Information Sciences Letters
An International Journal

http://dx.doi.org/10.12785/isl/020303

Design and implementation of multi-layer policies for database security

Ayman Mohamed Mostafa[1,*], *Mohamed Hashem Abdel-Aziz*[2] *and Ibrahim Mahmoud El-Henawy*[1]

[1] Faculty of Computers and Informatics, Zagazig University, Zagazig, Egypt
[2] Faculty of Computers and Informatics, Ain Shams University, Cairo, Egypt

Received: 28 Apr. 2013, Revised: 20 Jul. 2013, Accepted: 23 Jul. 2013
Published online: 1 Sep. 2013

Abstract: The security of database depends on a set of systems, roles, procedures, and processes that protect the entire database from unintended activities. Unintended activities can be categorized as authenticated misuse, malicious attacks or inadvertent mistakes made by authorized users. If any intruder succeeds in attacking the system network, the database security will be the last line of defense in protecting confidentiality, availability, and integrity. This paper presents interactive multi-layer policies for securing relational database that lies on the server side, monitor authorized users who may misuse their privileges on the client side, and monitor database administrators who may use their multiple privileges to penetrate the security system. These multi-layer policies can be combined together to create a defense system that puts the intruder under pressure at all security levels in order to protect integrity and confidentiality of database.

Keywords: Database Security, Network Security, Multi-Layer Policies, and Defense System

1 Introduction

The security process in any organization depends mainly on network security, physical security, and database security. The network security is considered as the outer security layer for protecting the system from external intruders. The main backbone for securing the system network is by using intrusion detection systems (IDS), firewalls and by using secure socket layer (SSL) and transport layer security (TLS). Physical security provides a kind of access control measures to deny access to unauthorized users. It can be used also to distinguish between authorized and unauthorized users, detects intrusions, and triggers appropriate responses with respect to attacks.

All these security measures will be useless if the attacks on the system were from inside. The inside intruders can misuse their legitimate privileges to attack confidential information inside database system. So, database security is considered as the last line of defense for securing database from external attackers and internal intruders.

The U.S. security magazine ”SECURE CYBERSPACE” survey stated that, 89% of the users install a firewall, 60% of the users install intrusion detection systems, but there are still 90% of the user’s system security has been damaged [1]. Classical database security relies on many different mechanisms and techniques, including access control, information flow control, network security, prevention of statistical inference, data and user authentication, data encryption, time-stamping, digital signatures, and other cryptographic mechanisms and protocols [2].

The main idea of this paper is to present different database security policies based on access control mechanisms, cryptography techniques, and intrusion detection systems. These policies are combined together in a harmonious behavior to build multi-layer security policies for protecting database from both legitimate users and database administrators.

The contribution of this paper has the following features:

–Provides security architecture for securing relational database based on multi-layer security policies.
–The multi-layer policies can be used for securing relational database that lies on the server side and

* Corresponding author e-mail: eng_ayman_m@hotmail.com

monitor authorized users who may misuse their privileges on the client side.
–The multi-layer policies can monitor database administrators who may use their multiple privileges to penetrate the security system by building a super administrator that can trace any attack operations from any administrator who are connected through the system session.
–Reduces the overhead on database administrators by providing controlled privileges to database users based on the concept of separation of duty (SoD).

2 Related Work

Database security applications depend on three main dimensions to provide a strong security system. These dimensions are known as: Confidentiality, Integrity, and Availability. Data confidentiality refers to the protection of database from unauthorized disclosure and is enhanced by using different database cryptographic techniques. Data integrity refers to the prevention of unauthorized users from modifying data and is enhanced by using access control techniques. Data availability refers to the capability of data to be available on the web even if there is a system crash and is enhanced by machine learning techniques [3].

The integrity of database can be maintained by using access control mechanisms. Access control is the process of mediating every request to resources and data maintained by a system to determine whether this request should be granted or denied. One of the recent researches in access control is to build hierarchical access control on database [4]. It proposes automated design method to reduce the number of operations for user data spaces creation. Leon Pan [5] presents a novel criterion-based access control approach to deal with multilevel database security. In this approach, authorization rules are transformed to security criteria, security criterion expressions, and security criterion subsets. Security criterion expressions are associated with sub-objects to serve as lock, and security criteria are associated with users to serve as keys. Access control mechanisms are used to grant access to some portion of data while the remaining data kept private. Shyni et al [6] provides a model for privacy protection in object relational database. The key feature of this model is that it allows multiple purposes to be associated with each data element and also supports explicit prohibitions. Access control mechanisms can be conducted also to web databases in a model called **RBAC+**. The central idea of RBAC+ includes the concepts of application, application profile and sub-application session when controlling the access to web databases [7].

The confidentiality of database is enhanced by using different cryptographic techniques that are applied to data when it is stored on secondary storage or transmitted on a network. Recently, the use of encryption techniques has gained a lot of interest in the context of outsourced data management. One of the recent achievements in encrypting database is to provide mixed cryptography operation over database [12]. It proposes a framework in order to encrypt databases over un-trusted networks in a mixed form using many keys owned by different parties. The proposed framework is very useful in strengthening the protection of sensitive data even if the database server is attacked at multiple points from the inside or outside. There are also different researches that have been targeted on implementing cryptography algorithms for securing database as presented in [13,14,15,16]. There are different policies have been presented for securing database. The authors in [17] implement different policies based on different types of access control mechanisms. The authors in [18] provide a study on different types of access control such as discretionary, mandatory, and role-based access control. This paper provided advantages and disadvantages on these access control types on different applications.

All previous researches focused only on securing database from external and internal users without providing a way for protecting confidentiality and integrity of database from database administrators. This paper presents a comprehensive integration between access control mechanisms, cryptography techniques and intrusion detection systems for providing multi-layer policies for protecting database. These multi-layer policies monitor any intrusive behavior inside database and make the intruder (user or database administrator) under pressure at all levels of security policies.

This pressure is made by each policy that presents a security point to protect database from a particular dimension but each single policy can be penetrated by any malicious database administrator or system user. This problem can be solved by protecting each single policy with the existence of another policy that provides a second layer of security and so on.

The interconnected policies are combined together in a harmonious behavior to provide a final database security system based on multi-layer policies.

3 Security Architecture

Unlike database security frameworks that exist today, which mostly detect imminent problems, generate an alert, and produce a report, this architecture provides different database security policies that protects database from internal intruders while keeping the database in a consistent state.

The database security architecture depends on the super administrator (**SA**) who builds a number of **N** database administrators who can connect to the database. The super administrator (**SA**) also determines a number of **K** shadows which are the minimum number of database administrators (**DBAs**) who can manage and control any transactions through the database server. A

checking procedure is executed to determine if the number of **K DBAs** is more than the number of **N DBAs**, then a system error will rise to put the parameters again such that the number of **N K** shadows must be less than or equal to the number of **N DBAs**. Each policy will be explained in detail in the following section.

4 Multi-Layer Security Policies

4.1 Super Admin Initiation Policy

Most database security researches are based on database administrators (**DBAs**) who can control and manage database transactions [8,7,3,19] but the most challenging problem is that the database admin may be the intruder who can penetrate the database system resources based on his wide privileges. So, a super admin (**SA**) must be found to monitor all database administrators and control their privileges.

The first policy starts by building a super admin (**SA**) who manages and monitors all database administrators, their users, and all data transmissions through database server. If there are any intruders inside database whether they are database administrators (**DBAs**) or legitimate users, the super administrator will have the ability to suspend or disconnect this malicious behavior. So, the super admin is considered as a trusted third party (**TTP**) that can be the manager of the organization or the owner or creator of database. Once the super admin is created, he starts building a number of **K DBAs** shadows by using Lagrange interpolation polynomial secret key sharing algorithm. The super admin determines the total number of **DBAs** that are connected to the database server (**N DBAs**), and the number of shadows (**K DBAs**). As discussed in the previous section, the number of **K** shadows is the minimum number of **DBAs** who can manage any transaction through database server. Lagrange interpolation polynomial secret key sharing algorithm is presented as follows:

–If the database scheme contains **N DBAs** such that any **K DBAs** shadows can combine together to control the transmission process, the **DBMS** generates the following polynomial equation:

$$F(x) = (ax^{k-1} + bx^{k-2} + cx^{k-3} + \cdots + nx^{0}) \mod p \quad (1)$$

Where (**a**, **b**, **c**,. . . , **n**) are random coefficients and **p** is a prime number. These parameters are inserted into the system by the super admin (**SA**). Any **K** shadows can be used to create **K** equations by evaluating the polynomial at **n** different points such that:

$$k_i = F(x_i) \quad (2)$$

These secret keys must be kept private in a smart card such that any **K** DBAs out of **N** DBAs can make a decision for any policy modification.

Another important operation here is that the super admin (**SA**) is only has the ability to classify database users who are connect to the system from the client side. The super admin (**SA**) provides the maximum number of operations (insert-update-delete-select) allowed for each user to differentiate between ***active***, ***intermediate*** and ***inactive*** users. For example, the super administrator fills in the parameters for (**Insert**) operation as follows **Insert**: Max (20), Active (15), Intermediate (8), Inactive (1). This means that any active user can perform between 15 and 20 insert operations. Intermediate users can execute insert operations ≥ 8 and < 15. Inactive users can execute insert operations ≥ 1 and < 8.

4.2 User Profile Setup Policy

This policy represents the preparation of system administrators to control and monitor data transmission of users. In this stage, each database administrator from a number of **K** DBAs uses the privileges granted to him from the super admin (Policy 4.1) to enter his users profiles, creating database roles, and granting each role to a set of database users.

Once the database administrator fills in these parameters, a request is sent to all **K DBAs** shadows whether to grant or deny this request. The importance of classifying users according to their activities on the system arises to determine the suitable response action that must be taken to stop the anomalous behavior from harming the system.

The research in [8] presents a response action for intrusion behavior when the intruder is the DBA but it eliminates how to control malicious users and malicious DBA at the same time. In addition, the research in [8] provides a security policy for controlling DBAs in large organizations but most organizations architectures are small or medium. This means that there will be a small number of DBAs or at most only one DBA, but there will be a large number of users inside the organization. As a result, there is a dire need to give the users of the organization some controlled degree of freedom to perform their actions.

This paper presents a complete vision of how to secure the database system from inside intruders whether they are legitimate users or DBAs. Different database systems that depend mainly on modeling user profiles result in a large false alarm rate [7]. So, Policy 4.3 provides a third layer of security to control user profiles is presented.

4.3 Secret Sharing RBAC Policy

Role-based access control (RBAC) models represent arguably the most important recent innovation in access

control models. Under an RBAC model, all authorizations needed to perform a given activity are granted to the role associated with that activity, rather than being granted directly to users [3].

The main problem is that if one of the users inside the same role is a malicious user, then he will have a full access to system resources granted to him as illustrated in Figure 1.

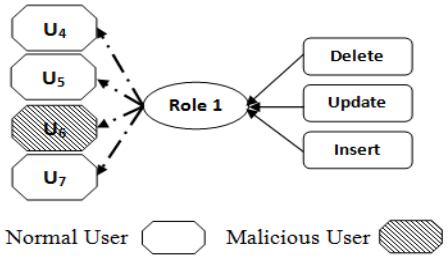

Fig. 1: Normal and Malicious Users inside RBAC

As presented in Figure 1, the malicious user (U_6) can perform any operation inside database depending on his granted role and this will increase the false positive operations. The simplest way to avoid this problem is to give each user a specific role and specific operations inside database that he can not deviate from it but this will generate a great overhead on database administrators, overhead on the remote server, and the size of database will be increased.

This paper presents a novel policy to create a RBAC for different users and then provide a secret authentication parameter inside the same role for each user connects to the remote server such that each user who wants to perform a specified action inside database must share with other users inside the same role according to his operation priority (insert update delete select) to provide a final signature to allow the transmission to be completed.

As presented in Figure 2, the malicious user (U_6) must pass through the authentication wall that resides on the remote server to provide an authentication parameter in order to proceed to his operation. This policy presents a strong layer of security to monitor user behavior inside the same role without the intervention of DBAs to manage and control the operation. So, there will be no overhead on database administrators or on remote server.

The authentication parameter that must be provided is based on the priority of DML operations for each user inside the same role. As presented in Table 1, the super admin (**SA**) or a number of **K** DBAs will generate different roles and pass these roles to different users. The second step is to provide priority for each DML operation the user can manipulate inside the same role.

Based on Table 1, if the malicious user (U_6) wants to perform (Insert) operation, then this will be his 3rd priority. He must provide an *authentication parameter*

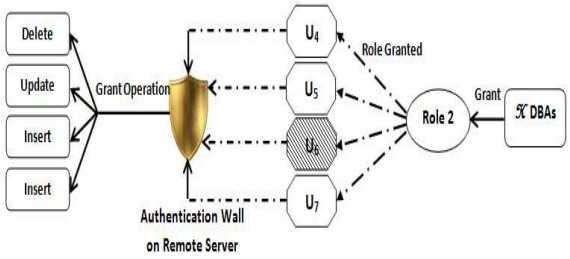

Fig. 2: Secret Sharing RBAC

Table 1: Operations Priority

Role	User	Priority of Operation			
		Priority 1	Priority 2	Priority 3	Priority 4
Role 1	U_4	Delete	Update	Select	Insert
	U_5	Select	Insert	Delete	Update
	U_6	Update	Delete	Insert	Select
	U_7	Insert	Update	Select	Delete
.....	.....	.....	.....	.....	.....

through the database administrator on the remote server to the user(s) who have a higher insert priority (U_5 or U_7). The user (U_7) will lead the operation because he has the 1^{st} priority. This is presented in Figure 3.

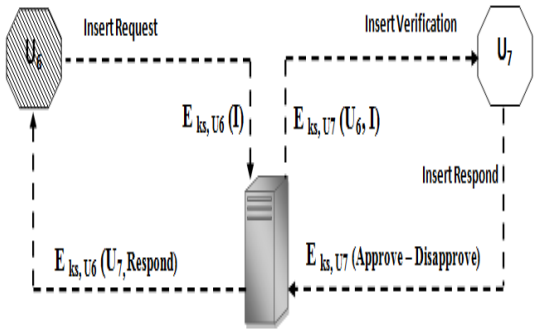

Fig. 3: Authentication parameter

As presented in Figure 3, user (U_6) interacts with the remote server by sending encrypted request using a secret key (between user (U_6) and the remote server). This request contains the insert request from user (U_6). The remote server sends encrypted request to user (U_7) using a secret key (between user (U_7) and the remote server) and contains the insert request from user (U_6).

User (U_7) receives the request and verifies whether this request should be granted or denied and sends the respond to the remote server. Finally, the remote server passes the respond from user (U_7) to user (U_6).

4.4 Separation of Duty (SoD) Policy

This policy presents reinforcement to the architecture of the previous policy. The policy presented in section 4.3 provides a security layer for monitoring user behavior but the most challenging problem here is that policy 4.3 will be time consuming especially when the user wants to perform an urgent operation and the criteria of time is very critical in it and any delay in this operation will make the operation useless.

Policy 4.4 presents a support to policy 4.3 by implementing policy 4.3 based on the classification of database attributes. Database attributes can be classified as normal attributes (**An**), sensitive attributes (**As**), and most sensitive attributes (**Ams**). Normal attributes are part of database columns in which there will be no harm when there is a disclosure of data. Sensitive and most sensitive attributes are part of database columns in which there will be a great harm when there is a disclosure of data.

Many researchers present the process of classifying database attributes bases on its sensitivity in order to perform an encryption operation into it. Zhou et al [9] presents a comprehensive view for encrypting database based on database level, table level, record level, and attribute level. Liu et al [10] presents a new paradigm for database encryption in which database encryption can be provided as a service to applications with seamless access to encrypted database. Practically, the best way to implement cryptographic techniques is through encrypting sensitive fields only. This can reduce the overhead on server processing and also the execution time for performing encryption process.

This policy supports policy 4.3 by implementing it based on sensitive and most sensitive attributes only. This means that if the user needs to perform a specific action on a normal attribute inside database, then policy 4.3 will be deactivated and will pass the required action. If the user needs to perform a specific action on a sensitive or most sensitive attributes, then policy 4.3 will be activated again and will ask for a secret key sharing.

4.5 Authorization Block Policy

This policy provides a fifth layer of security into which an authorization data block is created in order to analyze the user behavior inside database and compares it with the stored user profile that was embedded inside database by the super admin (**SA**) or the **K** DBAs. Any user behavior that deviates from the stored user profile is considered as malicious user.

As presented in Table 2, the monitoring block consists of the following fields:

–**Username**: To determine the name of the user connected to database.
–**User Profile**: To determine the classification of the user based on his activity on database whether an active user, intermediate user, or inactive user as presented in policy 4.2.
–**Table/View Name**: The name of the table or view that the user performs his DML operations into it.
–**Sensitive & Most Sensitive Attributes**: The attributes of database which the user will perform his operations into it as presented in policy 4.4.
–**Timestamp (T1, T2, T3, and T4)**: The time that the DML operation will be executed based on the format DD-MON-YYYY HH:MI:SS
–**Insert**: The number of insert operations allowed for each user to perform inside database.
–**Max Insert**: The maximum number of insert operations allowed for each user.
–The remaining attributes (update-Delete-Select) will be on the same manner like the (Insert) attribute.

4.6 Intrusion Detection and Response Policy

With greater data integration, aggregation and sharing, preventing data theft has become a big challenge, both from outside and within. Standard database security controls such as access control mechanisms, authentication, encryption technologies and so forth are not of much help when it comes to preventing data theft from insiders [11].

The main objective of this policy is based on identifying the maximum allowed number of insert, update, delete, or select operations granted for each user profile. Each time the user performs an insert operation for instance, the (**Insert**) attribute in Table 2 will be incremented by 1 until the (**Insert**) and (**Max Insert**) attributes are equal. If the number of insert operations (**Insert**) exceeds the maximum number of insert operations allowed for each user (**Max Insert**), an alarm will be raised determining that the user is an intrusive user.

The intrusion response action on this intrusive behavior will be based on the classification of the user profile as presented in policy 4.2. Any active user (U_A) or intermediate user (U_{IM}) who is detected as intrusive user means that an aggressive action must be taken to automatically disconnect his connection from the application server. He will need to reconnect again but under the control of the super admin (SA). Any inactive user (U_I) who is detected as intrusive user will block or suspend his profile from database server until the super admin (**SA**) or a number of **K** DBAs deactivate the suspension again.

4.7 Cryptography Policy

An additional layer of security can be combined to the security architecture by encrypting the authorization block in policy 4.5 in order not to give the user the ability

Table 2: Authorization Block Processes

Username	User Profile	Table/ View name	Sensitive & most sensitive attributes	Insert	Max Insert	T_1	Update	Max Update	T_2	Delete	Max Delete	T_3	Select	Max Select	T_4
User1	Active	Emp	Salary	0	5	Timestamp	0	4	---	0	3	---	0	3	---
User2	Inactive	Client	Balance	0	0	Timestamp	0	5	---	0	0	---	0	4	---
----	----		----	----	----	----	----	----	--	----	----	--	----	----	--

to change the policy itself by increasing his/her DML operations.

The encryption process will be executed on the server side by the super admin (**SA**) or a number of **K** DBAs who share their secret key signatures using LaGrange interpolation polynomial algorithm. The secret key sharing is provided in order not to allow any single DBA to perform the cryptography process alone.

The encryption process will be executed using AES (Advanced Encryption Standard) algorithm which is considered as one of the modern algorithms in symmetric cryptography.

The AES algorithm uses 128-bit, 192-bit, or 256-bit key size and plaintext block size of 128-bit. The super admin (**SA**) or a number of **K** DBAs will be the only trusted component that will generate the cryptography process.

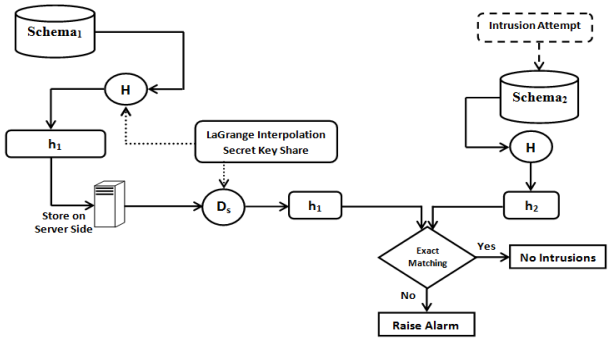

Fig. 4: Authentication Process

4.8 Authentication Factor Policy

The security architecture from policy 4.1 to policy 4.7 depends on securing database that lies on the server side by monitoring different users and DBAs who are performing actions on database. One of the major challenges presented by kamra et al [8] is that there is a dire need to provide a second factor of authentication that will provide a second layer of defense when certain anomalous actions are executed against critical system resources on database.

Policy 4.8 deals with this problem and is considered as the last layer of defense in protecting confidential data from unintended activity or inadvertent mistakes made by authorized users or DBAs.

This policy presents a second factor of authentication by using hash function algorithm. A hash function accepts a variable size message M as input and produces a fixed-size message called message digest as output.

The main mechanism as presented in Figure 4 depends on executing the hash function (**H**) on the entire database schema (**Schema**$_1$) using secret key sharing of **K** DBAs or the secret key of the super admin (**SA**).

The secret keys are generated by using LaGrange interpolation polynomial algorithm to produce a fixed length hash called ($\mathbf{h}_1$). The resulting ciphertext (**C**) is stored on the server side.

If any intruder succeeded in passing all previous policies and provides any modifications on the entire database schema (**Schema**$_1$), this will produce a new database schema (**Schema**$_2$). By implementing the hash function (**H**) on the second schema (**Schema**$_2$), this will produce the second hash ($\mathbf{h}_2$).

When the **K** DBAs or the super admin (**SA**) wants to retrieve the original database schema (**Schema**$_1$), the decryption process will be implemented on the ciphertext (**C**) that was stored on the server side. A secret key sharing must be generated again using LaGrange interpolation polynomial algorithm to retrieve the original hash ($\mathbf{h}_1$).

If there is no exact matching between the retrieved hash ($\mathbf{h}_1$) and the modified hash ($\mathbf{h}_2$), then an intrusion will be exist otherwise no intrusion will be found. The main algorithm is presented as follows:

$h_1 = H(Schema_1, K)$

$h_2 = H(Schema_2, K^*)$

$D_s(C, K) = h_1$

If $h_1 = h_2$ No intrusion exist

If $h_1 \neq h_2$ An intrusion exist

5 Conclusion

Database security mechanisms and techniques remain important goals in any data management systems whether these systems are commercial, industrial, educational, medical, or even military systems. This paper builds a security architecture based on access control mechanisms, cryptography techniques, and intrusion detection and response systems. These mechanisms and techniques are combined together to present an interactive multi-layer policies for securing relational databases. The paper provides a kind of security for monitoring and controlling both legitimate users and database administrators to protect database system from internal intruders.

Acknowledgement

This work is partially supported by the ministry of higher education in Egypt and we are grateful to them.
The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

References

[1] J. Yu, P. Yan, X. Zheng, Database Encryption and Confirmation Mechanism Research, Proceeding in the IEEE International Conference on Multimedia Technology, 1-4 (2010).
[2] U. Maurer, The Role of Cryptography in Database Security, Proceedings of ACM SIGMOD International Conference on Management of Data, USA, (2004).
[3] E. Bertino, R. Sandhu, Database Security - Concepts, Approaches, and Challenges, IEEE Transactions on Dependable and Secure Computing, **2**, 2-19 (2005).
[4] S. Antoshchuk, A. Blazhko, E. Saoud, Automated Design Method of Hierarchical Access Control in Database, Proceeding in the IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems, Italy, 360-363 (2009).
[5] L. Pan, Using Criterion-Based Access Control for Multilevel Database Security, Proceeding in the IEEE International Symposium on Electronic Commerce and Security, 518-522 (2008).
[6] C. Shyni, S. Swamynathan, Purpose Based Access Control for Privacy Protection in Object Relational Database Systems, Proceeding in the IEEE international conference on Data Storage and Data Engineering, 90-94 (2010).
[7] A. Bouchahda, N. Thanh, A. Bouchahda, F. Labbene, Enforcing Access Control to Web Databases, Proceeding in 10th IEEE International Conference on Computer and Information Technology (CIT), 612-619 (2010).
[8] A. Kamra, E. Bertino, Design and Implementation of an Intrusion Response System for Relational database, IEEE Transactions on Knowledge and Data Engineering, **23**, (2011).
[9] Z. Yuping, W. Xinghui, Research and Realization of Multi-level Encryption Method for Database, Proceeding in the IEEE 2nd International Conference on Advanced Computer Control (ICACC), 1-4 (2010).
[10] L. Liu, J. Gai, A New Lightweight Database Encryption Scheme Transparent to Applications, Proceeding in the IEEE International Conference on Industrial Informatics (INDIN), Korea, 135-140 (2008).
[11] A. Kamra, E. Bertino, G. Lebanon, Mechanisms for Database Intrusion Detection and Response, Proceeding in the 2nd ACM Workshop on Innovative Database Research, Vancouver, Canada, (2008).
[12] H. Kadehm, T. Amagasa, H. Kitagawa, A Novel Framework for Database Security based on Mixed Cryptography, Proceeding in the 8 IEEE 4th International Conference on Internet and Web Applications and Services, Japan, 163-170 (2009).
[13] J. Hou, Research on Database Security of E-Commerce Based on Hybrid Encryption, Proceeding in the International Symposium on Computational Web Information Systems and Applications, China, 363-366 (2009).
[14] W. Xing-hui, M. Xiu-jun, Research of the Database Encryption Technique Based on Hybrid Cryptography, Proceeding in the IEEE International Symposium on Computational Intelligence and Design, China, 68-71 (2010).
[15] K. Kaur, K. Dhindsa, G. Singh, Numeric to Numeric Encryption of Databases: Using 3Kdec Algorithm, Proceeding in the IEEE International Conference on Advanced Computing, India, 1501-1505 (2009).
[16] D. Lee, S. Lee, T. Nam, Y. Song, A Bucket ID Transformation Scheme for Efficient Database Encryption, Proceeding in the IEEE International Conference on Information Networking, 1-5 (2008).
[17] Z. Rashid, A. Basit, Z. Anwar, TRDBAC: Temporal Reflective Database Access Control, Proceeding in the IEEE 6th International Conference on Emerging Technologies (ICET), 337-342 (2010).
[18] B. Qing-hai, Z. Ying, Study on the Access Control Model, Proceeding in the IEEE international conference Cross Strait Quad-Regional Radio Science and Wireless Technology, 830-834 (2011).
[19] A. Bouchahda, N. Thanh, A. Bouchahda, F. Labbene, RBAC+: Dynamic Access Control for RBAC-Administered Web-Based Databases, Proceeding in the IEEE 4th International Conference on Emerging Security Information Systems and Technologies, 135-140 (2010).