Improved Secure and Privacy Preserving Keyword Searching Cryptography

Mohammad M. Tajiki [1] and Mohammad A. Akhaee [1]

[1]School of Computer and Electrical Engineering, College of Engineering, University of Tehran, Tehran, Iran.

***Abstract*—Using storage systems outside a company may endanger data security. This leads users to mostly encrypt their information for risk mitigation. Although encryption improves confidentiality, it causes inefficiency such as the encrypted data is not easily searchable. In this paper, data would be stored in a cloud storage provider (CSP) in a way that it is secure and simultaneously searchable. To this end, secure and privacy preserving keyword searching (SPKS) scheme has been employed and since it has a security defect, an improved version of SPKS has been introduced. The improvement consists of embedding an authentication process within the algorithm in a way that avoid imposing any communication overhead. The encryption algorithm employs CSP for partially decryption of the cipher texts. Consequently, the client computational and communication overhead in decryption will be reduced. Although the CSP participates in the deciphering process, we show that it cannot detect any information about the plaintext.**

***Index Terms*—Searchable encryption, Cloud storage data security, Public key encryption;**

I. Introduction

Recent progress in the network technology and increasing demands for IT resources makes companies to outsource their IT equipment [1].In the contexts, the data center hardware and software is referred to as a cloud. The Cloud allows users to pay for whatever they use [2].The application of encryption is to provide confidentiality by hiding all useful information about the plaintext. Encryption, on the other hand, often renders data useless in the sense that one loses the ability to do operation on it [3].

Since abstraction decreases the computational requirements of the client system, cloud technology will try to employ abstraction. However, there exist a lot of security problems in this way [4, 5, 6]. Sometimes it is rational for the encrypted message to have some information about the plaintext in order to avoid some upcoming difficulties in the cloud storage systems [3]. As an example, searching in the stored data is inevitable for cloud storage servers (CSS) showing why searchable encryption is demanding [7]. If a server does not apply this type of encryption, it has to send the encrypted message to its clients completely. The client should decrypt the entire data and checks if it is the proper case or not. This will impose lots of CPU processing overhead. Moreover, a great deal of network communication overhead would be occurred. These effects contradict with the goal of using cloud systems [9, 10]. This fact leads CSS to exploit a searchable encryption method.

Using an encryption scheme with the ability of keyword searching was first introduced by Song *et al.* [11]. Since then, there have been plenty of researches in this field [9]-[14]. Boneh *et al.* [12] proposed a public key encryption with keyword searching (PEKS) scheme, which customs a gateway to test whether certain keywords contained in an email. Liu *et al.* [8] improved the performance of PEKS method and introduced an efficient privacy preserving keyword searching scheme (EPPKS). Inspired by the work of Diament *et al.* [22], EPPKS announced the notation of partial decipherment. The SPKS[1] algorithm is the improved version of EPPKS offered by Liu *et al.* [8]. Kamara *et al.* [1] proposed some searchable encryption techniques in their technical reports which does not attract practitioners attention due to the lack of description.

In this paper, the SPKS as a special searchable encryption method is introduced [8] and its security has been analyzed. After finding a security bug in this algorithm, which causes an attacker to forge the encrypted message, a modification has been proposed to overcome the forging attack. We will proof that the

mahditajiki@ut.ac.ir
akhaee@ut.ac.ir

[1]Secure and privacy preserving keyword searching

improved SPKS is secure against forging the client signature. Finally we make a comparison between the SPKS and its improved version for better clarification.

The rest of the paper is organized as follows: in section 2, the principal of the SPKS algorithm is introduced. Section 3 presents an attack to the SPKS along with our improvement upon the algorithm. It also includes the comparison between the old SPKS and its improved version. Security proof of the modified SPKS algorithm would be proposed in Section 4. Finally, Section 5 concludes the paper.

II. An introduction to SPKS

In this section, some preliminaries which are required throughout the paper are reviewed. After illustrating the general view of SPKS method, we will present the details of this algorithm.

A. Preliminaries

Let G_1 and G_2 be two cyclic groups of a large prime order q. We view G_1 as an additive group and G_2 as a multiplicative one [1].

Bilinear map: we call $G_1 \times G_1 \rightarrow G_2$ a bilinear map if:

1) It is computable in polynomial time
2) For all $g, h \in G_1$ and $x, y \in Z_q^*$ we have $e(g^x, h^y) = e(g, h)^{xy}$
3) If g is the generator of G_1 then e(g, g) is the generator of G_2

BDH[2] parameter generator: IG[3] is called a BDH parameter generator if IG takes a sufficiently large parameter k which is greater than zero, runs in polynomial time in k, and outputs G_1, G_2 and a bilinear map $e : G_1 \times G_1 \rightarrow G_2$

BDH assumption: let IG be a BDH parameter generator and (G_1, G_2, e) are its outputs for a large number. Let $x, y, z \in Z_q^*$ be some random element. Given g^x, g^y, g^z, and g, the advantage $ADV_G[B]$ of algorithm *B* in finding $e(g, g)^{xyz}$ is negligible.

The client message which is going to be saved on a cloud storage server will be referred to as M. w_i are the keywords which are going to be searched. A keyword, $w_i \subset \{0, 1\}^*$, can be a title, date of creation or any other specification of information. $x, y \in Z_q^*$ are the private keys of client and server, while the public keys of the client and server denoted as $g^x, g^y \in G_1$, respectively. There exist five hash functions as below:

$$H_1, H_3 : \{0, 1\}^* \rightarrow G_1^*$$
$$H_2, H_5 : G_2 \rightarrow \{0, 1\}^{log\ q}$$
$$H_4 : G_2 \rightarrow \{0, 1\}^n \quad (1)$$

B. SPKS functions description

The SPKS algorithm has seven random functions with polynomial time that are described below.

i. **KeyGen**[4]:This function takes sufficiently large parameters k_1 and k_2, and produces public/private keys of the client and server.
ii. **EMBEnc**[5]:A public key encryption algorithm which takes public keys of the client and server as input. C_m, the encrypted message, is the output of this function.
iii. **KWEnc**[6]:A public key encryption method based on the public key of the client that encrypts the keywords. Output of this step is C_w.
iv. **TComput**[7]:Takes a keyword and the private key of the client as input and produce a trapdoor for the keyword.
v. **KWTest**[8]:Takes the client public key as input, the trapdoor, and the encrypted message (g^x, T_{w_j} and C_{w_j}) respectively. If T_{w_j} belongs to a keyword which is encrypted as C_{w_j}, the function will return 1, else 0 would be returned.
vi. **PDecrypt**[9]:It will partially decrypt the encrypted message using the server private key and sends it to the client.
vii. **Recovery**:in this function the partially decrypted message will be completely decrypted using the private key of the client.

C. Implementation of SPKS

Up to now, a general outline of the SPKS is presented. In the following, we will describe the details of the encryption and decryption algorithm.

1) Encryption: Two random elements $r \in Z_q^*$ and $p \in \{0, 1\}^n$ are generated. K_1 and K_2 are sufficiently large private parameters. Based on K_1, and K_2 , the public/private keys of the user and server will be

[2]Bilinear Deffie-Hellman
[3]Instance Generator
[4]Key Generator
[5]Email Message Body encryption
[6]Keywords encryption
[7]Trapdoor Computation
[8]Keywords Test
[9]Partial Decryption

produced. Thereafter, the following values are calculated:

$$U_1 = g^r$$

$$U_2 = p \bigoplus H_5(e(g^r, g^y)^x)$$

$$U_3 = m \bigoplus H_4(e(H_3(p), g^x)^r \quad (2)$$

$C_m = (U_1, U_2, U_3)$ is the encrypted message. The keywords are encrypted in the following manner:

$$C_{w_j} = H_2(e(g^x, H_1(W_j)^r)) \quad (3)$$

The encrypted keywords are sent along with the encrypted message to the server as follows.

$$(C_m, C_{w_1}, C_{w_2}, ..., C_{w_k}) \quad (4)$$

2) Decryption: In order to achieve the plaintext of the cipher text, the server performs a partial decryption (C_p) using its private key. However, the final decryption is done in the client system.

$$C_p = e(H_3(p), g^r) \quad (5)$$

Note that p can be computed in the following way:

$$p = U_2 \bigoplus H_5(e(g^x, g^y)^r) = U_2 \bigoplus H_5(e(g^x, g^r)^y)$$
$$= U_2 \bigoplus H_5(e(g^x, U_1)^y)$$

Partial decipherment is possible here using the private key of the server or user. This make it infeasible for a third party to do it without the private key. After partial decipherment, the pair (U_3, C_p) is sent to the user. Finally, the user accomplishes the final decryption.

$$m = U_3 \bigoplus H_4(C_p^x) \quad (6)$$

3) Search : After investigating the encryption and decryption stages, the search algorithm is explained here. The algorithm will make a trapdoor to look for a special keyword.

$$T_{w_j} = H_1(W_j)^x \quad (7)$$

Then, T_{w_j} is sent to the server. The server will check the following constraint:

$$C_{w_i} = H_2(e(U_1, T_{w_j})) \quad (8)$$

If the encrypted keyword is the correct one, it will be partially decrypted and sent to the client. It should be noticed that if $W_i = W_j$ then:
$C_{w_i} = H_2(e(g^x, H_1(W_i)^r))$
$= H_2(e(g^r, H_1(W_i)^x)) = H_2(e(g^r, H_1(W_j)^x))$

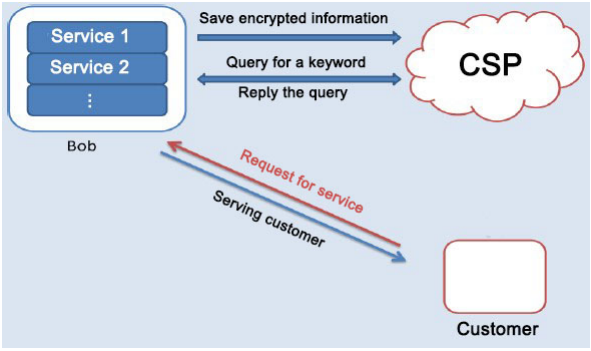

Fig. 1. Normal communication

$$= H_2(e(g^r, T_{w_j}))$$

Therefore, server can check which cipher text is requested.

III. The Proposed Attack On SPKS And Its Solution

Now, we analyze the SPKS algorithm and show the possible attack to this scheme. Then, our improvement will be proposed.

A. Forging Attack

The attack is presenting via an example. Let Bob stores information of the customers who were paid for a special service in a cloud storage provider (CSP). When someone requests the service, as it is illustrated in Figure 1, Bob sends the customer name to the CSP to check whether he/she is on the list. It is worth mentioning that CSP can acquire no information about the requested search.

According to Figure 2, the attacker will send a record to the CSP before sending the service request to Bob. In this way, the attacker generates two random number $r \in Z_q$ and $p \in \{0,1\}^n$. As g^x, g^y are plain to everyone, attacker makes the following variables.

$$U_1 = g^r$$

$$U_2 = p \bigoplus H_5(e(g^y, g^x)^r)$$

$$U_3 = m \bigoplus H_4(e(H_3(p), g^x)^r) \quad (9)$$

The encryption is completed because the EMBEnc method can accomplish with one of the x, y or r variables. While for the decryption x and y are required. The keywords will be encrypted in the same way. At the end, encrypted keywords and message are sent to the server.

$$C_{w_j} = H_2(e(g^x, H_1(W_j)^r)) \quad (10)$$

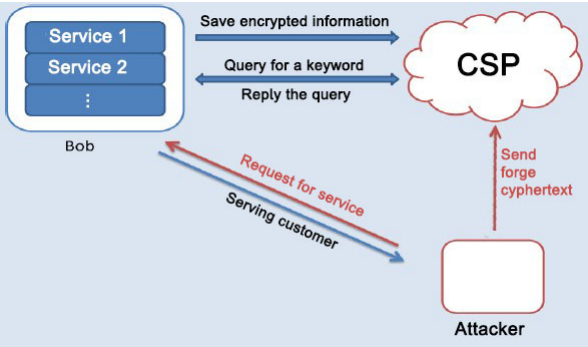

Fig. 2. Attacker forges Bob's signature

It can be seen that for the encryption, one of the variables x and r is enough.

B. The SPKS Improvement

As followed, one can produce a valid cypher text without using any private key; thus, the attacker can offer a valid cipher text which is not produced by the client. In order to solve this problem, we rewrite the following variables.

$$U_2 = p \bigoplus H_5(e(g^y, g^y)^{xr}) \tag{11}$$

$$C_{w_i} = H_2(e(g^y, H_1(w_i)^r)^x) \tag{12}$$

Other variables are not changed except the matching condition of trapdoors.

$$C_{w_i} == H_2(e(U_1, T_{w_j})^y) \tag{13}$$

It should be noticed that:

$$\begin{aligned} &C_{w_i} = H_2(e(g^y, H_1(W_i)^r)^x) = \\ &H_2(e(g^r, H_1(W_i)^x)^y) = H_2(e(g^r, T_{w_i})^y) \\ &= H_2(e(U_1, T_{w_i})^y) \end{aligned} \tag{14}$$

In Section 6, we show that by this modification the forgery problem would be solved. We call this modified scheme as Improved Secure Preserving Keyword searching (ISPKS).

C. The comparison between SPKS and ISPKS

As it is shown, the SPKS algorithm does not support user signature, i.e. the SPKS scheme could not warranty that the user is the only one who can create a cipher text correctly. Up to now, we have proposed a model in which the private key of the user or server is required to make a valid cipher text. In the Table I, the main advantages made over SPKS have been demonstrated.

TABLE I
COMPARISON BETWEEN SPKS AND ISPKS

	SPKS	ISPKS
Cipher text contains server signature	Yes	Yes
Confidentiality of keyword and data	Yes	Yes
User can decrypt the encrypted data without server collaboration	Yes	Yes
Cipher text contains user signature (third party resistant)	No	Yes

It can be seen that the ISPKS is resistant against the third party attacks in forging the cypher text.

IV. SEMANTIC SECURITY

In this part, we will propose a security proof for the ISPKS scheme. Here, the BDH problem is assumed to be NP hard. Suppose that H_1, H_2, H_3, and H_4, are random oracles.

Theorem 1: ISPKS is semantically secure assuming the BDH problem is NP hard. Let A be an IND-CPA[10] adversary consisting of two polynomial time algorithms A_1, and A_2. Let A_1 be an IND-CPA adversary that has the advantage ϵ_1 in breaking KWEnc. Suppose A_1 makes $q_T > 0$ trapdoor queries and $q_{H_2} > 0$ hash queries to H_2. Let A_2 be an IND-CPA adversary that has the advantage ϵ_2 against EMBEnc. Suppose A_2 makes $q_{H_4} > 0$ hash function queries to H_4. Let A be an IND-CPA adversary that has the advantage $\epsilon = \epsilon_1 + \epsilon_2$ against the ISPKS scheme. Then there exists an algorithm B that solves the BDH problem with the advantage at least: $ADVB(K) \geq 2\epsilon_1 / q_{H_2} + 2\epsilon_2 / q_{H_4}$. The running time of B is O(time(A)).

The ISPKS algorithm includes two public key encryption algorithms called KWEnc and EMBEnc. We prove the above theorem in two steps. KWEnc algorithm is semantically secure if the BDH problem holds.

[10]Indistinguishable under chosen plaintext attack

Lemma 1: Let A_1 be an IND-CPA adversary which has the advantage ϵ_1 in breaking the KWEnc algorithm. A_1 can make $q_T > 0$ query for trapdoor function and $q_{H_2} > 0$ query for the random oracle H_2. Then there exists an algorithm B_1 solving the BDH problem with the advantage of $2\epsilon_1 / q_{H_2}$ in O(time(A_1)).

Proof: let $< g, G_1, G_2, q >$ be the parameters of the BDH problem, where the order of sets G_1, and G_2, is q, and e is the bilinear map. Let $V_0 = g, V_1 = g^x, V_2 = g^y$, and $V_3 = g^r$, in which x, y, and r, are some random elements of Z_q^* and g is a random generator of set G_1. B_1 is given V_0, V_1, V_2, and V_3, as parameters. The solution of the BDH would be $D_1 = e(g,g)^{xyr}$.

H_1-queries: H_1 is initially an empty list, its elements are like $< W_i, h_i, a_i >$, where h_i is the value of $H_1(W_i)$ and a_i is a random value from Z_q^*. B_1 responds to the queries like $H_1(W_i)$ as follows:

1) B_1 tries to find W_i in the list. If W_i already exists, h_i will be returned.
2) Else a random number will be assigned to the variable a_i.
3) The value of h_i will be g^{a_i}.
4) At last B_1 adds $< W_i, h_i, a_i >$ to the H_1 list.

Note that in the mentioned list, there exists one h_i for a special W_i.

H_2-queries: H_2 is initially an empty list that its elements are like $< W_i, h_i >$. The value of $H_2(W_i)$ is h_i. Every time A_1 queries for $H_2(W_i)$, B_1 responds as follows:

1) B_1 tries to find W_i in the list. If it exists, proper h_i will be returned.
2) Else a random string from $\{0,1\}^{log\ q}$ will be assigned to the variable h_i.
3) At last B_1 adds $< W_i, h_i >$ to the H_2 list.

Phase 1: A_1 queries for trapdoor of W_i.B_1 initiates H_1 for W_i and computes the amount of $H_1(W_i)^x$ as follows.

$T_{w_i} = H_1(W_i)^x = h_i^x = (g^{a_i})^x = (g^x)^{a_i} = v_1^{a_i}$

Challenge: Once A_1 finished the first phase, A_1 sends W_0, and W_1, to B_1.

1) B_1 initializes H_1 for W_0, and W_1.
2) B_1 generates random $b \in \{0,1\}$ and selects a random string s_b from $\{0,1\}^{log\ q}$.
3) B_1 sends (V_3, s_b) to A_1.

It should be noted that:

$s_b = H_2(e(g^y, H_1(W_i)^x)^r) =$

$H_2(e(g^y, g^{xa_i})^r) = H_2(e(g,g)^{rxya_i})$

Guess: A_1 outputs its guess $b' \in \{0,1\}$ for b.

The solution of BDH problem: B_1 chooses one of the H_2 elements randomly. It is the correct result with the probability of $2\epsilon_1 / q_{H_2}$.

For comprehending of the guess, suppose V is the solution of the BDH problem. We will find the probability of choosing V. let Q be the event of selecting V, and $p[b = b']$ is the probability of finding valid answer. We know that in real attacks $p[b = b' | \neg Q] = 1/2$. But here, as A_1 has the advantage ϵ_1, then $p[b = b' | \neg Q] - 1/2 \geq \epsilon_1$ is true, meaning that it is not completely random.

$p[b = b'] = p[b = b' | \neg Q] \times pr[\neg Q] + p[b = b' | Q] \times p[Q] \leq 1/2\ p[\neg Q] + p[Q] = 1/2 + 1/2\ p[Q]$

$p[b = b'] \geq p[b = b' | \neg Q] \times p[\neg Q] = 1/2\ p[\neg Q] = 1/2(1 - p[Q]) = 1/2 - 1/2\ p[Q]$

Therefore, pr[Q]$\geq 2\epsilon_1$. A_1 queries for the correct result with the probability $2\epsilon_1$. If the correct result exists in the H_2 list, the probability of selecting the correct result would be q_{H_2}. Consequently, B_1 will find the correct result with the probability of $2\epsilon_1 / q_{H_2}$.

■

In the next step, the semantic security of EMBEnc is checked.

Lemma 2: Let A_2 be an IND-CPA adversary which has the advantage ϵ_2 in breaking the EMBEnc algorithm. A_2 can make $q_{H_4} > 0$ query for the random oracle H_4. Then, there is an algorithm B_2 that can solve the BDH problem with the advantage of $2\epsilon_2 / q_{H_4}$ in O(time(A_2)).

Proof: let $< g, G_1, G_2, q >$ be the parameters of the BDH problem, where the order of sets G_1, and G_2, is q, and r is the bilinear map. Let $V_0 = g, V_1 = g^x, V_2 = g^y$, and $V_3 = g^r$, where x, y, and r, are some random elements of Z_q^* and g is a random generator of set G_1. B_1 is given V_0, V_1, V_2, and V_3, as parameters. $D_1 = e(g,g)^{xyr}$ is the solution of the BDH problem.

H_3-queries: H_3 is initially an empty list, its elements are like $< W_i, h_i, a_i >$, where h_i is the value of $H_3(W_i)$ and a_i is a random value from Z_q^*. Every time A_2 queries for $H_3(W_i)$, B_2 responds as follows:

1) B_2 tries to find W_i in the list. If W_i already exists, h_i will be returned.
2) Else a random number will be assigned to the variable a_i.
3) The value of h_i will be g^{a_i}.
4) At last B_1 adds $< W_i, h_i, a >$ to the H_3 list.

Note that in the mentioned list for a special a_i there is just one h_i.

H_4-queries: H_4 is initially an empty list that its elements are like $< W_i, h_i >$. The value of $H_4(W_i)$ is h_i. B_2 responds to the queries like $H_4(W_i)$ as the following:

1) B_2 tries to find W_i in the list. If it exists, proper h_i will be returned.
2) Else a random string from $\{0,1\}^{log\ q}$ will be assigned to the variable h_i.
3) At last B_2 adds $< W_i, h_i >$ to the H_4 list.

Challenge: Once A_2 finished the phase one, it sends W_0, and W_1, to the B_2. B_1 generates random $b \in \{0,1\}$ and select a random string from $\{0,1\}^{log\ q}$ and sends (V_3, s_b) to the A_2.
Note that:
$s_b = H_2(e(g^y, H_1(W_i)^x)^r) =$
$H_2(e(g^y, g^{xa_b})^r) = H_2(e(g,g)^{xrya_b})$
Guess: A_2 outputs its guess $b' \in \{0,1\}$ for b. B_2 chooses one of the H_4 elements randomly. It is the correct result with the probability of $2\epsilon_2 / q_{H_4}$.
Similar to the proof of lemma 1, A_2 queries for the correct result with the probability of $2\epsilon_2$. If the correct result exists in the H_4 list, the probability of selecting correct result would be q_{H_4}. As a consequence, B_2 will find the correct result with the probability of $2\epsilon_2 / q_{H_4}$.

■

Proof of theorem: Base on the lemmas 1, and 2, it can be deducted that the interaction with an IND-CPA adversary can lead to solving the BDH problem.

V. Conclusion

In this paper, we introduced an improved version of SPKS algorithm (ISPKS). SPKS has a security bug, which cause an attacker to forge the encrypted message. The security of ISPKS encryption algorithm is enhanced to overcome the forging attack while the performance is kept to the equivalent schemes. Therefore, ISPKS is a qualified encryption algorithm for the purpose of providing confidentiality in cloud storage systems. The future work would be dedicated to build an encryption algorithm that is resistant against the server attacks.

VI. References

1) Kamara, Seny, Charalampos Papamanthou, and Tom Roeder. " *Cs2: A searchable cryptographic cloud storage system." Microsoft Research*, TechReport MSR-TR-2011-58 (2011).
2) Armbrust, Michael, et al. "*A view of cloud computing.*" Communications of the ACM 53.4 (2010): 50-58.
3) Chase, Melissa, and Seny Kamara. "*Structured encryption and controlled disclosure.*" Advances in Cryptology-ASIACRYPT 2010. Springer Berlin Heidelberg, 2010. 577-594.
4) Zunnurhain, Kazi, and Susan V. Vrbsky. "*Security in cloud computing.*" Proceedings of the 2011 International Conference on Security & Management. 2011.
5) Subashini, S., and V. Kavitha. "*A survey on security issues in service delivery models of cloud computing.*" Journal of Network and Computer Applications 34.1 (2011): 1-11.
6) Wang, Qian, et al. "*Enabling public verifiability and data dynamics for storage security in cloud computing.*" Computer SecurityESORICS 2009. Springer Berlin Heidelberg, 2009. 355-370.
7) Curtmola, Reza, et al. "*Searchable symmetric encryption: improved definitions and efficient constructions.*" Proceedings of the 13th ACM conference on Computer and communications security. ACM, 2006.
8) Liu, Qin, Guojun Wang, and Jie Wu. "*Secure and privacy preserving keyword searching for cloud storage services.*" Journal of network and computer applications 35.3 (2012): 927-933.
9) Chow, Richard, et al. "*Controlling data in the cloud: outsourcing computation without outsourcing control.*" Proceedings of the 2009 ACM workshop on Cloud computing security. ACM, 2009.
10) Vaquero, Luis M., et al. "*A break in the clouds: towards a cloud definition.*" ACM SIGCOMM Computer Communication Review 39.1 (2008): 50-55.
11) Song, Dawn Xiaoding, David Wagner, and Adrian Perrig. "*Practical techniques for searches on encrypted data.*" Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on. IEEE, 2000.
12) Boneh, Dan, et al. "*Public key encryption with keyword search.*" Advances in Cryptology-Eurocrypt 2004. Springer Berlin Heidelberg, 2004.
13) Chang, Yan-Cheng, and Michael Mitzenmacher. "*Privacy preserving keyword searches on remote encrypted data.*" Applied Cryptography and Network Security. Springer Berlin Heidelberg, 2005.
14) Hacigm, Hakan, et al. "*Executing SQL over encrypted data in the database-service-provider model.*" Proceedings of the 2002 ACM SIGMOD international conference on Management of data. ACM, 2002.
15) Liu, Qin, Guojun Wang, and Jie Wu. "*An efficient privacy preserving keyword search scheme in cloud computing.*" Computational Science and Engineering, 2009. CSE'09. International Conference on. Vol. 2. IEEE, 2009.
16) Shi, Elaine, et al. "*Multi-dimensional range query*

over encrypted data." Security and Privacy, 2007. SP'07. IEEE Symposium on. IEEE, 2007.

17) Zhu, Robert W., Guomin Yang, and Duncan S. Wong. "*An efficient identity-based key exchange protocol with KGS forward secrecy for low-power devices.*" Theoretical computer science 378.2 (2007): 198-207.
18) Boneh, Dan, and Matt Franklin. "*Identity-based encryption from the Weil pairing.*" Advances in CryptologyCRYPTO 2001. Springer Berlin Heidelberg, 2001.
19) Boneh, Dan, and Brent Waters. "*Conjunctive, subset, and range queries on encrypted data.*" Theory of cryptography. Springer Berlin Heidelberg, 2007. 535-554.
20) Ren, Kui, et al. "*A novel privacy preserving authentication and access control scheme for pervasive computing environments.*" Vehicular Technology, IEEE Transactions on 55.4 (2006): 1373-1384.
21) Bellare, Mihir, et al. "*Key-privacy in public-key encryption.*" Advances in CryptologyASIACRYPT 2001. Springer Berlin Heidelberg, 2001. 566-582.
22) Goyal, Amit, and Sara Dadizadeh. "*A survey on cloud computing.*" University of British Columbia Technical Report for CS 508 (2009): 55-58.
23) Fox, Armando, et al. "*Above the clouds: A Berkeley view of cloud computing.*" Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS 28 (2009).
24) Leavitt, Neal. "*Is cloud computing really ready for prime time.*" Growth 27.5 (2009).
25) Chen, Yanpei, Vern Paxson, and Randy H. Katz. "*Whats new about cloud computing security?.*" University of California, Berkeley Report No. UCB/EECS-2010-5 January 20.2010 (2010): 2010-5.
26) Diament, Theodore, et al. "*The dual receiver cryptosystem and its applications.*" Proceedings of the 11th ACM conference on Computer and communications security. ACM, 2004.
27) Zhang, Qi, Lu Cheng, and Raouf Boutaba. "*Cloud computing: state-of-the-art and research challenges.*" Journal of Internet Services and Applications 1.1 (2010): 7-18.
28) Rimal, Bhaskar Prasad, Eunmi Choi, and Ian Lumb. "*A taxonomy and survey of cloud computing systems.*" INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on. Ieee, 2009.
29) Chen, Yanpei, et al. "*Towards understanding cloud performance tradeoffs using statistical workload analysis and replay.*" University of California at Berkeley, Technical Report No. UCB/EECS-2010-81 (2010).