*Journal of Engineering Sciences, Assiut University, Vol. 39, No 3, pp.607 -624, May 2011*

# SECUIRTY USING THE EFFICIENT PUBLIC KEY ENCRYPTION WITH KEYWORD SEARCH SYSTEM

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

*(Received Jaunary18, 2011 Accepted Arial 18, 2011)*

*Efficient Public key encryption with key word search (EPEKS) 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. We have already published a previous paper explaining this scheme the Efficient Public Key Encryption with Keyword Search (EPEKS). In this paper, we focused on the relationship between the security and (EPEKS). Firstly we briefly review the construction of this 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 explained and compared between previous published papers and clarified the relationship between security and (EPEKS). Finally, we mentioned the encryption mechanisms regarding the system.*

***KEYWORDS**: EPEKS, Encryption email, Hash function, Mail server, Refreshing keywords, Security.*

## 1. INTRODUCTION

### 1.1 Basic Concept

In [1] 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.

[1] Bassant.Morgan@live.com
[2] Assistance Professor, Communication and Electronics Department, Helwan University
[3] Professor, Communication and Electronics Department, Helwan University.

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.

In this paper, we focused on the relationship between security and EPEKS. We mentioned some encryption mechanisms and clarified the security level of EPEKS based on their definitions.

## 1.2 Related work and our Contributions

There are few papers directly related to Public Encryption with keyword Search PEKS. In [1] the author built a new EPEKS scheme based on Public key Encryption. However, PEKS was first introduced in Boneh [2], and later improved in Baek [3], Gu [4], and Khader[5]. The works in [2, 3, 4 and 5] depend mainly on Pairing Based Cryptography, and Identity Based Encryption IBE.

In [1] the author presented scheme called EPEKS where Alice sends secret key to the receiver in order to get the original encrypted message from the mail server which stores all the incoming messages from the sender Bob. Once Alice receives a notification from the mail server stating that there is a message waiting for her, she sends the secret key to the mail server. The secret keys 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. The author clarified the security of the system by adding some functions to the system which are: Refreshing Keywords and Handling Multiple keywords search. In [2] the authors presented general scheme called PEKS where Alice gives trapdoors for the words she wants the gateway to search for. 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 [3] 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 [3] the authors pointed out two features that were not covered in [2]. 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 [3] 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 [4] 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 [5] 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 searches, and remove the need of the secure channels.

In [2-5] 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- Brief description for the construction of EPEKS scheme.
2- Security and EPEKS.
3- Encryption mechanisms and EPEKS.
4- Neither secure channel nor pairing has been discussed in this paper.

## 2. PRELIMINARIES

In this section we will go through brief description for the construction of EPEKS scheme.

### 2.1 Definitions of EPEKS

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.

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.1.

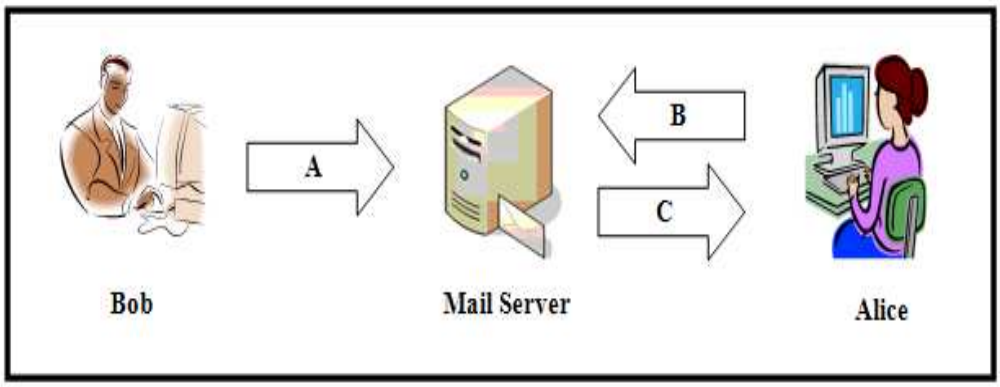


Fig.1 The parties of the new scheme EPEKS

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.

### 2.1.1 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*).

### 2.1.2 The mail server

Contains a database which consists of the encrypted email Ekua(M), the hashed value H(W||r), and the encrypted variable value Ekua(r).

### 2.1.3 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*).

## 2.2 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.

### 2.2.1 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 KUa. 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 Ekua(M), the encrypted variable value Ekua(r), and the hashed value

H(W||10). The outputs will act as inputs to Alice's mail server as shown in Fig.2. 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

$x_1 = E_{KUa}[M]$

$x_2 = H[W \| r]$

$x_3 = E_{KUa}[r]$

$X = x_1 \| x_2 \| x_3$

$$X = E_{KUa}[M] \| H[W \| r] \| E_{KUa}[r] \quad (1)$$

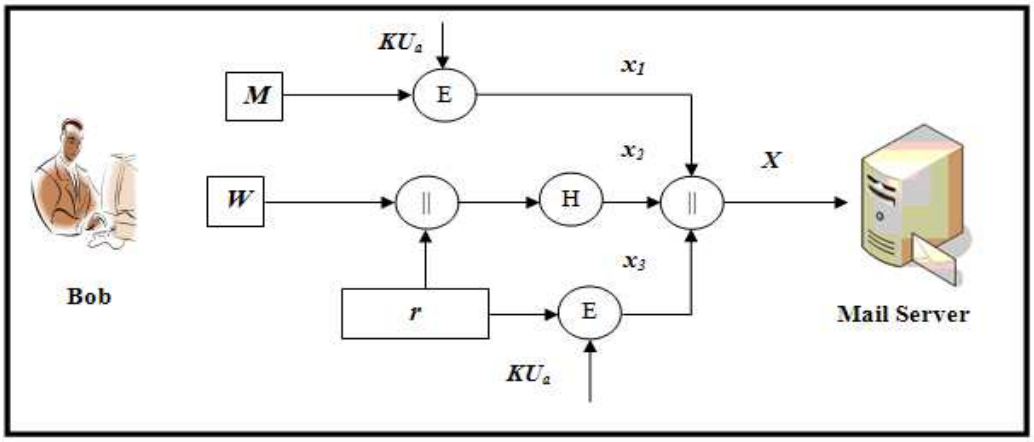


Fig.2 Sender Party

## B) Mail Server Party

The mail server receives Eq.1 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 3.

**Table 1 Mail Server Database**

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

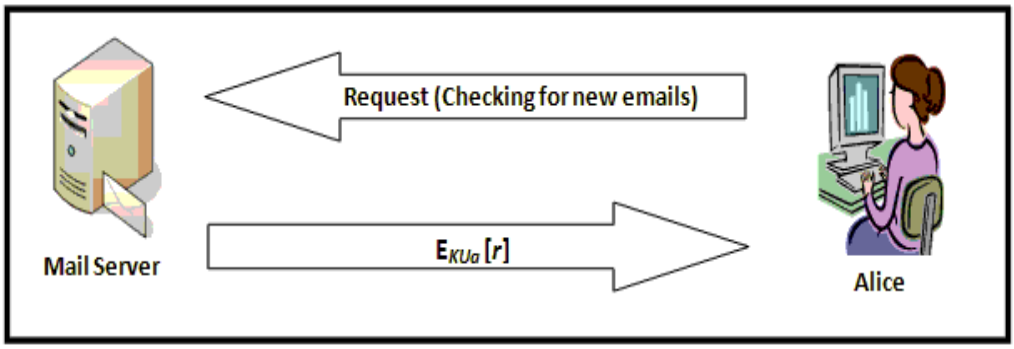


Fig.3 The communication between the Mail Server and the Receiver

### 2.2.2 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 EKUa [r]. Alice decrypts the r by using her private key DKRa [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.4

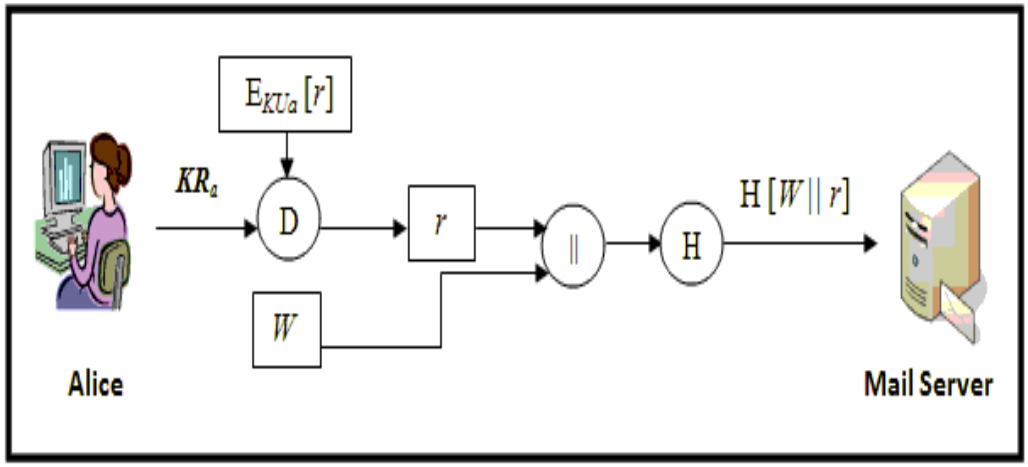


Fig.4 The Decryption Process at the Receiver Party

### 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.5.

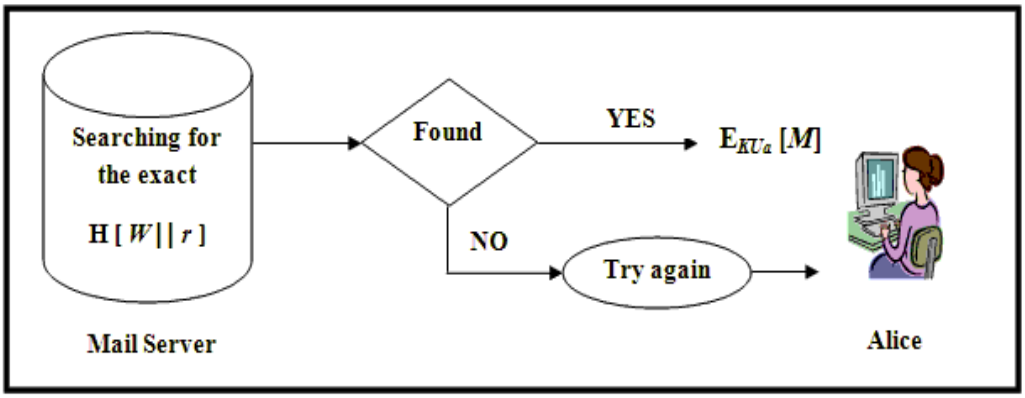


Fig.5 Matching process in the Mail Server

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 [1] is more instance than this section.

## 3. SECURITY AND EPEKS

In [2], the author mentioned that for a PEKS to be considered secure he needed to guarantee that no information about a keyword is revealed unless the trapdoor of that word is available. To define security against an active adversary A he used the following game between A and challenger.

- CKA-Setup: The challenger runs the key generation algorithm and gives the Apub to adversary A and keeps Apriv to itself.
- CKA-Phase 1: A asks the challenger for trapdoors corresponding to keywords of its choice.
- CKA-Challenge: The adversary decides when phase 1 ends. Then it chooses two words W0,W1 to be challenged on. The two words should not be among those for which A obtained a trapdoor in phase 1. The challenger picks a random bit b ∈ {0, 1} and gives attacker. C = PEKS (Apub,Wb).
- CKA-Phase 2: A asks for more trapdoors like in phase 1 for any word of its choice except for the W0, W1.
- CKA-Guess: A outputs its guess of b' and if b' = b that means A guessed the encrypted message and the adversary wins.

He said that the scheme is secure against a chosen keyword attack (CKA) if A has a low advantage of guessing the right word being encrypted.

He proved that this system is a non-interactive searchable encryption scheme semantically secure against a chosen keyword attack in the random oracle model. The proof of security relies on the difficulty of the Bilinear Diffie-Hellman problem (BDH). Indeed, it is currently an open problem to build a secure IBE, and hence a PEKS, without the random oracle model. In [3] the authors used the security notion for PEKS schemes, "indistinguishability of PEKS against chosen keyword attack" which was introduced in [2] to present important issues regarding PEKS, which were not addressed in [2] paper which are: Refreshing keywords, Removing Secure Channel, and Handling Multiple keywords. Firstly, the author discussed the refreshing keywords

issue and he presented a method which makes the size of a keyword space infinite and makes it useless for the server to keep trapdoors. Hence, the security notion for PEKS scheme IND – CKA, now becomes meaningful in reality. Secondly, In the PEKS scheme in [2] there is a need to have a secure channel between Alice and the server, so that an eavesdropper (Eve) cannot get hold of the trapdoors sent. No one but the server should be capable of testing emails for certain keywords. This is one of the drawback that the authors of [3] tried to solve by generating a public and a private key that belong to the server. The PEKS algorithm was modified to encrypt keywords using both Alice's and the server's public key, while the testing algorithm needs the server's private key as an input. In this way the scheme is secure channel free (SCF-PEKS) because Eve cannot obtain the server's private key, therefore cannot test.

The SCF-PEKS is said to be IND-SCF-CKA secure when it ensures that the server that has obtained the trapdoors for given keywords cannot tell a PEKS ciphertext is the result of encrypting which keyword, and an outsider adversary that did not get the server's private key cannot distinguish the PEKS ciphertexts, even if it gets all the trapdoors for the keywords that it queries. He proved that IND – SCF – CKA secure in the random oracle model assuming that the BDH problem is intractable. Finally, he mentioned that in practice, one may need to relate multiple keywords to one message. As Boneh [2] suggested, one can achieve this by simply creating E(pkR,M)||PEKS(pkR,w1)||…….||PEKS(pkR,wn), where E denotes a secure public key encryption function, however, that no formalization, efficient construction, and issues related to disjunctive and conjunctive search were given in [2], and he dealt with these problems by defining a PEKS scheme with multiple keywords and defining a security notion for MPEKS, which he called "indistinguishability of PEKS with multiple keyword search against chosen keyword attack (IND-SCF-CKA)". He proved that IND – MK – CKA secure in the random model assuming that BDH problem is intractable. However, at the end of his paper, he concluded that the server's attack by storing trapdoors seems to be inherent weakness of PEKS. Another open problem is to find a more efficient and convenient way to refresh frequently-used keywords than the one proposed in his paper. In [4], the author provided further IND – MK – CKA discussion on the notion of SCF – PEKS scheme, gave a formal security model and presented an efficient SCF – PEKS scheme. The new scheme can also be proved to be secure in the random oracle model. In [5], a new scheme was suggested for conjunctive search called PECK. The scheme substitutes the PEKS algorithm with a PECK algorithm that encrypts a query of keywords. The testing is done with a trapdoor for each query instead of each word. So Bob sends Alice the following:

$$[\mathrm{E}(A_{\mathrm{pub}},M),\ \mathrm{PECK}(A_{pub},\ (W_1,W_2,\ ...,W_m))]$$

She said that the scheme is secure against a chosen keyword attack (CKA) if an adversary has a low advantage in guessing the right query of keywords being encrypted. The author constructed a new scheme (KR-PEKS) the KResilient Public Key Encryption with Keyword Search. The new scheme is secure under a chosen keyword attack without the random oracle. The ability of constructing a Public Key Encryption with Keyword Search from an Identity Based Encryption was used in the construction of the KR-PEKS. The security of the new scheme was proved by showing that the used IBE has a notion of key privacy. She showed that since the PEKS was built from the KRIBE and KRIBE has key privacy notions then PEKS should logically

be proved to be secure under a CKA. The K-Resilient PEKS scheme [5] is based on the Decisional Diffie Hellman problem (DDH). The security of such scheme is based on the difficulty of solving DDH and whether the hash functions used are collision resistant or not.

In this paper, we haven't mentioned any secure channel. We discussed refreshing keywords and handling multiple keywords under chosen keyword attack.

## 3.1 Refreshing keywords:

In [1] refreshing keywords in our mechanism depends on number of variable values which are: a variable secret value r which will be added to the keyword, and a hashed value, 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

Assume Bob and Alice have chosen a keyword W = Urgent.

Bob encrypts M under Alice's public key EKUa[M].

Bob chooses and encrypts variable secret value r under Alice's public key EKUa[r]. r and W will be hashed by hash function at Bob's end H [ W ||r ].

Assume that the attacker is the administrator of the mail server.

The three values EKUa[M], EKUa[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 CKA 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 in EPEKS against CKA has been proved in this document through the below example.

Example 2: Bob would like to send Alice two messages M1,M2 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 one (1).

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 attacker would like to get to W, it will be impossible because he doesn't know either r1 or r2 to reach the hashed values. In case if he gets either r1 or r2, still W is unknown to get the hashed value. Due to H1 [Urgent ||10] is not equal to H2 [Urgent ||20], it will be difficult for the attacker 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 under CKA.

## 3.2 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 searches in a PEKS is the capability of searching for more than one word either disjunctively or conjunctively. She continued that in [2] 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 searches, 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.

**Example3:** 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* ].

EKUa[M], H [Urgent ||10 ], H [ Important ||10 ], and EKUa[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 EKUa[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. Besides, the more keywords will be used in the scheme the more complex the scheme will be. EPEKS scheme is secure against a chosen keyword search attack (CKA) has a low advantage in guessing the right keywords being encrypted. To break the scheme, it is recommended from the attacker to guess either the keywords or the variable r in order to reach the encrypted message.

# 4. ENCRYPTION MECHANISMS AND EPEKS

## 4.1 Semantic Security

Semantic security is a widely-used definition for security in an asymmetric key encryption algorithm. For a cryptosystem to be semantically secure, it must be infeasible for a computationally-bounded adversary to derive significant information about a message (plaintext) when given only its ciphertext and the corresponding public encryption key. Semantic security considers only the case of a "passive" attacker, i.e., one who generates and observes ciphertexts using the public key and plaintexts of her choice. Unlike other security definitions, semantic security does not consider the case of chosen ciphertext attack (CCA), where an attacker is able to request the decryption of chosen ciphertexts, and many semantically-secure encryption schemes are demonstrably insecure against chosen ciphertext attack. Consequently, semantic security is now considered an insufficient condition for securing a general-purpose encryption scheme.

In [6] the notion of semantic security was first put forward by Goldwasser and Micali in 1982. However, the definition they initially proposed offered no straightforward means to prove the security of practical cryptosystems. Goldwasser/Micali subsequently demonstrated that semantic security is equivalent to the definition of security called ciphertext indistinguishability. [7] This later definition is more common than the original definition of semantic security because it better facilitates proving the security of practical cryptosystems.

Indistinguishability under Chosen Plaintext Attack (IND-CPA) is commonly defined by the following game:

1. A probabilistic polynomial time-bounded adversary is given a public key, which it may use to generate any number of ciphertexts (within polynomial bounds).

2. The adversary generates two equal-length messages $m_0$ and $m_1$, and transmits them to a challenge oracle along with the public key.
3. The challenge oracle selects one of the messages by flipping a fair coin, encrypts the message under the public key, and returns the resulting ciphertext c to the adversary.

The underlying cryptosystem is IND-CPA (and thus semantically secure under chosen plaintext attack) if the adversary cannot determine which of the two messages was chosen by the oracle, with probability significantly greater than 1 / 2 (the success rate of random guessing). Variants of this definition define indistinguishability under chosen ciphertext attack and adaptive chosen ciphertext attack (IND-CCA, IND-CCA2).

Because the adversary possesses the public encryption key in the above game, a semantically secure encryption scheme must by definition be probabilistic, possessing a component of randomness; if this were not the case, the adversary could simply compute the deterministic encryption of m0 and m1 and compare these encryptions with the returned ciphertext c to successfully guess the oracle's choice.

Semantically secure encryption algorithms include Goldwasser-Micali, El Gamal and Paillier. These schemes are considered provably secure, as their semantic security can be reduced to solving some hard mathematical problem (e.g., Decisional Diffie-Hellman or the Quadratic Residuosity Problem). Other, non-semantically-secure algorithms such as RSA, can be made semantically secure (under stronger assumptions) through the use of random encryption padding schemes such as Optimal Asymmetric Encryption Padding (OAEP).

The EPEKS mechanism is secure against passive attacker because the mechanism goes into three rounds. The first round when the mail server sends EApub(r) to Alice (the receiver). At this point, if we assumed that the attacker knows two variables (r1, r2) and we applied the above game, then EPEKS is IND-CPA (and thus semantically secure under chosen plaintext attack) if the adversary cannot determine which of the two variable values r was chosen with probability significantly greater than 1/ 2 (the success rate of random guessing). The second round when the receiver gets the variable (r) and adds the keyword W to hashes them to get the required hashed value and returned the value back to the mail server. In the third round, the mail server searches and matches the hashed value in its database and returns back the possible encrypted message to Alice. For the attacker to get to the real message (plaintext), firstly, the attacker must assume that he knows the hashed value which a combination between the variable value r and the keyword W. Consequently, if we assumed that the attacked got the hashed value and he would be able to send the hashed value to the mail server on the receiver's behalf to get the message, he must know the receiver's private key to decrypt the message. Secondly, the EPEKS has the refreshing keywords function which uses different values of r for each message per sender, which means that the attacker needs to break all the hashed values of the real messages which come from the mail server. The probability to this assumption is very week because each message has its own r and W and both of them are variables, therefore the hashed value won't be constant. It will be changed regarding the changing of r and W.

## 4.2 Chosen-plaintext attack (CPA)

CPA is an attack model for cryptanalysis which presumes that the attacker has the capability to choose arbitrary plaintexts to be encrypted and obtain the corresponding ciphertexts. The goal of the attack is to gain some further information which reduces the security of the encryption scheme. In the worst case, a chosen-plaintext attack could reveal the scheme's secret key.

This appears, at first glance, to be an unrealistic model; it would certainly be unlikely that an attacker could persuade a human cryptographer to encrypt large amounts of plaintexts of the attacker's choosing. Modern cryptography, on the other hand, is implemented in software or hardware and is used for a diverse range of applications; for many cases, a chosen-plaintext attack is often very feasible. Chosen-plaintext attacks become extremely important in the context of public key cryptography, where the encryption key is public and attackers can encrypt any plaintext they choose.

Any cipher that can prevent chosen-plaintext attacks is then also guaranteed to be secure against known-plaintext and ciphertext-only attacks; this is a conservative approach to security.

In our paper, we proved that EPEKS mechanism based on variables of values r, W and H [r‖W]. For the attacker to choose arbitrary plaintexts to be encrypted and obtain the corresponding ciphertexts, the attacker must try to get the variable r which will be sent by the mail server to the receiver. In this mechanism, r changes regarding the message per sender; therefore, the ciphertext will be changed too. If we assumed that the attacker would like to get EApub(M), then he needs to get the hashed value which is a variable value per sender as well regarding the values r and W.

## 4.3 Chosen-ciphertext attack (CCA)

CCA is an attack model for cryptanalysis in which the cryptanalyst gathers information, at least in part, by choosing a ciphertext and obtaining its decryption under an unknown key. In the attack, an adversary has a chance to enter one or more known ciphertexts into the system and obtain the resulting plaintexts. From these pieces of information the adversary can attempt to recover the hidden secret key used for decryption.

A number of otherwise secure schemes can be defeated under chosen-ciphertext attack. For example, the El Gamal cryptosystem is semantically secure under chosen-plaintext attack, but this semantic security can be trivially defeated under a chosen-ciphertext attack. Early versions of RSA padding used in the SSL protocol were vulnerable to a sophisticated adaptive chosen-ciphertext attack which revealed SSL session keys. Chosen-ciphertext attacks have implications for some self-synchronizing stream ciphers as well. Designers of tamper-resistant cryptographic smart cards must be particularly cognizant of these attacks, as these devices may be completely under the control of an adversary, who can issue a large number of chosen-ciphertexts in an attempt to recover the hidden secret key.

When a cryptosystem is vulnerable to chosen-ciphertext attack, implementers must be careful to avoid situations in which an adversary might be able to decrypt chosen-ciphertexts (i.e., avoid providing a decryption oracle). This can be more

difficult than it appears, as even partially-chosen-ciphertexts can permit subtle attacks. Additionally, some cryptosystems (such as RSA) use the same mechanism to sign messages and to decrypt them. This permits attacks when hashing is not used on the message to be signed. A better approach is to use a cryptosystem which is provably secure under chosen-ciphertext attack, including (among others) RSA-OAEP, Cramer-Shoup and many forms of authenticated symmetric encryption.

In simple words, in this attack, the attacker has somehow acquired some encrypted data and he doesn't know what it means. Usually this data is captured off a network connection with a sniffer. Boris has two ways to try to crack the ciphertext:

- He can send the ciphertext back to the victim and social-engineer the victim to decrypt it and send it back. With both the ciphertext and plaintext, the attacker can figure out the key.
- He can try to find some plaintext that is probably included in the ciphertext and work backwards from there.

This sort of attack is usually tried against e-mail that has been encrypted with a public/private key combination.

Assume that the attacker would like to walk through the first way to figure out the hidden key. In EPEKS, Bob (the sender) sends three different values EKUa[M], EKUa[r], and H [ W ||r ] to the mail server. The three values will be stored in the mail server database and we have already assumed that the attacker is the mail server administrator. Formally, we define security against administrator using the following game between EPEKS and the administrator:

1- If the administrator sends $E_{KUa}[r]$ to Alice to get the real message. that won't be happened because regarding the EPEKS system, H [ $W \| r$ ] will be sent to the mail server to be matched with others hashed values sorted in the mail server database. Therefore, the administrator will get a hashed value hidden underneath it the keyword and the original r not the plaintext (original message). In this round, he won't be able to get the real message.
2- If the administrator send $E_{KUa}[M]$ to Alice, Alice will keep the original message stored at her mail box and won't resend it back to the mail server because it is not logic to resend the decrypted message to the mail server.

Assume that the administrator will walk through the second way to get the plaintext (the original message). Based on the EPEKS construction we will notice that the three variable values are stored in the mail server data base and if we assumed that the administrator has the ability to access these information in order to try to get the original message that means that he will focus on the keyword as it might has any information related to the original massage to be able to know its contents. Actually the keyword is added to the variable r and hashed together to get a complete hashed value. So the administrator needs to break the hashed value and the main concept for CCA is to gather information in order to get the hidden key not break the hashed value. If we assumed that the administrator would like to focus on r instead of w, then he needs to gather all the encrypted r in order to get to the hidden key but by using the EPEKS mechanism which stating that each message has its own r which will be very difficult for the administrator to get the hidden key.

Chosen-ciphertext attacks, like other attacks, may be adaptive or non-adaptive. In a non-adaptive attack, the attacker chooses the ciphertext or ciphertexts to decrypt in advance, and does not use the resulting plaintexts to inform their choice for more

ciphertexts. In an adaptive chosen-ciphertext attack, the attacker makes their ciphertext choices adaptively, that is, depending on the result of prior decryptions.

### 4.3.1 Adaptive chosen-ciphertext attack

An adaptive chosen-ciphertext attack (abbreviated as CCA2) is an interactive form of chosen-ciphertext attack in which an attacker sends a number of ciphertexts to be decrypted, and then uses the results of these decryptions to select subsequent ciphertexts. It is to be distinguished from an indifferent chosen-ciphertext attack (CCA1).

The goal of this attack is to gradually reveal information about an encrypted message, or about the decryption key itself. For public-key systems, adaptive-chosen-ciphertexts are generally applicable only when they have the property of ciphertext malleability — that is, a ciphertext can be modified in specific ways that will have a predictable effect on the decryption of that message.

#### 4.3.1.1 Practical attacks

Adaptive-chosen-ciphertext attacks were largely considered to be a theoretical concern until 1998, when Daniel Bleichenbacher of Bell Laboratories demonstrated a practical attack against systems using RSA encryption in concert with the PKCS#1 v1 encoding function, including a version of the Secure Socket Layer (SSL) protocol used by thousands of web servers at the time [8].

The Bleichenbacher attacks took advantage of flaws within the PKCS #1 function to gradually reveal the content of an RSA encrypted message. Doing this requires sending several million test ciphertexts to the decryption device (eg, SSL-equipped web server.) In practical terms, this means that an SSL session key can be exposed in a reasonable amount of time, perhaps a day or less.

#### 4.3.1.2 Preventing attacks

In order to prevent adaptive-chosen-ciphertext attacks, it is necessary to use an encryption or encoding scheme that limits ciphertext malleability. A number of encoding schemes have been proposed; the most common standard for RSA encryption is Optimal Asymmetric Encryption Padding (OAEP). Unlike ad-hoc schemes such as the padding used in the early versions of PKCS#1, OAEP has been proven secure in the random oracle model [9]. OAEP was incorporated into PKCS#1 as of version 2.0 published in 1998 as the now-recommended encoding scheme, with the older scheme still supported but not recommended for new applications.

#### 4.3.1.3 Mathematical model

In complexity-theoretic cryptography, security against adaptive chosen-ciphertext attacks is commonly modeled using ciphertext indistinguishability (IND-CCA2).

In our mechanism, if we give the attacker the chance to choose cipher texts and send them to the system to be able to get the secret key or the real message itself, it will be very difficult to get them. The EPEKS system is based on variables. The first variable is r, if we assumed the attacker will send to the system r on the behalf of the mail server. Then the receiver will get r and try to add it to the known keyword from

his choice to get the hashed value therefore to send it to the mail server again to get the correct message. If the mail server didn't get the correct hashed value, it will send a message asking the receiver to try again, consequently the attacker won't be able to get to the real message unless he knows the keyword W (and the keyword is known only for both the sender and the receiver). The second variable is the keyword W, as it is known for both the sender and the receiver. If we assumed that the attacker succeeds to get to the keyword W and gets the hashed vales as well. Then at this point, you might think that the system has been attacked, but this is not true because the system has two functions which the refreshing keyword, and the multiple keywords. Refreshing keyword gives the system the availability to add number of variables for each message at the same round. Therefore, it is a must for the attacker to get all the variable values r and to guess their keyword to be able to reach the ciphertext and get either the real message or the secret key. Multiple keywords is another function for EPEKS, if we assumed the both the sender and the receiver have decided to choose 2 keywords instead of one and use one variable value r for both keywords, then the attacker must guess the correct 2 keywords and added them to r to be able to break the mechanism.

# 5. EFFICENT CONSIDERATION, COMPUTATIONS AND COMPLEXITY

## 5.1 Traffic (Transmission):

From this paper, you will notice that we are making number of traffic especially between the mail server and the receiver (Alice). The journey of the traffic begins by sending the mail server a notification for the receiver (Alice) that it has received a new message for her. At this moment, the mail server sent the encrypted value (r). The second trip, when the receiver Alice sends the hashed value in order to get the real encrypted message. The third trip, when the mail server sends the real message to the receiver (Alice). At this stage, anyone can think that the traffic scenario has been ended. This is not true. On other words, if we would like to add another scenario, we can assume that Alice has sent a wrong hashed value which is related to an incorrect value (r) or an incorrect known word (W), consequently the hashed value was wrong. In this case, the mail server will create a new traffic to Alice asks her to resend the correct hashed value in order to get the correct message. You will notice that the traffic in the EPEKS mechanism or the transmission is different than what was presented in Bon's paper. The traffic can be seen as a gap in this mechanism which will allow the trackers to try to get any information to get the real message or the secret key or to be under the traffic analysis control. Traffic analysis is the process of intercepting and examining messages in order to deduce information from patterns in communication. It can be performed even when the messages are encrypted and cannot be decrypted. In general, the greater the number of messages observed, or even intercepted and stored, the more can be inferred from the traffic. Traffic analysis can be performed in the context of military intelligence or counter-intelligence, and is a concern in computer security.

Traffic analysis tasks may be supported by dedicated computer software programs, including commercially available programs such as those offered by i2, Visual Analytics, Memex, Orion Scientific, Pacific Northwest National Labs, Genesis

EW's GenCOM Suite and others. Advanced traffic analysis techniques may include various forms of social network analysis. To fix the above issue and to protect our mechanism from the traffic analysis, a Traffic flow security has appeared. Traffic-flow security is the use of measures that conceal the presence and properties of valid messages on a network to prevent traffic analysis. This can be done by operational procedures or by the protection resulting from features inherent in some cryptographic equipment. In the encryption mechanism section which found in this paper, we have shown and proved how EPEKS is difficult to be attacked as EPEKS depends on number of variable values which made multiple keywords and the refreshing keywords advantages.

### 5.2 Computation:

Assume that the public key algorithm is RSA, and hash algorithm is SHA-512. In [10], 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 [11], 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.

## 6. 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 and we mentioned the encryption mechanisms regarding the system.

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. B. Morgan, M. Hamada, and G. Abdelfadel. Efficient Public Key Encryption with Keyword Search. Journal of Engineering Science, Assuit University, Vol. 38, No. 3, pp.749-761, May 2010.

2. 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.
3. J. Baek, R. Naini, and W. Susilo. Public key encryption with keyword search revisited. Cryptology ePrint Archive, Report 2005/191, 2005.
4. C. Gu, Y. Zhu, and Y. Zhang. Efficient Public Key Encryption with Keyword Search Schemes for pairings. Cryptology ePrint Archive, Report 2006/108, 2006.
5. D. Khader. Public Key Encryption with Keyword Search based on K-Resilient IBE. Cryptology ePrint Archive, Report 2006/358, 2006.
6. Goldwasser and S. Micali, Probabilistic encryption & how to play mental poker keeping secret all partial information, Annual ACM Symposium on Theory of Computing, 1982.
7. Goldwasser and S. Micali, Probabilistic encryption. Journal of Computer and System Sciences, 28:270-299, 1984.
8. Bleichenbacher, Daniel (1998). "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1" (PDF). CRYPTO '98. pp. 1–12. http://www.springerlink.com/index/j5758n240017h867.pdf. Retrieved 2009-01-13.12 2009
9. Fujisaki, Eiichiro; Okamoto, Tatsuaki; Pointcheval, David; Stern, Jacques (2004). "RSA-OAEP Is Secure under the RSA Assumption". Journal of Cryptology (Springer) 17 (2): 81–104. doi:10.1007/s00145-002-0204-y. http://www.iacr.org/cryptodb/archive/2001/CRYPTO/21390260.pdf. Retrieved 2009-01-12. 2009
10. 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. 2003
11. 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. 2006

## 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.