor, Communication and Electronics Department, Helwan University*

[3] *Professor, Communication and Electronics Department, Helwan University*

(Received November 12, 2009 Accepted March 9, 2010).

Public key encryption with key word search (PEKS) enables user Alice to send a secret key to a server that will enable the server to locate all encrypted messages containing the keyword, but learn nothing else. In this paper, we propose a new scheme the Efficient Public Key Encryption with Keyword Search (EPEKS). Firstly we explained the construction of new scheme (EPEKS). (EPEKS) doesn't base on Identity Based Encryption or pairing which was used in the construction of the (PEKS) that proposed in Boneh's paper and other papers; it is based on Public Key Cryptosystem. Secondly, we proposed provably secure (EPEKS) scheme that refreshes keywords and processes multiple keywords. Finally, we described the security of the new scheme by using Cryptographic algorithm and Hash function.

KEYWORDS*: Public key encryption, RSA, Encryption email, Hash function, Mail server, Refreshing keywords.*

1- INTRODUCTION

1.1 Basic concept

The Efficient Public Key Encryption with Keyword Search (EPEKS) realizes the following scenario. Suppose Alice, who is a manager of a bank, is having a holiday and away from work. She is equipped with a smart phone that can be used to check her important emails, in case there is an urgent email that requires her attention. In this scenario, Alice should be able to select her important emails to be read during her holiday, but not all of them. Due to the importance of her email, all the emails sent to her will be encrypted using her public key. This ensures that nobody else, other than Alice, will be able to retrieve the emails directed to Alice. To enable Alice to select her important emails, she must send a "hashed value" to the server, so that the server can use this information to select the emails that Alice wants to read. For instance, assume that the keyword W is known by both the sender "Bob" and the receiver "Alice", and a variable value r will be created by Bob. Both the keyword W and the variable value r will be conjunct, hashed and sent by Bob. Bob would like to send an email to Alice, he encrypts his email and the variable value r by using Alice's Public Key,and appends the hashed value to the resulting ciphertexts. The ciphertexts and the hashed value will be saved in Alice's mail server. When Alice wants to read any urgent emails, she will send a request to the mail server regarding the new emails. The mail server will reply

by sending the encrypted variable value *r*. Alice will decrypt the variable value *r* by using her private key, conjunct both the known keyword *W* and the decrypted variable value *r*, and hash them to get the hashed value that will be sent to the mail server to get the appropriate email.

In short, EPEKS provides a mechanism that allows Alice to have the email server to extract emails that contains a particular keyword by providing a hashed value corresponding to the keyword, while the email server doesn't learn anything else about the email.

1.2 Related work and our Contributions

There are few papers directly related to PEKS. PEKS was first introduced in Boneh [1], and later improved in Baek [2], Gu [3], and Khader[4]. All of these works depend mainly on Pairing Based Cryptography, and Identity Based Encryption IBE.

In [1] the authors presented general scheme called PEKS where Alice gives trapdoors for the words she wants the gateway to search for. The trapdoors come in the form of some kind of data that is used to test the existence of keywords within an email without revealing any other information. In practice, the system will be used over many rounds. The server which received the trapdoor for a keyword *W* can store the trapdoor and use it to learn all future emails with that category. One can assume that the server cannot memorize trapdoors but this is a very restrictive assumption and not easy to implement in practice. The paper does not specify what happens if the server memorizes the trapdoor information related to the keyword sent by Alice, and the protection against this situation is not discussed. In [1] the authors mentioned that some search will be done by multiple keywords, but they didn't discussed how one can formalize the concept of the multiple keywords search, and create the PEKS cipertexts for multiple keywords. In [2] the authors pointed out two features that were not covered in [1]. The first one was the ability to search for multiple keywords. The second one was the requirement of secure channels, for sending trapdoors. However, in [2] the authors mentioned that there were open problems, such as the design of the PEKS, and the way to find an efficient and convenient way to refresh keywords. In [3] the authors presented PEKS based on pairing, and their paper provided a discussion on removing the secure channels from PEKS, and presented secure channel free PEKS. In [4] the author mentioned that the security of her new scheme was proved by showing that the use of Identity Based Encryption IBE has a notion of key privacy, besides to the modifications which were done to enable multiple keyword search, and remove the need of the secure channels.

In [1-4] the authors mentioned that Public key encryption with keyword search PEKS based on the pairing scheme. Constructing a PEKS is related to Identity Based Encryption IBE, though PEKS seems harder to construct. They showed that PEKS implies Identity Based Encryption, but the converse is currently an open problem.

In this paper, we discuss the following issues:

1- Construct new scheme EPEKS based on public key cryptosystem, instead of the PEKS which based on IBE.
2- Give Alice the ability to search using multiple keywords.
3- Prevent the mail server to memorize the keywords by refreshing its database.
4- Neither secure channel nor pairing has been discussed in this paper.

2. PRELIMINARIES

In this section we will go through some definitions, such as cryptography algorithm and hash function, which will be used further in this document. For more detail see, [7]

2.1 The cryptography algorithm

Public key algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic: a) It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic and the encryption key. In addition, some algorithms, such as RSA also exhibit the following characteristic. b) Either of the two related keys can be used for encryption, with the other used for decryption.

A Public key encryption scheme consists of the following items:

1- Plaintext: This is the readable message or data that is fed into the algorithm as input.
2- Encryption algorithm: The encryption algorithm performs various transformations on the plaintext.
3- Public and Private Key: This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input.
4- Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the key. For a given, two different keys will produce two different ciphertexts.
5- Decryption algorithm: this algorithm accepts the ciphertext and the matching key and produces the original plaintext.

The essential steps are the following:

1- Each user generates a pair of key to be used for the encryption and decryption of messages.
2- Each user places one of the two keys in a public register or other accessible file. This is the public key. The combination key is kept private.
3- If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
4- When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

Let us take a closer look at the essential elements of a public key encryption scheme. See Fig.1. There is some source A that produces a message in plain text. $X = [X_1, X_2, \ldots\ldots, X_M]$. The M elements of X are letters in some finite alphabet. The message is intended for destination B; B generates a related pair of keys: a public key, KU_b, and a private key KR_b, KR_b is known only to B, whereas KU_b is publicly available and therefore accessible by A.

With the message X and the encrypted key KU_b as input, A forms the ciphertext

$$Y = [Y_1, Y_2, \ldots\ldots Y_N]$$
$$Y = \mathrm{E}_{KUb}(X) \quad (1)$$

The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$X = D_{KRb}(Y) \quad (2)$$

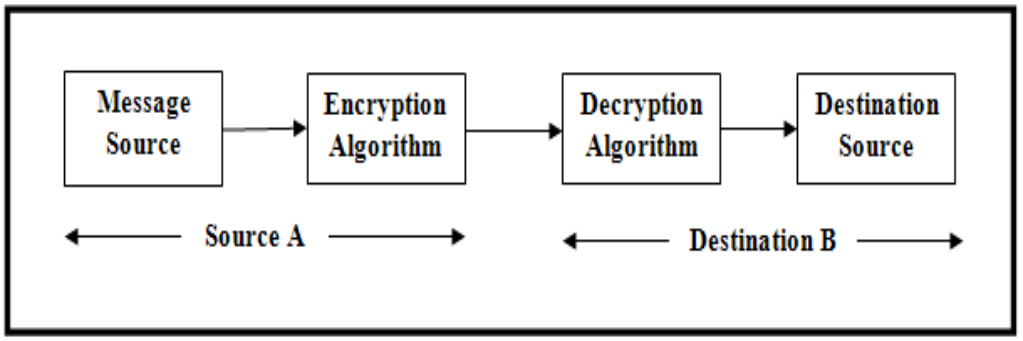

Fig.1 The essential elements of a public key encryption scheme

2.2 Hash Functions

A hash value h is generated by a function H of the form

$$h = H(M), \quad (3)$$

where M is a variable-length message and H (M) is the fixed-length hash value. The hash code is a function of all the bits of message and provides an error-detection capability: A change to any bit or bits in the message results in a change to the hash code. The hash value is appended to the message at the source at a time when a message is assumed or known to be correct. The receiver authenticates that message by recomputing the hash value. See Fig.2

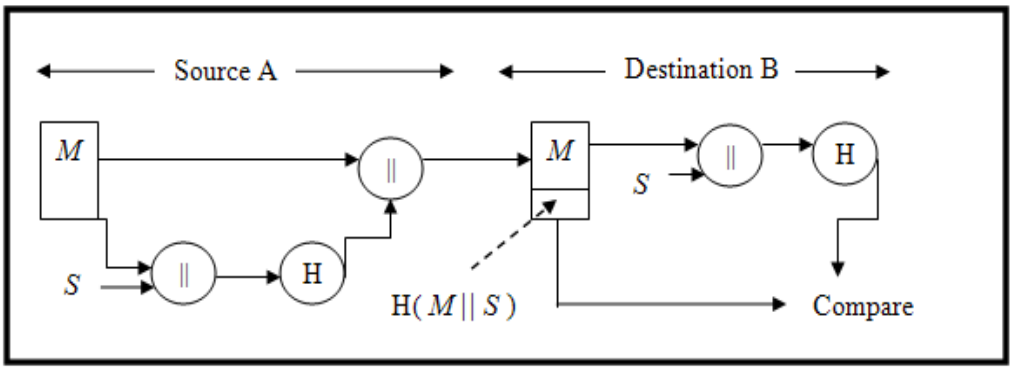

Fig.2 Compute hash code of message plus secret value

The above figure provides authentication as only A and B share S.

3. EFFICIENT PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH

An encrypted email is sent from Bob to Alice. The gateway wants to check whether a certain keyword exists in an email or not. Nevertheless Alice doesn't want the email to

be decrypted by anyone except her, not even the gateway. This is a scenario where efficient public key encryption with keyword search EPEKS is needed.

3.1 Definitions of EPEKS

In our new scheme, three parties called "sender", "receiver", and "server" are involved. The sender “Bob” is a party that creates and sends encrypted message and variable value which we call "ciphertext". The server “mail server” is a party that receives the encrypted message and variable value "ciphertext", stores them in its database, and performs search upon receiving the request “check for new mail” from the receiver. The receiver “Alice” is a party that sends the requests “check for new emails” to the server to get the required data. The below diagram describes the process in a simple steps. See Fig.3.

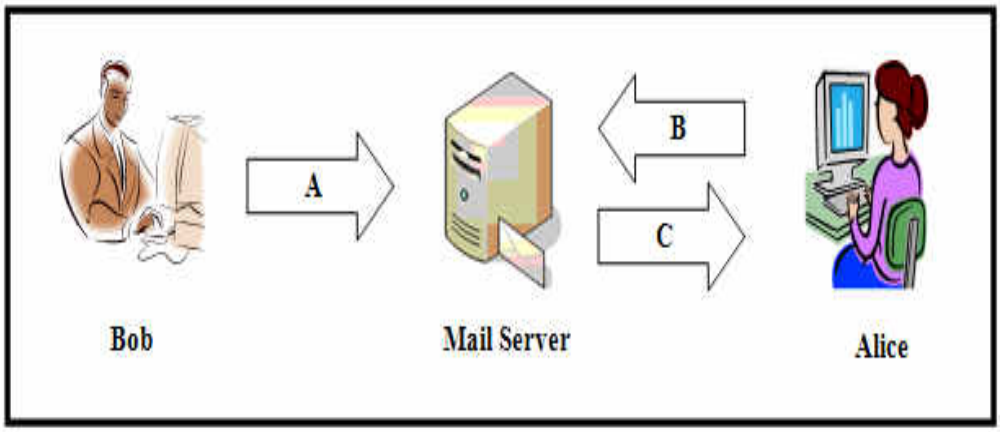

Fig.3 The parties of the new scheme EPEKS

A: The encrypted message “email” is sent by Bob.
B: The request “Check for new Emails” is sent by Alice.
C: The required data “new emails” is sent by the mail server.

3.1.1 The sender party has the following elements:

1- The encrypted message (M).
2- Sender's public key (Alice's public key KU_a)
3-The chosen keyword (W), the keyword is known for both the sender and the receiver.
4- Hash function (H).
5- Variable value (r).

3.1.2 The mail server

Contains a database which consists of the encrypted email $E_{kua}(M)$, the hashed value $H(W||r)$, and the encrypted variable value $E_{kua}(r)$.

3.1.3 The receiver party has the following elements:

1- Receiver's Private Key (Alice's private key KR_a).
2- Hash function (H).
3- The chosen keyword (W).

3.2 Construction of EPEKS

The below section describes the construction of EPEKS by using both RSA as cryptography algorithm, and hash function as authentication function. The below section explain the EPEKS scheme in two stages. The first stage is the encryption process, and the second stage is the decryption process.

3.2.1 The Encryption Process

The encryption is the first stage in our scheme, and it is done by the sender "Bob" under the receiver's "Alice" public key.

A) The Sender Party

Assumptions:

1- The keyword *W* is known by both the sender "Bob", and the receiver "Alice".
2- By using RSA algorithm, public key *KU* and private key *KR* are known by Bob and Alice.
3- The variable secret value *r* is chosen and known by "Bob".

Consider Bob sends an encrypted message to Alice, using her public key KU_a. Let the keyword *W*. This keyword will be added to the variable value *r*. Assume *r* is a number, such as *10*. The variable value *r* plus the keyword will be hashed by the hash function.

It is important to hide *r* from the mail server and from anyone wants to reach Bob's encrypted message, however Alice must know this variable value so as to get Bob's encrypted message.

To solve this problem, Bob encrypts *r* under Alice's public key. Therefore, Alice will be the only one who can decrypt *r* and reaches Bob's encrypted message.

Therefore, the three outputs from the encryption stage are: the encrypted message $E_{kua}(M)$, the encrypted variable value $E_{kua}(r)$, and the hashed value $H(W||10)$. The outputs will act as inputs to Alice's mail server as shown in Fig.4. Note that *r* could be either a number or a word. In this document, *r* has chosen as number in section 4, and 5.

So to send a message with keyword *W*, Bob sends

$x_1 = E_{KUa}[M]$

$x_2 = H[W \| r]$

$x_3 = E_{KUa}[r]$

$X = x_1 \| x_2 \| x_3$

$$X = E_{KUa}[M] \| H[W \| r] \| E_{KUa}[r] \quad (5)$$

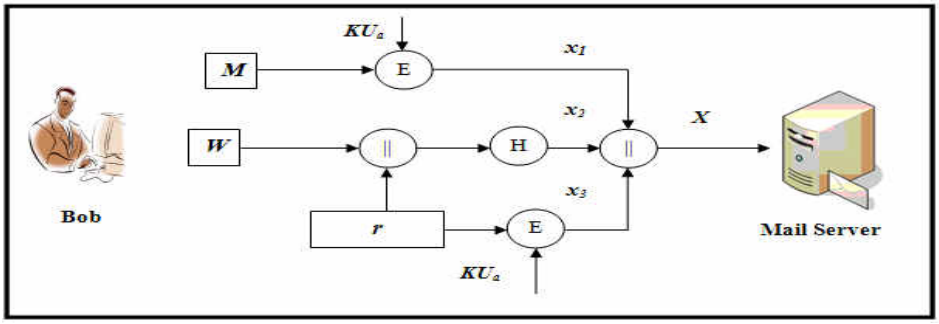

Fig.4 Sender Party

B) Mail Server Party

The mail server receives Eq.5 as input. Assume that the mail server database divided into four columns: sender column, the encrypted message column, the hashed value column, and the encrypted variable column. Each value will be directed and located in its appropriate column (this behavior is done by the mail server itself, and it hasn't been discussed in this document). We assumed in this section that the database has only one data value (one email) related to "Bob" as shown in table 1. In this document we ignored the mail server application type.

The mail server stores these inputs in its database and gets ready to perform search upon receiving the request (check for new emails) from the receiver to send her the encrypted variable value as shown in Figure 5.

Table 1 Mail Server Database

Sender	Encrypted Message	Hashed Values	Encrypted Variable Value
Bob	$E_{KUa}[M]$	$H[W \| r]$	$E_{KUa}[r]$

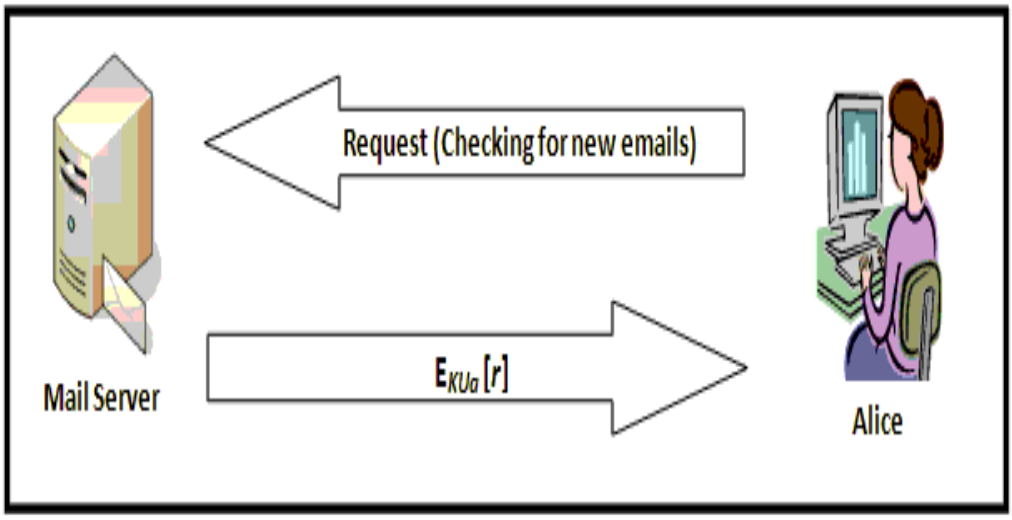

Fig.5 The communication between the Mail Server and the Receiver

3.2.2 The Decryption Process

A) The Receiver Party

Alice sends a request to check for her new emails, the mail server replies by sending the encrypted variable value $E_{KUa}[r]$. Alice decrypts the r by using her private key $D_{KRa}[r]$. She adds the variable value to the known keyword and hashes them by using the hash function to get the hashed value $H[W \| r]$. Alice sends the hashed value to the mail server to be compared with the one which was sent by the "Bob" and stored in the mail server database as shown in Fig.6

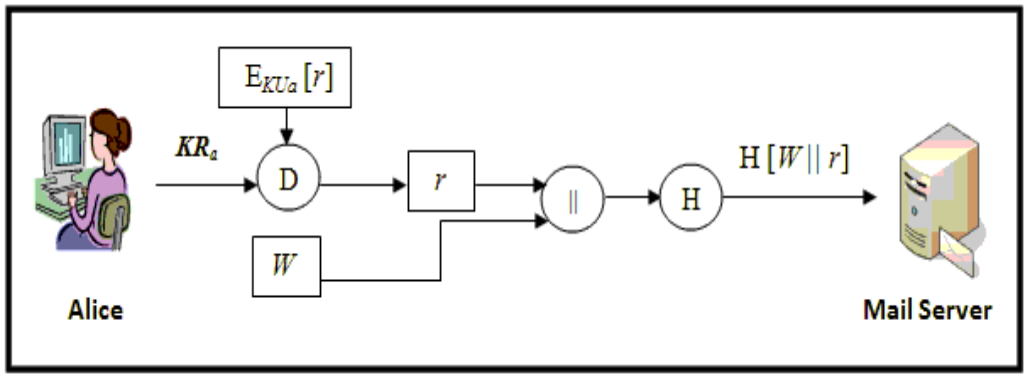

Fig.6 The Decryption Process at the Receiver Party

B) Mail Server Party

The hashed value received by the mail server. The main role for the mail server is searching for any matching in its database regarding the hashed value. If the server found the exact hashed value which Alice asked for, the server would send the encrypted message to Alice, otherwise the mail server would send a message asking Alice to try again as shown in Fig.7.

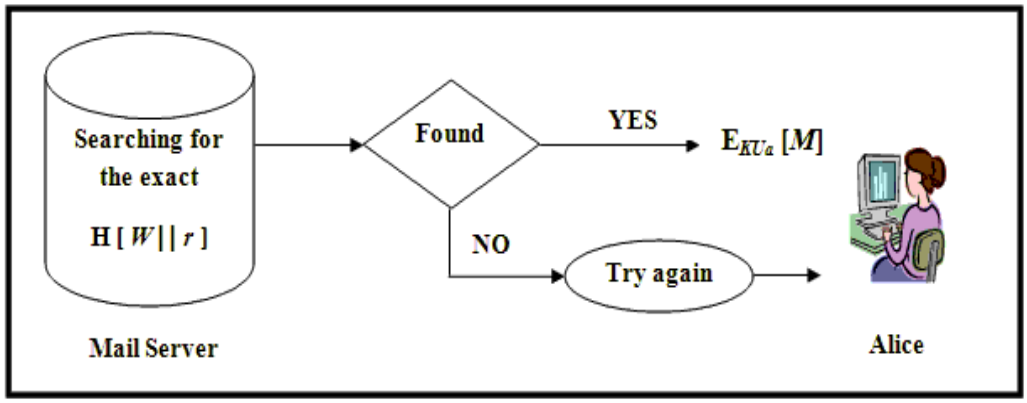

Fig.7 Matching process in the Mail Server

Due to the first assumption that the mail server database contains only one data value (one email), then Alice will receive Bob's email, and won't get "Try again". In section 4, Fig.7 is more instance than this section.

4- REFRESHING KEYWORDS

Bob wants to send Alice a message with keywords. Bob encrypts the message using Alice's public key. He then appends to the resulting ciphertext a list of PEKS ciphertext of each keyword. This kind of encrypted message may be stored in a server. Alice can give the server a certain trapdoor that enables the server to test whether one of the keywords associated with the message is equal to the word of Alice's choice. As

a natural (not artificial) example, assume that there are three words, say "high priority", "normal", and "less priority" that are frequently used in the system. Now assume that whenever Alice sends a trapdoor, the server stores it in its memory. Then, at the same point, the server gets trapdoors for all the keywords used within the system and can decide which PEKS ciphertext encrypts which keyword without receiving trapdoors from Alice. Since the storage capacity of the computers are increasing rapidly these days, even if more keywords are used, say 100, the server can store all trapdoors generated by the receiver and conduct search by itself.

In [1], the paper didn't specify what happens if the server memorizes the trapdoor information related to the keyword sent by Alice, and the protection against this situation is not discussed. In [2] the authors mentioned a solution is to refresh the frequently-used keywords by attaching time period information to them. For example, a keyword $W = Urgent$ now becomes $W' = Urgent \| 01/07/04$, where 01/07/04 denotes "1 July 2004". In the above example, once the receiver releases a trapdoor, the server can search PEKS ciphertexts that correspond to W' without receiving a trapdoor for it until the end of the "day". In our point of view, the above solution is not efficient and convenient way to refresh frequently-used keywords.

Imagine that the attacker is the administrator of the mail server which stores all trapdoors in it. A "day" is a convenient duration for the administrator to keep trying to get the trapdoors.

Nevertheless, one possible solution for the above problem is to refresh keywords by adding a variable secret value r to the keyword, and hashed them by using hash function, besides each message will has its own variable secret value. For instance, find the below examples.

Example 1: Bob would like to send Alice a message M

a. Assume Bob and Alice have chosen a keyword $W = Urgent$.
b. Bob encrypts M under Alice's public key $E_{KUa}[M]$.
c. Bob chooses and encrypts variable secret value r under Alice's public key $E_{KUa}[r]$.
d. r and W will be hashed by hash function at Bob's end $H[W \| r]$.
e. Assume that the attacker is the administrator of the mail server.

The three values $E_{KUa}[M]$, $E_{KUa}[r]$, and $H[W \| r]$ will be sent from Bob to the mail server and saved into its database as mentioned before in table 1.

Formally, we define security against an active attacker using the following game between EPEKS and the attacker:

1- If the administrator would like to get r, he needs either Alice's private key or he needs to break the hashed value.
2- If the administrator would like to get M, he needs Alice's private key.
3- If the administrator would like to get W, he needs to break the hashed value.

M, W, r, and the hashed value are unknown values for the administrator. The administrator will not be able to decrypt r because he doesn't know Alice's private key, besides he doesn't have the keyword to be added to r to get the hashed value.

Therefore, the administrator will learn nothing about M, W, r, and the hashed value. Even if he reach one of them, it will be difficult to get the rest.

From the above, we can get that the security of EPEKS system depends on number of variables. It is too difficult for any attacker to get all the variables at the same time to reach the encrypted message. One can decide that there is no need for the

refreshing keywords process, because all the variables are unknown. Despite of the unknown variables, we would like to get the highest level in security by refreshing the keywords.

Refreshing keywords has been proved in this document through the below example.

Example 2: Bob would like to send Alice two messages M_1,M_2 by using the same keyword.(The below is the second game between EPEKS and the attacker).

If we assumed that W = Urgent , and $r = 10$, Bob will send the message normally as shown in example 1, but if Bob decided to use the same keyword W in the second message, he will create a new r and this is the trick. Therefore the second hashed value will be different from the previous one which was mentioned in example 1. (Regarding the differences in the hash codes, return back to the definition of the Hash Function Section 2.2).

a. Let $W = Urgent$, and W is known by Bob and Alice.
b. Bob encrypts M under Alice's public key $E_{KUa}[M]$.
c. Assume each M has its own r. [$r_1 = 10$ and $r_2 = 20$ for M_1 and M_2 respectively].
d. Bob chooses and encrypts two variable secret values r_1, and r_2 under Alice's public key $E_{KUa}[r_1]$, and $E_{KUa}[r_2]$.
e. r_1, and r_2 will be added to W and hashed by hash function at Bob's end. H_1 [*Urgent* ||*10*], and H_2 [*Urgent* ||*20*].
f. Assume that the attacker is the administrator of the mail server.

Bob sent messages to the same receiver Alice, using the same public key, using the same hash function, and using the same keyword in both messages. If the administrator would like to get to W, it will be impossible because he doesn't know either r_1 or r_2 to reach the hashed values. In case if he gets either r_1 or r_2, still W is unknown to get the hashed value. Due to H_1 [*Urgent* ||*10*] is not equal to H_2 [*Urgent* ||*20*], it will be difficult for the administrator to reach the encrypted messages.

Based on the above, Bob can use W as a keyword several time without effecting the security of the EPEKS scheme. Even if the mail server has the ability to store large number of hashed values, it won't be able to memorize the hashed value because they are not equal to each other due to the variable r. hence the security method of EPEKS scheme is easy to implement, and difficult to break.

5- HANDLING MULTIPLE KEYWORDS

Multiple Keyword search in the EPEKS is the capability of searching for more than one word in the mail server database. In [4], the author mentioned that multiple keyword search in a PEKS is the capability of searching for more than one word either disjunctively or conjunctively. She continued that in [1] the only way to do this is to search for each word separately and then do the disjunctive or conjunctive operations on the result testing algorithm. In her point of view, this technique is impractical when it comes to a large number of keywords on one conjunctive search request, because every email is searched for every single keyword. She suggested a new scheme for conjunctive search called PECK. This scheme substitutes the PEKS algorithm with PECK algorithm that encrypts a query of keywords. The testing is done with a trapdoor for each query instead of each word. She said that the scheme is secure against a

chosen keyword search attack (CKA) if an adversary has a low advantage in guessing the right query of keywords being encrypted.

We presented the above opinion for the related works regarding multiple keyword search, however, in this document we proposed different mechanism which is not related to the previous works. The below example explains the multiple keyword search process.

Example 3: Bob would like to send Alice a message M with two keywords W_1, and W_2

a. Assume Bob and Alice have chosen two keywords W_1 = *Urgent* W_2 = *Important.*
b. Bob encrypts M under Alice's public key $E_{KUa}[M]$.
c. Bob chooses and encrypts variable secret value (r) under Alice's public key $E_{KUa}[r]$.[assume $r = 10$]
d. Each (W) will be added to (r) and hashed by the hash function at Bob's end. H [$W_1 \| r_1$], and H [$W_2 \| r_1$], then the hashed values are H [*Urgent* ||*10*], and H [*Important* ||*10*].

$E_{KUa}[M]$, H [*Urgent* ||*10*], H [*Important* ||*10*], and $E_{KUa}[10]$ will be saved in the mail server database. Alice will send a request asking for new emails. The mail server will reply by sending $E_{KUa}[10]$ to be decrypted under Alice's Private key at Alice's end. Alice will add [*10*] to [*Urgent* and *Important*], and hashed them. If we assumed that the mail server contains 100 encrypted emails from Bob to Alice, and Alice would like to search for an important email in a short time. She will send the hashed values H [*Urgent* ||*10*], and H [*Important* ||*10*] to reach the encrypted message quickly.

Based on the above example, we can prove that EPEKS is convenient to handle multiple keywords search, besides the keywords could be increased depending on the known keywords which were assumed between Bob and Alice.

6- SECURITY RELATED TO EPEKS

The Efficient Public Key Encryption with Keyword Search EPEKS scheme based on variable values which are $E_{KUa}[M]$, $E_{KUa}[r]$, and H [$W \| r$]. In example 1, and 2 while presenting the refreshing keywords process, EPEKS proved its security. The proposed game between EPEKS and the attacker clarified that it is too difficult to break the system without learning the variable values which were changing each time.

Each encrypted message has its own variable value (r). we can complicate the system more, to reach high level of security, by asking Bob to change the keywords for each message. Therefore, each encrypted message will have its own keywords, and variable (r).

In addition to the above, EPEKS system based on public key cryptosystem, and hash function. Firstly, public key encryption scheme is vulnerable to a brute force attack. Public key systems depend on the use of some sort of invertible mathematical function. The complexity of calculating these functions may not scale linearly with the number of bits in the key grows more rapidly than that. Thus, the key size must be large enough to make brute force attack impractical but small enough for practical encryption and decryption. Another form of attack is to find some way to compute the private key given the public key. To date, it has not been mathematically proven that this form of attack is infeasible for a particular public key algorithm. Secondly, the

hash function, which is one way function, for any given code h, it is computationally infeasible to find x such that $H(x) = h$. The strength of a hash function against brute force attacks depends solely on the length of the hash code produced by the algorithm.

Based on the above, EPEKS scheme depends on the keyword W, the variable r, hashed value H $[W||r]$, and Alice's private key. If the attacker would like to reach the encrypted email, all the values must be known by the intruder, which is impossible to get all the values at the same time to break the scheme.

7- COMPUTATIONS AND COMPLEXITY

Assume that the public key algorithm is RSA, and hash algorithm is SHA-512. In [5], test is performed on Pentium III machine. The time required to encrypt the message approximately is 0.054 seconds, and the hashed value that is related to [6], could be obtained after 40.2 cycles/byte if we assumed that 1 block = 128 bytes. These calculations are done at Bob's side, the encryption stage. At Alice's side, the time required for sending the request could be negligible, also at the mail server side, the time required to send the total encrypted r values could be negligible. Alice receives the total r to obtain the hashed value; it could be similar to the first hashed value. The time required to search for the hashed value in the mail server database, depends on the size of the mail server, and the speed of the processor to execute one instruction, and it changes due to the processor model. Alice decrypts the message under her private key in 0.903 seconds.

8- CONCLUSIONS

In this paper we defined EPEKS the Efficient Public Key Encryption with Keyword Search mechanism. We explained the construction of the EPEKS. Constructing the EPEKS is related to public key cryptosystem, not Identity Based Encryption IBE which was used in the rest of PEKS papers. EPEKS is easier to be constructed than PEKS because any public key encryption algorithm can be used to construct EPEKS. We discussed the refreshing keywords process, and the multiple keywords search process. We described the security of the new scheme by using cryptographic algorithm and hash function.

In short, EPEKS provides high efficiently where any public key algorithm can be used widely in this scheme, high security where it is forbidden to either the mail server or any intruder to reach the keywords due to the refreshing process, and the multiple keywords, and high privacy, because it gives Alice the ability to be the only one who could search for her encrypted emails by using encrypted keywords.

REFERENCES

1. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, Public Key Encryption with Keyword Search, IN Eurocrypt 2004, LNCS 3027, pages 506-522, Springer - Verlag, 2004.
2. J. Baek, R. Naini, and W. Susilo. Public key encryption with keyword search revisited. Cryptology ePrint Archive, Report 2005/191, 2005. http://eprint.iacr.org/.

3. C. Gu, Y. Zhu, and Y. Zhang. Efficient Public Key Encryption with Keyword Search Schemes for pairings. Cryptology ePrint Archive, Report 2006/108, 2006. http://eprint.iacr.org/.
4. D. Khader. Public Key Encryption with Keyword Search based on K-Resilient IBE. Cryptology ePrint Archive, Report 2006/358, 2006. http://eprint.iacr.org/.
5. R. Biswas S. Bandyopadhyay, A. Banerjee. A fast implementation of the RSA algorithm using the GNU MP library. Research 2003. Available: http://www.cs.ucr.edu/~anirban/index.swf
6. A. Hartikainen, T. Toivanen, and H. Kiljunen. Whirlpool hashing function, Lappeenranta University of Technology.2006. Available: http://www.it.lut.fi/kurssit/05-06/Ti5318800/assign/Whirlpool
7. William Stallings, "Cryptography and Network Security", Principles and Practices, Third Edition, Prentice Hall, 2003.

التشفير الكفء بالمفتاح العام وكلمه البحث

الهدف الرئيسي من البحث هو بناء نظاما كفء للبحث عن كلمات مشفرة داخل الرسالة الالكترونية المشفرة. النظام يمكن بناءه باستخدام أكثر من نظام تشفير بالمفتاح العام. فعلى سبيل المثال إذا أراد المرسل (بوب) إرسال رسالة مشفرة باستخدام المفتاح العام للمستقبل (أليس)، وأرادت (أليس) بدورها أن تعطي الأولوية لخادم البريد ليختبر وجود كلمة " مهم " في الرسالة المشفرة مع عدم معرفة أي معلومة عن محتويات الرسالة. فباستخدام النظام الكفء لتشفير بالمفتاح العام نستطيع أن نصل إلى هذا الغرض، فالنظام يحتوي على نظام حماية عال وكفء لحماية خصوصيات المستقبل ورسائله من أن تسرق أو تزيف فلا يستطيع المتطفل أن يعلم محتوي الرسالة ولا الكلمات التي تم استعمالها من قبل المستقبل للوصول إلى الرسالة.أولا لقد تطرقنا في بداية هذا البحث إلى كيفية بناء هذا النظام باستخدام نظرية المفتاح العام ولم يتم بناءه بطريقة الأبحاث الأخرى. فطريقة بناء هذا النظام مختلفة تمام الاختلاف عن باقي الأبحاث.ثانيا لقد عرضنا بطريقة أمنة كيفية تجديد الكلمات المراد البحث عنها و كيفية استخدام أكثر من كلمة في البحث. و أخيرا لقد تم شرح أسلوب الأمان المتبع باستخدام كل من نظرية المفتاح العام ودالة المزج.