RSA Based Key Compromised Resistant Protocol(KCR) For Large Databases

Fatemah Alharbi and Huiping Guo

Abstract—Data communication and networking are essential in our daily lives. Companies rely on computer networks and internetworks to exchange information that is normally stored in large databases, with their customers. Nonetheless, it is not guaranteed that all networks are reliable; therefore, database content should be protected against any unauthorized access. One of the most powerful strategies that has been used in existing database security systems to protect databases is database encryption. Few of these systems are practical to be used with large databases since one important issue is not satisfactorily addressed which is concerning key management. It may take days to encrypt the huge databases. Imagine how the situation will be when a key is compromised! Straightforward solutions to address this problem demand that the keys used in database encryption to be replaced with new keys. Consequently, a database re-encryption process has to be executed. In this thesis, we propose "RSA-based Key Compromised Resistant Protocol (KCR)" to effectively address this problem.

Index Terms—cryptography; database encryption; database security; key management; secret sharing scheme

I. INTRODUCTION

IN the past, people had to exchange information physically back and forth, but nowadays we live in electronically connected world age. The invention of the Internet connects computer networks to serve people all over the world. The enormous amount of information resources available and ease of communication have made the Internet the most valuable tool in various settings of a person's life. However, the Internet is not considered a trustworthy network and the transmitted information is vulnerable to attacks which brings up the important issues of information security. Information security is the practice of protecting information and information systems against unauthorized access or alteration of information. It is a combination of data security and network security. Regarding data security, three security goals need to be achieved: confidentiality (hide data from attackers), integrity (keep data safe from unauthorized modifications), and availability (give authorized users access to their data) [1]. Along with these goals, there are five security services related to networking security: confidentiality, integrity, authentication, nonrepudiation, and authentication. Some of these services are for entities and messages sent or received by those

entities [2]. An optimal security application should efficiently implement these goals and services.

Other than information security, facilitating the access to information resources is a fundamental subject to make people's life much easier. Databases are used as storages to gather information and to effectively store a large number of records. While data should be available to legitimate users, database administrators are responsible for restricting the access privileges for protection against any kind of attacks. According to [3] and [4], such approach is one of the strategies of database security. Other strategies and methods have been studied for some time. Early work [5], [6] considers physical security, operating system security, Database Management System (DBMS) security and data encryption as the major strategies that support database security. Regarding physical security, this strategy excludes any remote access without legal permission to the database. Although making physical contact with database contents is an expensive measure, it partially ensures data integrity [5],[7]. On the subject of operating system security, according to [8], any operating system protection model is consisted of three elements: objects within the system, entities who access these objects, and regulations that manage how entities access objects. However, it is difficult to govern entities from disclosing these regulations to unwanted parties; this type of attack is called client colluding attack. The third strategy concerns DBMS security. Available DBMS security solutions assume the database grants the appropriate privileges to legitimate users and this obviously is not guaranteed against Trojan horse attacks [9]. All of these strategies by themselves do not completely satisfy the requirements of database security [5], [6]. The forth strategy that uses encryption as a technology is the practical solution to dominate database security [5], [6], [7], [8]. Many systems have been developed and evaluated to support database encryption. For example, the scheme in [10] proposes an encryption mechanism to provide security and improve query processing efficiency. Many other schemes have been studied to improve some encryption algorithms for various purposes. For example, privacy protection using the Chinese Remainder Theorem [11] has been examined (e.g. [5], [6]). The studies are mainly based on encryption and anonymizing. Regarding key management for encipherment systems, strategies such as key division ciphers and commutative ciphers (e.g. [12] and [13]) have been implemented.

Although these techniques and methods can improve the encryption and decryption procedures, none of them provides a key management system that prevents the re-encryption process of large databases when a key is compromised. For example, consider a massive organization - industrial, government, or military - where all employees

(referred to as users) are granted the appropriate privileges to access the organization's database that contains thousands of records. We assume that a modern Public Key Infrastructure (PKI) is available and all users are capable to perform their requests by applying the appropriate encryption techniques. Now suppose that a key pair (public key and private key) is generated and that the public key is used to encrypt the huge database and the private key, which is shared among all users in the organization, is used for decryption. Presume that an unauthorized user breaks the organization's security firewall and gains access to the decryption key. All database contents would be disclosed. One possible approach is to periodically change the database encryption and decryption keys, but this strategy addresses two main drawbacks. First, the overhead caused by frequently changing the keys is overkill. The second drawback is the re-encryption process of large databases that requires time and a lot of resources. To the best of my knowledge, there is no efficient and practical solution to such a problem.

In this paper, we propose a novel key management protocol called "RSA-based Key Compromised Resistant protocol", referred to as KCR. The primary objective of this protocol is to effectively avoid re-encrypting large databases in case their keys have been compromised. KCR protocol consists of a set of strategies where the protocol operates practically. The protocol has been implemented and some experiments have been conducted for the purposes of evaluation. Simulation results show that my proposed protocol is works well.

I. KCR PROTOCOL

A. General description

In this study, the focus is on key management for database encryption. The essential objective of KCR protocol is to solve the problem of re-encrypting a large database when its decryption key is compromised. Before I demonstrate the design details of KCR and how this protocol works, there are some assumptions and general explanations that should be taken into account:

- KCR protocol uses one key for encryption and a set of keys for decryption. When any of the decryption keys is compromised, it is discarded and replaced with another decryption key without the need of re-encrypting the large database.
- Clients are grouped. Each group is granted a decryption key that is going to be used by all clients in the group. When a key is compromised, any of the group members contacts the responsible agency then gets a new decryption key for her/his group. The protocol assumes that the groups are predefined and the number of groups is less than the number of decryption keys.
- Distribute control in the context of decryption keys among different parties is proposed as a solution to strengthen the security measures of large database contents. The primary cryptographic algorithm of KCR protocol is RSA which demands two keys: public key and private key. The public key is used for database encryption while the private key is used for database decryption. The private key is divided into a set of key pairs in which the product of the two entries of each key

pair equals the original private key; in other words, the entries represent factors of the original private key. These entries are distributed between the database server and the groups.
- To decrypt a database object, the workload of the decryption process is distributed between the client who wants to access the object and the database server.
- The protocol provides secure communications between involved parties.

B. KCR Encryption and Decryption

Essentially, KCR protocol is based on RSA cryptosystem which consists of three processes: keys generation, encryption, and decryption. The proposed protocol adds more features to these processes.

Keys Generation. The public and private keys are generated as follows. First, a server selects two large primes, P and Q. Then, it computes $N = PQ$ and the totient function of N which is $\phi(N) = (P-1)(Q-1)$. The server selects randomly an integer number E such that the greatest common devisor of D and $\phi(N)$ equals 1. An integer number D is calculated as follows: $D = E^{-1} \bmod \phi(N)$. Now, the keys are ready to be used, the public key is (E, N) and the private key is (D, N). The server computes the factors of D: $F_1, F_2, F_3, ..., F_n$. Then, it generates a set of key pairs $\{(D_1, D_{1'}), (D_2, D_{2'}), (D_3, D_{3'}), ..., (D_n, D_{n'})\}$ in which $D = D_1 D_{1'} = D_2 D_{2'} = D_3 D_{3'} = ... = D_n D_{n'}$.

Encryption. A plaintext P is encrypted to generate a ciphertext C using the public key (E, N) as follows:

$$C = P^E \bmod N$$

Decryption. To get the original plaintext P, partial decryption is done on the ciphertext C using the key pair $(D_i, D_{i'})$ and the modulus N:

$$C' = C^{D_i} \bmod N$$

$$P = (C')^{D_{i'}} \bmod N$$

In case the key pair $(D_i, D_{i'})$ is compromised, it must be discarded and replaced by a new key pair $(D_j, D_{j'})$. To get P using the new key pair, the partial decryption process is as follows:

$$C' = C^{D_j} \bmod N$$

$$P = (C')^{D_j} \bmod N$$

Proof of KCR. The following proves that encryption and decryption are inverses of each other:
We want to prove that: $P = C^D \bmod N$ where $D = D_i D_{i'}$

$$P = (C')^{D_{i'}} \bmod N$$

$$= (C^{D_i})^{D_{i'}} \bmod N \qquad // \ C' = C^{D_i} \bmod N$$

$$= (C)^{D_i D_{i'}} \bmod N \qquad // \ ((X)^y)^z = (X)^{yz}$$

$$= C^D \bmod N \qquad // \ D = D_i D_{i'}$$

A summary of all notations used in KCR encryption and decryption procedures is shown in Table I.

Table 1: NOTATIONS USED IN KCR ENCRYPTION AND DECRYPTION

P, Q	Two large primes
N	Modulus
(E, N)	Public key
(D, N)	Private key
F_i	The ith private key factor
$(D_i, D_{i'})$	The ith private key pair
D_i	The first entry of the ith key pair
$D_{i'}$	The second entry of the ith key pair
P	Plaintext
C	Fully decrypted ciphertext
C'	Partially decrypted ciphertext

C. Design

The design architecture of KCR protocol comprises four components: Keys server, KDC server, Database server, and Client.

Components. The basic idea that explains the tasks of each component is as follows:

1) Keys server: Keys server is responsible for KCR keys generation process. It calculates the RSA public key *(E, N)* and private key *(D, N)* . When this process is accomplished, the server sends the public key *(E, N)* to the Database server. In addition, it stores the private key pairs *{(D₁, D₁'), (D₂, D₂'), (D₃, D₃'), ..., (Dₙ, Dₙ')}* in its database, and it maintains all information regarding these pairs. An important information that Keys server should store in its database is key pairs status K_{status}. The initial value of key pairs status is "inactive". When a key is granted, the status value is changed to "active". When a key is compromised, the status is replaced with the value "compromised". In addition to these tasks, Keys server is primarily responsible of the key distribution process. When a key pair first entry D_i is granted to a group G_i , Keys server forwards the key pair second entry $D_{i'}$ to the Database server along with information about the group which owns the key.

2) KDC server: KDC forms groups, which are predefined, for clients then it distributes those clients (as members) among the different groups according to predefined rules. The server contains a database to store information of the groups $(G_1, G_2, G_3, ..., G_n)$. The following example illustrates this approach. In a dentists clinic, suppose there are *d* dentists, *a* dentist assistants, and *p* patients. So, the groups are dentist, dentist assistant and patient, and the total number of clients is *d+a+p* in which each one of the clients is a member of her/his correspondent group. Other than forming groups, KDC is needed to manage the interaction between groups members and the Keys server. It acts as a trusted agency to send confidential messages from members to Keys server and vise versa. Also, this server is used to associate each group with a key. It establishes a connection with the Keys server to obtain the first entry of key pairs *(D₁, D₂, D₃, ... , Dₙ)* and grants each group a key.

3) Client: When a client wants to process a query to access a database object, she/he needs to join a group in the first place. She/he contacts the KDC server to join a group G_i. Then, the KDC sends a member ID M_{i}_ID and the key D_i to the client. I assume that the group IDs are published. To process a query Q_i , the client sends the ID of group G_i and Q_i to the Database server. After receiving the partially decrypted object from the Database server, the client finishes the decryption process by using her/his group's key D_i. Moreover, the client is responsible to inform KDC when the key is compromised. Consequently, KDC contacts Keys server, then the latter server updates the information stored in its database regarding the compromised key. After that, Keys server gives a new decryption key ,if available, to the group. Accordingly, Keys server forwards these updates to the Database server.

4) Database server: This server contains the main database that we need to encrypt. The encipherment is done using the RSA public key received from Keys server. In addition to this task, the server plays an important role in the decryption process. When a client sends a query Q_i to Database server along with other information to identify the group she/he belongs to, the server uses the appropriate key received from Keys server $D_{i'}$ and partially decrypts the object and sends the result to the client. Then, the client completes the decryption process by decrypting the received result using her/his key D_i.

Scenarios. As aforementioned, KCR protocol is based on the interaction between four essential components: Keys server, KDC server, Database server and Client. These interactions define five important scenarios: Public key distribution scenario, private key distribution in normal case scenario, join group scenario, private key distribution in case a key is compromised scenario, access database scenario. The following presents details of each scenario (Hint: *M, M', E( ), D( ),* and *Hash( )* denotes to transmitted message, encrypted message, encryption function, decryption function, and hash function, respectively).

- **Scenario 1 - Public key distribution:**
  After generating the key pairs (public and private key), Keys server sends a message that includes the public key to Database server. The message, *M = (E, N)*, is encrypted using a secret session key K_L between Keys server and Database server: *M' = E(M, K_L)*. Upon

receiving the encrypted message, Database server decrypts it with K_L: $M = D(M', K_L)$ to get the public key and encrypts the database according to the KCR encryption process discussed earlier. The design architecture of this scenario is shown in Figure 1. Table II shows the notations used in the scenario.
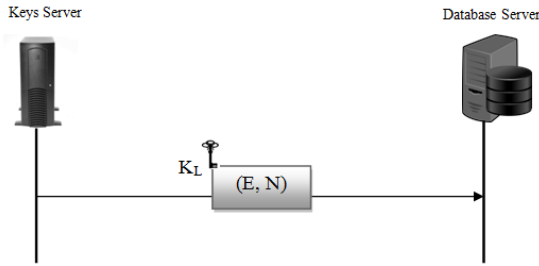

Figure 1: KCR protocol: Scenario 1 - Public key distribution

Figure 2. KCR protocol: Scenario 2 - Private key distribution in normal case

TABLE II
NOTATIONS USED IN SCENARIO 1

K_L	Encrypted with Keys-Database secret key
(E, N)	Public key for database encryption

- **Scenario 2 - Private key distribution in normal case:**
  This scenario consists of three steps:
  1) To grant a decryption key to a group G_i, the KDC creates a message that contains the group ID G_i_ID and the key status, $M = (G_i\_ID, K_{status})$. The key status is "inactive" to indicate that group G_i is a new group and does not own a key yet. The message is sent to Keys server. The message is encrypted using a secret session key K_M between KDC server and Keys server: $M' = E(M, K_M)$.
  2) When Keys server receives the encrypted message M', it decrypts it: $M = D(M', K_M)$ to get G_i_ID and K_{status}. Since K_{status} = "inactive", Keys server scans its database to extract a key pair $(D_i, D_{i'})$ then grants this key pair to the group G_i . Keys server updates the value of the key pair status which is stored in its database from "inactive" to "active". The server then replies to KDC by sending a message that contains the first entry of the private key pair (N is part of the private key); $M =(D_i, N)$. For confidentiality, the message is encrypted using the secret session key K_M as follows: $M' = E(M, K_M)$.
  3) Keys server forwards a message to Database server. The message includes the second entry of the private key pair, G_i_ID, and the hash value of the first entry of the private key; $M = ((D_{i'}, N), G_i\_ID, Hash(D_i))$. The message is encrypted with a secret session key K_L shared between Keys server and Database server. The format of the message is as follows: $M' = E(M, K_L)$.

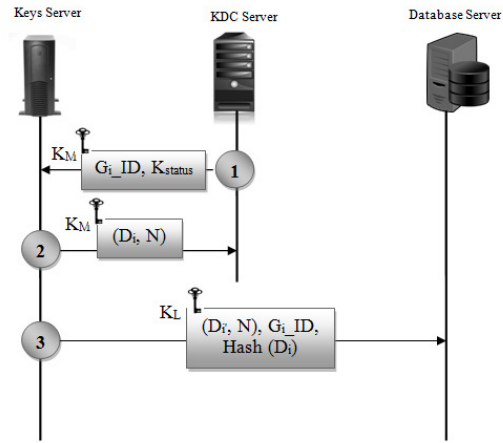 Figure 2 shows the scenario steps. Table III summarizes the notations used in the scenario.

TABLE III
NOTATIONS USED IN SCENARIO 2

K_L	Encrypted with Keys-Database secret key
K_M	Encrypted with KDC-Keys secret key
G_i	The ith group
G_i_ID	The ID of group G_i
K_{status}	Key status (active, inactive, compromised)
(D$_i$, N)	The ith first entry of the ith private key pair
(D$_{i'}$, N)	The ith second entry of the ith private key pair
Hash(D$_i$)	Hash value of D$_i$

- **Scenario 3 - Join group.** The following steps explain how a client joins a group:

  1) A client sends a short message that is encrypted by a secret session key K_N between her/him and the KDC server. The message contains the group ID G_i_ID of the group G_i that she/he wants to join, and it is encrypted as follows: $M' = E(G_i\_ID, K_N)$.
  2) KDC server decrypts the message as follows: $M = D(M', K_N)$ to get G_i_ID. Then, it adds the client as a member M_i to group G_i and responds by sending the member ID M_i_ID and the group's key; $M =(M_i\_ID,(D_i, N))$. The message is sent as an encrypted message: $M' = E(M, K_N)$. Then the client decrypts the message: $M = D(M', K_N)$.

  The steps are shown in Figure 3. The notations used in the scenario are summarized in Table IV.

- **Scenario 4 - Private key distribution in case a key is compromised.** The heart of the KCR protocol is handling the situation when a large database decryption key is compromised. This scenario presents the steps that explain the interaction between the involved components:
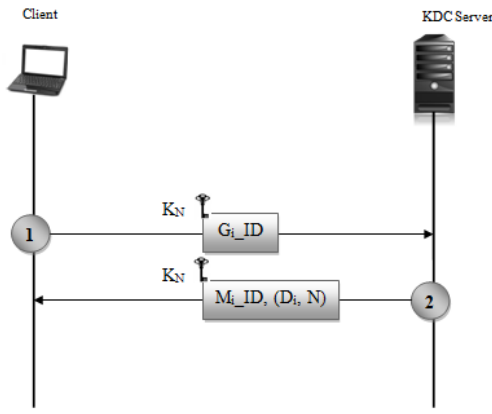
TABLE IV
NOTATIONS USED IN SCENARIO 3

K_N	Encrypted with Client-KDC secret key
G_i	The ith group
G_i_ID	The ID of group G_i
(D_i, N)	The ith first entry of the ith private key pair
M_i	The ith member of group G_i
M_i_ID	The ID of member M_i
(D_i, N)	The ith first entry of the ith private key pair

1) When a private key pair $(D_i, D_{i'})$ of a group G_i has been attacked, group members are responsible of informing the KDC about the situation. A client, a group member, sends a message to the KDC server that contains G_i_ID and K_{status} = "compromised" : $M = (G_i\_ID, K_{status})$. The message is encrypted using a secret session key K_N between the client and KDC server: $M' = E(M, K_N)$.

2) When KDC receives the encrypted message M, it decrypts it: $M = D(M', K_N)$ to get G_i_ID and K_{status}. Since K_{status} = "compromised", KDC forwards the message to Keys server to get a new key for group G_i. The message is encrypted with K_M, a secret key shared between KDC and Keys servers: $M' = E(M, K_M)$.

3) Then, Keys server decrypts message M': $M = D(M', K_M)$ and updates its database that contains information about group G_i and its key pair $(D_i, D_{i'}$; it changes the value of the key pair status from "active" to "compromised". The server then grants another key pair $(D_j, D_{j'})$ to group G_i. It sends a message containing the first entry D_j of the new key pair as a private key $(M = (D_j, N))$ to KDC after encrypting it using K_M : $M' = E(M, K_M)$.

4) After receiving the message from Keys server, KDC decrypts it: $M = D(M', K_M)$ to get the key (D_j, N) then grants the new decryption key to group G_i. Next, the server sends an enciphered message using K_N to the client. The message includes the new key D_j: $M' = E(M, K_N)$.

5) Keys server sends the second entry $D_{j'}$ to Database server. The message $(M = (D_{j'}, N), G_i\_ID, Hash(D_j)))$ is encrypted by K_L: $M' = E(M, K_L)$. Consequently, Database server discards the old key of group G_i and exchanges it with the new one.

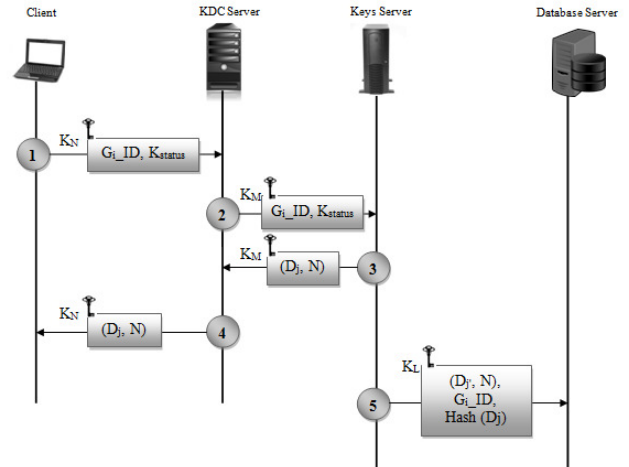The steps of this scenario are shown in Figure.4 and the notations used are summarized in Table V.

Figure 4. KCR protocol: Scenario 4 - Private key distribution in case a key is compromised

TABLE V
NOTATIONS USED IN SCENARIO 4

K_L	Encrypted with Keys-Database secret key
K_M	Encrypted with KDC-Keys secret key
K_N	Encrypted with Client-KDC secret key
G_i	The ith group
G_i_ID	The ID of group G_i
K_{status}	Key status (active, inactive, compromised)
(D_j, N)	The jth first entry of the jth private key pair
$(D_{j'}, N)$	The jth second entry of the jth private key pair
$Hash(D_j)$	Hash value of D_j

- **Scenario 5 - Access database.** The scenario steps are explained as follows:

1) To process a query Q_i, a client contacts the Database server and sends a message that contains Q_i along with an additional information that helps the Database server indicate which key to use. This information is either the group ID G_i_ID or the hash value of the group's key first entry D_i. If the client chooses G_i_ID, the message content is $M = (Q_i, G_i\_ID)$. If she/he forgets her/his group ID, she/he can still access the database using the hash
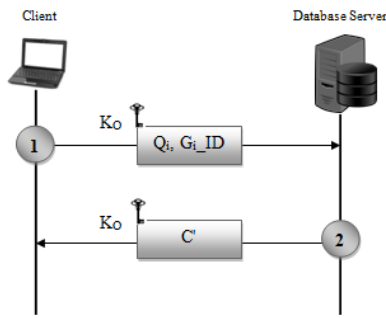
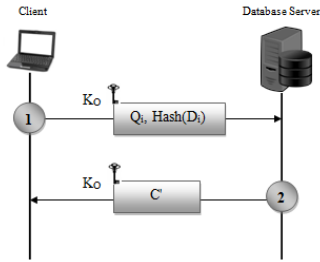Figure 5. KCR protocol: Scenario 5 - Database access with group ID choice

Figure 6. KCR protocol: Scenario 5 - Database access with hash value choice

value of her/his key. In this case, the message content is $M = (Q_i, Hash(D_{i}))$. A secret session key K_O is used to encrypt the transmitted message: $M' = E(M, K_O)$.

2) Upon receiving the query, Database server decrypts the message: $M = D(M', K_O)$, processes the query Q_i, and extracts the second entry $D_{i'}$ of the private key pair. Then, it decrypts the encrypted query result CR_i to generate a partially decrypted ciphertext: $C' = (CR_i)^D \mod N$. Database server sends an encrypted message to the client using K_O. The message includes the partially decrypted result C': $M' = E(C', K_O)$. When the client receives the message, she/he decrypts it: $M = D(M', K_O)$ to get C' then completes the decryption process to get the original Result R_i: $R_i = (C')^D \mod N$.

Figure .5 shows the steps of database access scenario with G_{i}_ID choice.

Figure 6 shows the steps of the scenario with hash value choice. Table VI summarizes the notations used in this scenario.

D. Conclusions

In this paper, we propose a novel key management system. Some features could be added to make the implemented project more efficient. Besides the tasks that KDC server is responsible for, KDC could manage adding and deleting groups. In this case, it is not necessary that groups be predefined. The same could be applied to members. Another feature might be generating public and private keys automatically in case Keys server is out of key pairs. Accordingly, all groups members are notified by email that their keys are compromised so they need to contact KDC to get their new keys.

TABLE VI
NOTATIONS USED IN SCENARIO 5

K_O 🔑	Encrypted with client-Database secret key
G_i	The ith group
Q_i	The ith client requested query
(D_i, N)	The ith first entry of the ith private key pair
$(D_{i'}, N)$	The ith second entry of the ith private key pair
$Hash(D_{i})$	Hash value of D_i
CR_i	The ith encrypted result of query Q_i
R_i	The ith result of query Q_i after decryption
C'	The partially decrypted cipher text

In applications where the performance of encrypting database is not satisfactory, the proposed scheme can be easily extended to improve the performance. Instead of encrypting the database using RSA, we can use any modern block cipher such as 3DES or AES to encrypt databases, and then use the proposed scheme to protect the encryption key.

REFERENCES

[1] B. A. Forouzan, Data communications and networking. New York: The McGraw-Hill Companies, 2007.
[2] B. Forouzan, Cryptography and network security. New York: The McGraw-Hill Companies, 2008.
[3] M. C. Murray, "Database security: What students need to know," Journal of Information Technology Education: Innovations in Practice, vol. 9, 2010, http://www.jite.org/documents/Vol9/JITEv9IIPp061-077Murray804.pdf.
[4] H. Kayarkar, Classification of various security techniques in databases and their comparative analysis, ACTA Technica Corviniensis-Bulltin of Angineering, Report ISSN 2067-3809, April-June, 2012, http://acta.fih.upt.ro/pdf/2012-2/ACTA-2012-2-25.pdf.
[5] G. I. Davida, D. L. Wells and J. B. Kam. "A Database encryption system with subkeys," Journal of ACM Transactions on Database Systems (TODS), vol. 6, no. 2, pp. 312-328, June 1981.
[6] M.-S. Hwang and W.-P. Yang. " Multilevel secure database encryption with subkeys," Data & Knowledge Engineering, vol. 22, pp 117-131, April 1997.
[7] J. A. Cooper, Computer and communications security: Strategies for the 1990s. New York: McGraw-Hill, 1989.
[8] G.S. Graham and P.J. Denning, "Protection-principles and practice," Joint Computer Conference , vol. 40, pp. 417-429, 1972.
[9] P. A. Dwyer, G. D. Jelatis, and B. M. Thuraisingham. "Multilevel security in database management systems." In Computers and Security, vol. 6, no. 3, pp. 252, June 1987.
[10] W. Zhao, D. - F. Zhao, F. Gao, and G. - H. Liu. "A Cryptography index technology and method to measure information disclosure in the DAS model," Journal of WSEAS Transactions on Information Science and Applications, vol. 6, no. 9, pp. 1443-1452, Sept. 2009.
[11] J. Grosschadl. "The Chinese reminder theorem and its application in a high-speed RSA crypto chip," in Proceedings of the 16th Annual Computer Security Application Conference, 2000, pp. 384-393.
[12] S. Chen, S. Chen, H. Guo, B. Shen, and S. Jadjodia, "Efficient proxy-based Internet media distribution control and privacy protection infrastructure," in Proceedings of the 14th IEEE International Workshop on Quality of Service, New Haven, CT, 2006, pp. 209-218.
[13] M. Malkin, T. Wu, and D. Boneh, "Experimenting with shared generation of RSA keys," in Proceedings of the Internet Society's 1999 Symposium on Networking and Distributed System Security (NDSS), 1999, pp. 43-56.