1998 International Computer Symposium
Workshop on Cryptology and Information Security
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

A New Multilevel Database Field Encryption and Authentication Cryptosystem

Chien-Yuan Chen, Chin-Chen Chang**, Wei-Pang Yang****

*Department of Information Engineering
I-Shou University, Kaohsiung County, Taiwan, 84008 R.O.C.
email: cychen@csa500.isu.edu.tw,
**Institute of Computer Science and Information Engineering,
National Chung Cheng University, Chiayi, Taiwan 621, R.O.C.
email: ccc@cs.ccu.edu.tw
***Department of Computer and Information Science,
National Chiao Tung University, Hsinchu, Taiwan 300, R.O.C.

Abstract

A new field encryption and authentication cryptosystem suitable for multilevel database is presented. In this system, we develop a way of combining the field encryption and decryption scheme with the record-oriented technique such that it possesses their advantages at the same time. It is immune from the ciphertext search attack and the plaintext or ciphertext substitution attack, and allows projections to be executed and individual field values to be decrypted and authenticated. It is noteworthy that because of the use of random filters in field authentication, the selection operation can be straightforwardly applied without decryption, which can eliminate the exposure of some unqualified records and will increase the speed of query.

1. Introduction

There are four methods to achieve database security [6]: physical security, operating system security, DBMS security, data encryption. However, the first three methods cannot completely solve the database security problem due to the following three reasons. First, raw data may be disclosed as it exists in readable form inside a database [3]. Next, it is improper for operating system and DBMS to control the disclosure of sensitive data which must frequently be backed up in storage media when systems fail or disks crash. Third, it is unlikely to authenticate the data which has been modified by an intruder [10].

To overcome the above mentioned problems, data encryption methods [2, 3, 4, 5, 7, 9, 10] are presented. In these methods, the problem of the disclosure of raw data is eliminated because the stored data in database are of the ciphertext form. In addition, the problem of authenticating the data is also resolved because an intruder cannot modify the ciphertext without knowing the correct encryption key.

Database encryption methods can be categorized into two. One is database encryption with a single key [8] while the other is that with subkeys [3, 4]. The former needs a trusted center who controls all accesses to the data stored in the database. In other words, all encryption and decryption is executed by the trusted center with the private key. In the latter, however, users can execute decryption with their own subkeys. In 1981, Davida et al. first presented a database encryption/decryption system with subkeys. This system, also called a record-oriented system, based on Chinese Remainder Theorem, has an important property of having subkeys that allow the encryption and decryption of fields within a record. Nevertheless, extra redundant bits in each field are required in their system to overcome the known plaintext attack. This drawback motivated [9, 10]. However, the record-oriented systems, such as [3, 9,10], have common disadvantages. First, projections cannot be applied before decryption to eliminate unneeded fields. Next, selections cannot be performed without decrypting up through the fields over which selections are going to be executed. Third, it is not suitable for applications that need to keep only one or two short fields in the clear forms of text for fast retrieval.

Database encryption and authentication at the field level is very attractive because projections are allowed to be executed and individual field values are allowed to be decrypted and authenticated. This field encryption and authentication system, however, is vulnerable to the ciphertext searching attack and the plaintext or ciphertext substitution attack. It is noteworthy that these problems do not arise in the record-oriented systems. To eliminate these problems, Denning [4] proposed a novel field encryption and authentication system. However, in this system, selections cannot be applied before decryption to eliminate unqualified records.

In this paper, we present a new multilevel database field encryption and authentication cryptosystem. In our system, the field encryption and authentication scheme based on random filters is proposed. According to our

proposed scheme, selections can be applied before decryption to eliminate partial unqualified records. Further, the speed of query is drastically enhanced because some decryption operations are not executed. Of course, projections can also be applied before decryption to eliminate unneeded fields. Furthermore, we present a scheme according to a hierarchy of security classes such that the classified data can be discriminated by the field classification label and decrypted by the legal users.

This paper is organized as follows. Section 2 shows a multilevel database field encryption and authentication scheme. In Section 3, cryptographic relational operations are given. Final section makes some conclusions.

2. Multilevel Database Field Encryption/Decryption Scheme

In this section, we first review the field encryption and decryption scheme [4]. Then, we present a new field authentication scheme with subkeys based on random filters in Subsection 2.2. This scheme can capitalize on the record authenticator to provide a field authentication function without storing any field authenticator. In Subsection 2.3, we introduce a way of dealing with the classified data and security classes. However, it poses three weaknesses. To overcome these three weaknesses, we present a new field encryption and authentication scheme for the multilevel database in Subsection 2.4. About computational complexity, storage space, and cryptanalysis of this scheme are shown in Subsection 2.5, 2.6, and 2.7, respectively.

2.1 The Field Encryption Scheme with Subkeys

As in [4], the field encryption is described as follows. Consider a file of N records where each record has M fields. The purpose is to seal the data in some field j of each record. Obviously, encrypting the field value with a secret key (or called subkey) x is the simplest way. Let $E_x(p)$ and $D_x(p)$ denote the encryption and decryption of p under the secret key x, respectively. The encryption and decryption technique we use is the Data Encryption Standard (DES) [12]. Let p_{ij} be the field value for the field j of the record i. Then, we have the corresponding ciphertext $c_{ij} = E_x(p_{ij})$. Then c_{ij} replaces p_{ij} as the j^{th} field value in the record i. This field encryption scheme with the same key is insecure as a result of the ciphertext searching attack. Therefore, one solution to this problem is that each field value p_{ij} is encrypted with a distinct cryptographic key x_{ij}. Obviously, the ciphertext is changed into $c_{ij} = E_{x_{ij}}(p_{ij})$. As for the generation of the cryptographic key x_{ij}, we must make some assumptions. First, each field has a field identification number which can uniquely identify that field. Next, let the first field of each record be the primary key which can uniquely identify the record and cannot be encrypted. Let r_i (= p_{i1}) be the primary key for the record i and f_j be the field identification number of the field j. Therefore, the cryptographic key x_{ij} is defined as $g(r_i, f_j, x)$, where $g(\cdot)$ is a key generation function satisfying that

$$g(r_i, f_j, x) = E_{(E_x(r_i))}(f_j).$$

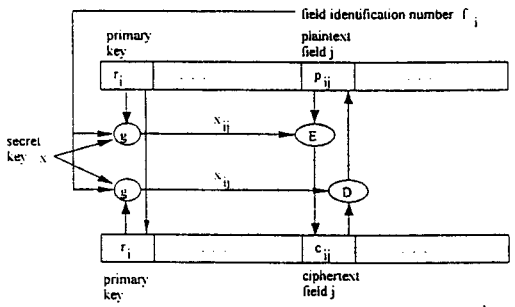


Figure 2. Field Encryption and Decryption

The encryption procedure E and the decryption procedure D of the field value p_{ij} are shown in Figure 2. From Figure 2, if we want to encrypt the field value p_{ij} for the field j of the record i, the cryptographic key x_{ij} has to be first generated. We can compute the cryptographic key $g(r_i, f_j, x)$ using the generation function $g(\cdot)$, the secret key x, the field identification number f_j, and the primary key r_i stored in the first field of the i^{th} record. Then, the field value p_{ij} is encrypted by the DES technique under the key x_{ij}. Thus, we have the ciphertext $c_{ij} = E_{x_{ij}}(p_{ij})$. This ciphertext c_{ij} will be stored as the j^{th} field value of the record i.

To recover the field value p_{ij} from the encrypted database, we only decrypt the ciphertext c_{ij}. Like encryption, the cryptographic key x_{ij} is first generated by computing $g(r_i, f_j, x)$. Then, using the DES technique to decrypt the ciphertext c_{ij} under the key x_{ij}, we can recover the original field value $p_{ij} = D_{x_{ij}}(c_{ij})$.

2.2 The Field Authentication Scheme with Subkeys

The purpose in this section is to verify whether some of the field values decrypted from the encrypted database have been modified or not. The trivial way of doing this is to store a cryptographic authenticator with each record. The authenticator, of course, is a cryptographic function of the entire record. Therefore, the authenticator is recomputed from the field values when the record is retrieved. If the computed authenticator is equivalent to the stored authenticator, the record may be modified with the probability $1/2^A$, where A is the length of the authenticator in bits. However, one cannot know whether an individual field value has been modified or not because the authenticator has something to do with all field values of a record. To authenticate an individual field, the way of using authenticators with individual fields was introduced in [4]. Unfortunately, it is vulnerable to plaintext or

ciphertext substitution attack. Let s_{ij} be the authenticator of the value c_{ij} in the field j of record i such that $s_{ij} = E_x(c_{ij})$, where x is the secret key. The pair (c_{ij}, s_{ij}) is stored in the encrypted database. If an intruder interchanges the pair (c_{ij}, s_{ij}) of the record i and the pair (c_{hj}, s_{hj}) of the record h, then any legal user cannot detect this change.

The above problem arises when the authenticators are generated by using the same secret key. To eliminate this problem, Denning [4] developed a field authentication scheme such that each field authenticator is generated by virtue of encrypting each field value with a distinct cryptographic key x_{ij}.

Using the above technique, we present a new field authentication scheme based on the concept of random filters. In our scheme, each record has a record authenticator related with all field values of that record. According to the record authenticator, a legal user can detect whether an individual field value has been modified or not without projecting other fields. Therefore, we can provide a field authentication function without storing any field authenticator.

Let a record R_i have M field values $p_{i1}, p_{i2}, ..., p_{iM}$, and have M field secret subkeys, say $y_1, y_2, ..., y_M$. How to generate a authenticator of the record R_i? The idea of our strategy mainly comes from random filters [13]. There are a piece of memory which is served as the hash space and several, say d, hash functions. Let hash space of the random filter be K individual addressable bits with addresses 0 through K - 1, where K bits are served as a record authenticator. To begin with, all bits in the hash space are initialized to 0. Each field value in the field j concatenates the subkey y_j to generate $(p_{ij} \| y_j)$, where "||" means the concatenation operator. The $(p_{ij} \| y_j)$ is then hash-coded into d bit addresses, say $b_1, b_2, ..., b_d$, where $b_1, b_2, ..., b_d$ need not to be all distinct. Finally, all these d bits addressed by $b_1, b_2, ...,$ and b_d are set to 1.

While a given decrypted field value is to be authenticated, we generate a sequence of bit addresses, say $b'_1, b'_2, ... b'_d$, according to that field value and the corresponding subkey in the same way as mentioned previously. If these d bits addresses by $b'_1, b'_2, ..., b'_d$ in the authenticator are all 1, this decrypted field value is considered to be correct.

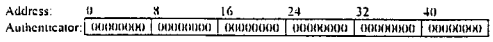
Example 2.2.1

Suppose a record R_i includes five fields, of which field values are $p_{i1}, p_{i2}, p_{i3}, p_{i4}$ and p_{i5}, and five field subkeys are y_1, y_2, y_3, y_4 and y_5, and the hash functions used for Random filters are $\{h_1, h_2, h_3, h_4\}$. We use a hash space of 48 individual addressable bits as a record authenticator, which are addressed by 0 through 47. The following lists the outputs of all hash functions with the inputs, $p_{i1}, p_{i2}, p_{i3}, p_{i4}$ and p_{i5}:

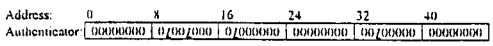
$h_1(p_{i1} \| y_1) = 12$, $h_2(p_{i1} \| y_1) = 34$, $h_3(p_{i1} \| y_1) = 9$, $h_4(p_{i1} \| y_1) = 17$,
$h_1(p_{i2} \| y_2) = 4$, $h_2(p_{i2} \| y_2) = 33$, $h_3(p_{i2} \| y_2) = 17$, $h_4(p_{i2} \| y_2) = 9$,
$h_1(p_{i3} \| y_3) = 7$, $h_2(p_{i3} \| y_3) = 21$, $h_3(p_{i3} \| y_3) = 18$, $h_4(p_{i3} \| y_3) = 7$,
$h_1(p_{i4} \| y_4) = 37$, $h_2(p_{i4} \| y_4) = 42$, $h_3(p_{i4} \| y_4) = 34$, $h_4(p_{i4} \| y_4) = 17$,
$h_1(p_{i5} \| y_5) = 1$, $h_2(p_{i5} \| y_5) = 36$, $h_3(p_{i5} \| y_5) = 30$, $h_4(p_{i5} \| y_5) = 1$.

Note that these hash values are not distinct, for instance, $h_1(p_{i3} \| y_3) = h_4(p_{i3} \| y_3) = 7$ and $h_1(p_{i5} \| y_5) = h_4(p_{i5} \| y_5) = 1$. According to the above hash values, a record authenticator of R_i can be generated as follows.

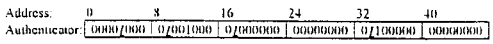
Initial state:



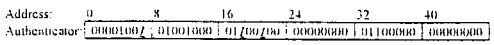
After the insertion of p_{i1}:



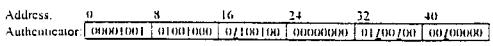
After the insertion of p_{i2}:



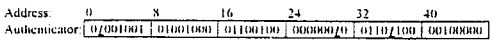
After the insertion of p_{i3}:



After the insertion of p_{i4}:



After the insertion of p_{i5}:



Finally, the final state of authenticator is stored as the record authenticator A_i.

If one wants to verify whether the decrypted field value p' corresponding to the field j is contained in the record R_i or not, he executes the following:

$h_1(p' \| y_j) = 4$, $h_2(p' \| y_j) = 33$, $h_3(p' \| y_j) = 17$, $h_4(p' \| y_j) = 9$.

As a result, we conclude that the decrypted field value p' corresponding to the field j is contained in the record R_i

because the forth, 33rd, 17th, 9th addresses of the record authenticator A_i are all equal to 1.

2.3 The Classified Data and the Security Classes

In a multilevel database, data are generally classified into four levels: top secret, secret, confidential, and unclassified. The classification labels are used by the trusted interface [11] to decide what data a given user is allowed to access, according to the user's clearness. The trusted interface must validate each retrieval against an illegal access. If the classification label attached to the data in the clear form of text, then that will result in the possibility of Trojan Horse attack (e.g., the response to a predetermined query may be an unclassified value equivalent to a classified one). To avoid such Trojan Horse attack, the classification labels should be sealed. In 1984, Denning presented a scheme which can protect the record classified labels. In Figure 3, we illustrate how the record classified label in the record i is protected. Let the record classified label p_{iM} be placed in the last field of the record i. Therefore, the stored record contains the ciphertext classification $C_{iM} = E_{x_{iM}}(p_{iM})$, where $x_{iM} = g(r_i, f_M, K)$, and the field authenticator $s_{ij} = E_{x_{ij}}(p_{ij} \| p_{iM})$, where $x_{ij} = g(r_i, f_j, K)$, for a key generation function $g(\cdot)$, and "||" means the concatenation operator. It is noteworthy that K denotes the secret key of the trusted interface.

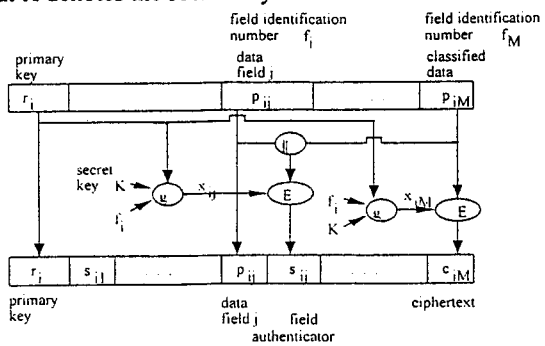


Figure 3. Protection for Classified Data

If a legal user wants to access the j^{th} field of the records, he has to project three fields, say the first field (r_i, s_{i1}), the j^{th} field (p_{ij}, s_{ij}), and the last field c_{iM}, and these data will be returned to the trusted interface. Therefore, the trusted interface uses its own secret key K to compute the record classification label $p_{iM} = E_{g(r_i, f_M, K)}(c_{iM})$ and verify the two equations $s_{i1} = D_{g(r_i, f_1, K)}(r_i)$ and $s_{ij} = D_{g(r_i, f_j, K)}(p_{ij} \| p_{iM})$. If the verification is valid, the trusted interface accepts user's request. From the above method, we find there are some weaknesses.

(1) The field classification labels are not treated.
(2) No one except the trusted interface can verify the correctness of the data.
(3) It cannot be applied to the encrypted database when the users are cleared to different security levels.

In the next subsection, we will develop a new multilevel database field encryption/decryption scheme to overcome the above weaknesses. In each record, the field classification label is attached to each field such that the field data, not only the record data, can be classified. It is noteworthy that we capitalize on the record authenticator mentioned in Subsection 2.2, instead of the field authenticator. Furthermore, all encryption and decryption involves the cryptographic subkeys of users which are generated according to the corresponding security classes. Therefore, legal users can decrypt the field values without the trusted interface. Obviously, the above three weakness are overcome.

2.4 Multilevel Database Field Encryption/Decryption Scheme

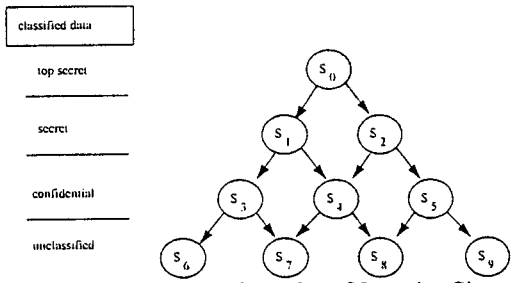


Figure 4. A Hierarchy of Security Classes

Now, let us consider the situation where the users and data in the database are classified into a hierarchy of security classes. Figure 4 illustrates an example. Assume that every user belongs to one of security class in $\{S_0, S_1, ..., S_9\}$. The set of classes is partially ordered by "≤". For example, $S_i \le S_j$ denotes the users in S_j can derive the secret keys of the users in S_i. As shown in Figure 4, we have $S_3 \le S_1$ and $S_4 \le S_1$. But, S_3 is not related with S_4. In addition, this figure gives the security levels of data. First, the data only accessed by the users in security class S_0 are called the top secret data. Next, the data, excepting top secret data, only accessed by the users in S_0, S_1, and S_2 are called the secret data. The third kind of data is called the confidential data which can be accessed by the users in any security class except S_6, S_7, S_8, and S_9. Last, all data except the mentioned ones are called the unclassified data.

According to Figure 4, we propose an access scheme which can be applied to the multilevel encrypted database. Let d_{ij} be the classified data in the field j of the record i. Each record R_i includes the record primary key $r_i = p_{i1}$, M-1 field values p_{i2}, p_{i3}, ..., $p_{i(M-1)}$, and M-1 field classified data d_{i2}, d_{i3}, ..., $d_{i(M-1)}$. Then, each field value p_{ij} in the field j of the record i can be encrypted as

$$c_{ij} = E_{g(r_i, f_j, y)}(p_{ij} \| d_{ij}),$$

where $g(\cdot)$ and f_j are aforementioned and y means a field secret subkey, or called a cryptograpgic subkey of some users. Therefore, the encrypted record R'_i is $[r_i, c_{i1}, c_{i2}, ..., c_{iM}, A_i]$, where A_i is a record authenticator.

Let p_{ih} be a confidential field value, and p_{ik} be a secret field value. If the subkey of the h^{th} field is y_3 which is held by the user U_3 in the security class S_3 and the subkey of the k^{th} field is y_1 which is held by the user U_1 in the security class S_1, we have

$$c_{ih} = E_{g(r_i, f_h, y_3)}(p_{ih} \| d_{ih}), \text{ and}$$
$$c_{ik} = E_{g(r_i, f_k, y_1)}(p_{ik} \| d_{ik}).$$

Now, when the user U_1 wants to access the field value in the field h of the record i, he first derives the secret key y_3 of the user U_3 by the key management scheme [11].(Here, we assume that the user U_1 can access to the owner of the field h.) He then obtains the encrypted h^{th} field value c_{ih}, a record primary key r_i, and a record authenticator A_i from the encrypted record, computes $p_{ih} \| d_{ih} = D_{g(r_i, f_h, y_3)}(c_{ih})$, and extracts the field value p_{ih} from $(p_{ih} \| d_{ih})$. Finally, he can verify the decrypted field value p_{ih} by using the field authentication scheme, and then obtains the classified data d_{ih}, called the cell classification. The cell classification is not a secret to the legal users. In our view, the cell classification has two important functions. First, it is a director of the classified level of the field value. Second, it plays a part in detection. It will be easily detected when the cell classification indicates the field value which does not entitle the user to access.

2.5 Computational Complexity

In this subsection, we analyze the complexity of encryption, decryption and authentication. Suppose that each record contains M fields and the number of bits of each field is B on the average. The computational time required for encryption, decryption and authentication of a record is stated as follows.

First, we derive the computational time of the field encryption scheme. This scheme includes two phases: generating a cryptographic key and encrypting the field value. Because the generation of the cryptographic key needs executing DES twice, computational time of this phase requires 2DES(64), where DES(64) means the time units required to encipher 64 bits of text using the DES device. When a record is encrypted by DES, each field value must be partitioned into pieces of 64 bits. The required computational time is $M(\lceil \frac{B}{64} \rceil DES(64))$. Combining this two phases, we have the computational time of encrypting a record, which is

$$2M(DES(64))+M(\lceil \frac{B}{64} \rceil DES(64)). \qquad (2.5.1)$$

Computing DES(64) needs sixteen rounds of one table-lookup and one XOR operation. This implies that $DES(64) = 16(t_{tl} + t_{xor})$, where t_{tl} and t_{xor} are the time needed by table-lookup and XOR operation, respectively. Thus, Expression (2.5.1) can be rewritten as

$$(32M + M\lceil \frac{B}{4} \rceil)(t_{tl} + t_{xor}).$$

Now, consider a situation that a relation T is encrypted. The computational time of encrypting a relation needs

$$(32N(1+M) + MN\lceil \frac{B}{4} \rceil)(t_{tl} + t_{xor})$$

because generating all the cryptographic keys needs N(1+M)DES(64) time units.

Next, the computational time of the field decryption scheme is equivalent to that of the field encryption scheme.

Last, the required computational time of the field authentication scheme is computed. This scheme contains two processes: to generate record authenticator and to verify the decrypted data. We first define t_h to be the time units needed by executing a search in a hashing function, t_{AND} to be that of an AND operator, and t_{OR} to be that needed by an OR operator. Therefore, the computational time of generating a record authenticator requires

$$dt_h + (M-1)\, t_{OR},$$

where d means the number of hashing functions.

On the other hand, the computational time of verifying the decrypted data requires

$$dt_h + t_{AND}.$$

2.6 Storage Space

Like the aforementioned assumption, there are N records in a database, M fields in each record, and an average of B bits in each field. Our proposed scheme needs M secret keys, M field numbers for the DES. The raw data in the database are of MNB bits, and the encrypted data stored in the encrypted database are of $(MN\lceil \frac{B}{64} \rceil 64 + NA)$ bits, where A denotes the number of bits of a record authenticator.

2.7 Cryptanalysis

We first emphasis the fact that the cipher searching attack and plaintext or ciphertext substitution attack pose no threat to our proposed scheme due to each field value encrypted with a different cryptographic key. Based on the same reason, one cannot derive another encrypted data from the given data and its ciphertext.

We next discuss some cryptographic considerations involved in the development of the field authentication scheme.

In an earlier version of the field authentication scheme, each field value p_{ij} in the field j of the record i is hash-coded into d bit addresses. Then, these d bit addresses of a record authenticator are all set to 1. Assume the domain D of some fields is small, which contains k data elements, say D_1, D_2, ..., D_k. We can execute the following attack to discover the original data. Given an encrypted record R'_i, including the record authenticator A_i, we select a data element D_j from the domain D and hash-code it into d bit addresses. If all these d bit addresses of the record authenticator A_i are all 1, then D_j may be the original field value. Once all elements in the domain D have been tried, we can collect all the candidate data elements which pass the above authentication. If there is one candidate data element, we find the one is the original field value; otherwise, we can reduce the size of the search domain D. However, this attack poses no threat to our proposed field authentication scheme because no data can be hash-coded without knowing the secret key.

3. Cryptographic Relational Algebra

Due to the field encryption scheme, projections can be regarded as one in the relation database. Here, in Algorithm 3.1, we present an another important operation: selection.

Algorithm 3.1: Selection

I

nput: N records $[c_{i1}= r_i, c_{i2}, ..., c_{iM}, A_i]$, i = 1, 2, ..., N, in encrypted database; the queried field value v associated to the field j

Output: encrypted records of which the j^{th} field value is equivalent to v

Step 1: Set an integer S of K bits to 0.

Step 2: Compute T = (v || y_j). /* y_j is the secret key of the field j.*/

Step 3: Hash-code T into d bit addresses, say b_1, b_2, ..., b_d. In S, set all these d bits addressed by b_1 through b_d to 1.

Step 4: Set k to 1.

Step 5: If $A_k \wedge S = S$ ("$\wedge$" denotes the bit AND operator), /* A_k is the record authenticator of the record k.*/ then

generate the cryptographic key $x_{kj} = g(r_k, f_j, y_j)$,

compute the j^{th} field value $p_{kj}||d_{kj}= D_{x_{kj}}(c_{kj})$,

and if p_{kj} = v then output the record k.

Step 6: Compute k = k + 1.

Step 7: If k > N then goto Step 8; else goto Step 5.

Step 8: Stop.

4. Conclusions

In this paper, we have presented a multilevel database field encryption and decryption cryptosystem. This system contains several schemes. First, the field encryption scheme can encrypt/decrypt field values at field level without resulting in the ciphertext search attack and the plaintext or ciphertext substitution attack. Next, the field authentication scheme based on random filters can perform field authentication without storing any field authenticator according to the record authenticator. Third, according to a hierarchy of security classes, we present a scheme such that the classified data can be discriminated by the field classification label and decrypted by the legal users.

In cryptographic relational algebra, projections can be applied before decryption to eliminate unneeded fields and selections can be performed to filter out some unqualified records. These two operators not only enhance the performance due to the elimination of unneeded decryption but also reduce the space of processing data.

5. References

1. S. G. Akl, and P. D. Taylor, "Cryptographic Solution to a Problem of Access Control in a Hierarchy," ACM Transactions on Computer System, VOL. 1, No. 3, pp. 239-247, 1983.
2. Y. M. Babad and J. A. Hoffer, "Data Element Security and Its Effects on File Segmentation," IEEE Transactions on Software Engineering, SE-6(5), pp. 402-410, 1980.
3. G. I. Davida, D. L. Wells, and J. B. Kam, "A Database Encryption System with Subkeys," ACM Transactions on Database Systems, Vol. 6, No. 2, pp.312-328 , 1981.
4. D.E. Denning, "Field Encryption and Authentication," Advances in Cryptology - CRYPTO'83, Springer-Verlag, pp.231-247, 1984.
5. R. Eriksson and K. Beckman, "Protection of Data-Bases Using File Encryption," Proc. of the First Security Conference, IFIP/Sec'83, pp.217-221, 1983.
6. E. B. Fernandez, R. C. Summurs, and C. Wood, Database Security and Integrity, Addison-Wesley, Massachusetts, 1980.
7. J. Feigenbaum, M. Y. Liberman, and R. N. Wright, "Cryptographic Protection of Database and Software," DIMACS Series in Discrete Math. and Theoretical Computer Science, Vol. 2, pp. 161-172, 1991.
8. E. Gudes, "The Design of a Cryptography Based Secure File System," IEEE Transactions on Software Engineering, SE-6(5), pp. 411-420, 1980.
9. M. S. Hwang, C. F. Yu, and W. P. Yang, "A Two-Phase Encryption Scheme for Enhancing Database Security", Journal of System and Software, Vol.31, No.12, pp.257~265. 1995.

10. C. H. Lin, C. C. Chang, and C. T. Lee, "A Record-Oriented Cryptosystem for Database Sharing," The Computer Journal, Vol. 35, N0. 6, pp.658-660, 1992.
11. Multilevel Data Managment Security, Committee on Multilevel Data Manegment Security, Air Force Studies Board, National Research Council, 1982.
12. National Bureau of Standard, Data Encryption Standard, FIPS, NBS, 1977.
13. C. Y. Wang, W. P. Yang, and J. C. R. Tseng, "Random Filter and Its Analysis," International Journal Computer Mathematics, Vol. 33, pp. 181-194, 1990.