An Experimental Evaluation for a New Column – Level Access Control Mechanism for Electronic Health Record Systems

Pham Thi Bach Hue[1], Sven Wohlgemuth[2], Isao Echizen[2],
Nguyen Dinh Thuc[1], Dong Thi Bich Thuy[1]

[1]University of Science, VNU – HCMC
227 Nguyen Van Cu, District 5, Ho Chi Minh City, Vietnam
{ptbhue,ndthuc,dtbthuy}@fit.hcmus.edu.vn

[2]National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
{wohlgemuth,iechizen}@nii.ac.jp

Abstract

Despite the fact that health care providers such as Google Health and Microsoft Corp.'s Health Vault comply with the U.S Health Insurance Portability and Accountability Act (HIPAA), the privacy of patients is still at risk. There needs to be a strategy for securing the privacy of patients when exchanging health records between various entities over the Internet. Several encryption schemes and access control mechanisms have been suggested to protect the disclosure of a patient's health record especially from unauthorized entities. However, by implementing these approaches, data owners are not capable of controlling and protecting the disclosure of the individual sensitive attributes of their health records. This raises the need to adopt a secure mechanism to protect personal information against unauthorized disclosure. We propose a new column-level access control mechanism that is based on subkeys, which would allow a data owner to further control the access to his data at the column-level. We also propose a new mechanism to reduce the number of keys held by each user in the system. Therefore, the number keys maintained by the data owner is also reduced. The experimental results show that our proposals can ensure system's availability. So they are applicable in the real world.

Keywords: *database security, column-level access control, database encryption.*

1. Introduction

Health information today, which is mainly stored on paper, is scattered and disconnected. An EHR system is a concept defined as a sysmtematic collection of electronic health information about individuals that can be shared via the Internet across various health care settings such as: hospitals, clinics, and pharmacies etc. EHR systems provide online services where patient's health records are created, used, exchanged, stored, and retrieved. There are three main entities in the EHR scenario: the *data owner* - i.e. patient who is the subject of the health records made available for controlled external use; the *user* – individual or organization that requests data from the EHR system; and the *server* - an organization that receives the data sent from the data owners and makes it available for distribution to users.

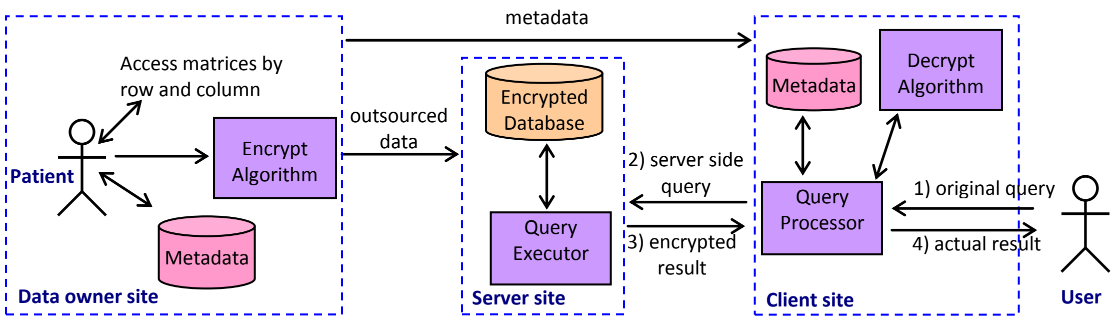


Fig. 1 Block diagram of an EHR system

In EHR systems, however, the patients' data is stored on an external server, which is typically not under their control. It is assumed that this server cannot be trusted with the confidentiality of the database's content. It is important to protect the health data of patients from unauthorized access by intruders and even the server's operators.

In existing EHR systems such as Google Health or Microsoft HealthVault, patients cannot control the providers' usage of their personal data. Database encryption was regarded as a solution for protecting data from unauthorized disclosure. Existing encryption schemes assume that the client has complete access to the query results ([11], [12]). Such encryption schemes do not fit applications such as the EHR system, where the data owner often requires the enforcement of access restriction to different users. Some access control mechanisms have recently been proposed, but by using them, the data owners can only control access at the tuple-level and they are not able to restrict access to some of the more sensitive attributes of the data being shared ([1], [3], [4], [5], [6], [7]).

The contributions of this study are: (a) a new access control mechanism which would allow a data owner to further control the access to his data at the column-level, (b) a new mechanism to reduce the number of keys maintained by the data owner or user in the system, (c) experimental evaluation of our proposals.

2. Privacy Requirements by Legislation

Privacy is described as "[...] the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent of information about them is communicated to others" [19]. This description of privacy as informational self-determination has been a basis for health data protection regulations such as HIPAA [13], European Data Protection Direction 95/46/EC [8], and Data Protection Act of Japan [15]. From them, and based on [10], we derive the following privacy requirements for an EHR system:

Requirement 1: Every patient should be able to control the flow of information related to himself. This requirement is very important for information systems to preserve patients' privacy as defined above. Since patient's records are stored on a service provider's site and the patient is not given enough information about the provider to decide if they are trustworthy, this leads to

Requirement 2: Patients should not be forced to trust any party except ones directly involved with their treatment. Accessing entire medical records is not always essential. Any intended purpose of use, disclosure, or request of protected health information should be at a necessary minimum. This is stated by

Requirement 3: Patients can limit usage and disclosure of their personal records to the necessary minimum. Linking information from several data requests could possibly generate unintended information about patients. This results in

Requirement 4: Access requests to the patients' data cannot be used to establish the profiles or additional knowledge about patients. Although the privacy mechanisms are implemented, they have no value if the user cannot verify that they work correctly. To counter this problem we state

Requirement 5: User should be able to verify that the result returned from the untrustworthy server is trustworthy.

Security solutions used in an EHR system must fulfill the above-mentioned five requirements.

3. Attack Model and Privacy Problems

Patients are required to trust the provider when using existing EHR systems such as Microsoft HealthVault [17] or Google Health [9]. They cannot control whether the providers really adhere to their promises to protect their data and disclose personal data only to parties the patients have agreed to. So, Microsoft HealthVault and Google Health fail to guarantee requirements 1 and 2, and thus, any of the remaining requirements. The encryption scheme proposed by the authors in ([11], [12]) can protect data from unauthorized access and fulfills requirements 1 and 2. However, this scheme used one key to encrypt the whole database. It means that it was designed for the simple model of outsourcing service where the data owner is also the unique user to request the data from the service provider. It assumed that a user always has full access to the outsourced database. Therefore, it does not fulfill requirement 3 and cannot be used in EHR systems in which the users should have different access privileges to the patients' health records and in which sensitive information needs to be limited in use and disclosure. Moreover, using the proposed access control mechanisms ([3], [4], [5], [6], [7]), patients are unable to restrict access to some sensitive columns, so requirement 3 is not guaranteed. Requirements 4 and 5 must be fulfilled by solutions for ensuring user privacy and query assurance, respectively, and they are out of the scope of this paper. In this paper, we address the first three privacy requirements. It raises the questions: "*What encryption scheme is suitable for column-level access control by the data owner?*" and "*How to manage the access control policies in an EHR system?*"

4. Related Work

Existing access control approaches combined cryptographic protection and authorization access control and enforced access control via selective encryption, which meant users can decrypt only the data they are authorized to access ([1], [3], [4], [5], [6], [7]). Most of these authors exploited a structure called the user tree hierarchy to represent the relationship between users and information items ([1], [6], [7]). The keys for accessing data of users at lower-level nodes are based on the keys of their predecessors by using a family of one-way functions [18]. The complexity of the algorithm for building the tree hierarchy is exponential and it needs reconstruction after most of the modifications of the access control policies. Zych et al. [5] presented a key management scheme that was based on a partial order between the user groups and used the Diffie-Hellman key generation algorithm for the key derivation. The algorithm for constructing the hierarchy and the key generation process were very complex and costly. De Capitani di Vimercati et al. [3] proposed a two-layer access control mechanism that was also based on the application of selective encryption. This solution had the benefits of not having to re-key when authorization policies were updated, but it needed two times of encryption which were very costly. El-khoury et al. [4] suggested a key management scheme using a structure called binary trie. A binary trie was constructed based on the access control policies which are modeled via an access matrix. Keys ring for each group of user were generated based on this binary trie structure. The key management complexity was reduced and most of the data access policy updates did not require significant changes to the structure, which reduced the number of key generation and data re-keying times. By some means or other, however, all of the above-mentioned encryption schemes and access control mechanisms can only support tuple-level access control and fail to guarantee requirement 3. By using them, the data owners cannot restrict access to some of the more sensitive attributes of the shared data or have to partition the relation containing the shared data into fragments for sharing in full. This creates a lot of relations, and thus creates difficulties in effectively managing the database and query processing.

5. Approach for Column-Level Access Control in EHR Systems

The database encryption with subkeys first proposed by Davida et al. [2] had the advantages of record orientation, security, and each field can be accessed under a different subkey. Using it, a user can read only some given fields depending on the granted subkeys he has without revealing all the attributes' values. We suggest using the encryption scheme proposed by Hacigümüs et al. [11] for EHR systems, but with a modification of the encryption function. The encryption scheme with subkeys is used instead of the block cipher techniques, such as AES, DES or RSA as in original scheme. For each relation R(A_1, A_2, ..., A_n), the data owner stores encrypted relation R^S(*etuple*, A_1^S, A_2^S, ..., A_n^S) on the server, where the attribute *etuple* is the encryption form of a record using the encryption scheme with subkeys. The data owner follows these steps to protect his data:

Step 1: He firstly describes the access policies to their data by using an access matrix by row (see Table 1). The access matrix by row A has the number of columns equals to the number of row of the considering relation, and number of row equals to the number of users in the system. A[i, j] = 1 if user i is allowed to access to the column j; 0 otherwise.

Step 2: He then groups the columns of A into disjoined row categories and groups the rows of A into disjoined user groups. Each row category consists of rows on which all the users have the same access permission. Each user group consists of users who have the same access permission to the rows of the considering relation.

Step 3: He constructs the binary trie [4] (see Fig. 2) corresponding to the access matrix obtained after step 2 to decide the number of keys that will be distributed to each group of users. The keys assigned to the user group at the leaf-level of the binary trie are used to encrypt/ decrypt the data. They are the encryption keys, and also the decryption keys. The keys granted to the users at the non-leaf level of the trie must be able to be used for deriving to the necessary decryption keys for reading the data with granted permission (see Table 2). The data owner restricts the access to the more sensitive columns by giving out only the subkeys corresponding to the columns that the user has access permission to. The key derivation method, which is described in section 6, is used for derivation.

Step 4: If the users in a group G have different access privileges to different columns of a row category R, the data owner describes the access policies using other kind of access matrix, called the access matrix by column. The number of rows of this matrix is equal to the number of subgroups of users in the group *G* and the number of columns it has equals to the number of groups of columns in R with different access authorizations. We construct the binary trie structure corresponding to this access matrix by column. Each subgroup of users holds just some subkeys. Appropriate subkeys are communicated to users by the data owner via a secure channel. These subkeys are used for deriving the encryption/ decryption subkeys in order to access the columns with the granted permission. The derivation is done by using the one-way function which takes in input an integer and outputs an integer for none-leaf level user groups, or by using the one-way function which takes in input an integer and outputs a prime for leaf-level user groups as above-mentioned. Using our proposed approach, the patient's data is protected from unauthorized disclosure and the patient can restrict access to the sensitive columns of the data being shared. Our proposed approach fulfills requirements 1, 2, and 3 of privacy requirements by legislation (section 2).

Table 1. Access matrix by row

	r_1	r_2	r_3	r_4	r_5
g_1	0	1	1	0	0
g_2	1	1	0	1	1
g_3	1	1	1	0	0
g_4	1	1	1	1	0

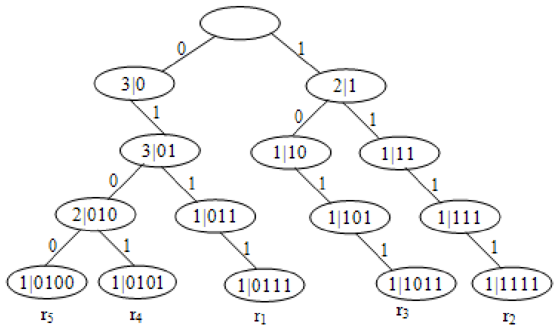


Fig. 2 Binary trie structure of the access matrix by row

Table 2. Keys held by each group of users and keys must be derivable

User group	Keys held by each group of users	Keys must be derivable
g_1	k_1	k_{1111}, k_{1011}
g_2	k_{01}, k_{11}	k_{0111}, k_{1111}, k_{0101}, k_{0100}
g_3	k_{011}, k_{101}, k_{111}	k_{0111}, k_{1111}, k_{1011}
g_4	k_{0111}, k_{1111}, k_{1011}, k_{0101}	

6. Key Management for Column-Level Access Control in EHR Systems

In this section, we present the procedure to encrypt the data in order to control access at the column-level. We suggest the key derivation method for reducing the number of keys maintained by each user in the system while still ensuring the properly working of the system. The key storage strategy at the end of this section is necessary for minimizing the amount of secret information held by each user in the system.

Database Encryption with Subkeys: There are variations when using the Chinese Remainder Theorem (CRT) for encrypting a database ([2], [14], [16]). In EHR systems, the convenience for a user to access data at a service provider is important, therefore an encryption scheme requiring less keys held by each valid user was chosen. We use an encryption with subkeys developed by Davida et al. [2] in EHR systems.

Suppose that there are m records in a given relation r consisting of n fields. The i^{th} record of r is of the form $x_i = (x_{i1}, \ldots, x_{in})$. Let C_i be the cipher text of x_i. The encryption procedure is done by forming:

$$C_i = \sum_{j=1}^{n} e_j (r_{ij} \| x_{ij}) \bmod D, \text{ for i = 1, …, m; } D = \prod_{j=1}^{n} d_j \text{ , j = 1, …, n}$$

where $e_j = (D/d_j)b_j$, b_j is the multiplicative inverse of (D/d_j) with moduli d_j. Each subkey d_j is a prime such that $a_{ij} < d_j$, $a_{ij} = (r_{ij} \| x_{ij})$, $\|$ is the concatenation, r_{ij} is the random value for the field j which is used for preventing attacks originated from using CRT. d_j, j = 1, …, n, is called the reading subkey for the field j^{th}. The decryption procedure is: $(r_{ij} \| x_{ij}) = C_i \bmod d_j$, j = 1,…,n. By discarding the random bit r_{ij}, one can obtain the j^{th} field data x_{ij} of the record i^{th}.

Key Derivation Method: In the encryption scheme with subkeys, each key consists of multiple different subkeys $(d_{i1}, d_{i2}, \ldots, d_{in})$. Each subkey d_{ij} of a key granted to the users at the non-leaf level SC_i of the binary trie is a large integer. In the case when SCi is at the leaf level of the trie, each subkey of SCi is a prime for properly using the CRT. The key of security class SC_j, which is an immediate child of SC_i, is computed by $(f_{j1}(d_{i1}), f_{j2}(d_{i2}), \ldots, f_{jn}(d_{in}))$, each f_{ji} is a one-way function [18]. In our context, we need two kinds of function f_{ji}. The first one is the one-way function that takes in input an integer and outputs an integer. This kind of function is used for deriving keys of the non-leaf level security class. The second one is also the one-way function which receives an integer as the input but outputs a prime, see appendix. This kind of function is used for deriving keys for the leaf-level security class.

Key Storage: In our proposed access control mechanism, each user owns some subkeys which are used for accessing or deriving to the necessary subkeys to access the fields he has access authorization. Management a number of subkeys is difficult for users. We propose using the key management method in [14] to handle this problem. Suppose user i is granted n subkeys:

(1) Assigns each subkey a public prime number p_j, for j =1, 2, …, n.

(2) Computes the secret master key MK_i by CRT for user i:
$MK_i = d_j \bmod p_j$ for some j, $1 \le j \le (n+1)$ where d_j is the reading subkey; d_{n+1} and p_{n+1} are the random secret key and the prime assigned to this subkey which are used for preventing users from colluding to disclose the secret master key of user i.
User i keeps only the secret master key MK_i.

(3) To read the subkey j^{th}, user i computes $d_j = MK_i \bmod p_j$ to get subkey d_j.

7. Experimental Evaluation

We implemented a tool that takes in input a relation and we investigated four aspects of our proposals: database encryption time, respond time (which includes decryption time), key derivation time and the reduction of number of keys which are held by the data owner or users. Our experiment was carried out on a 3.18GHz E8500 Core2 with 2.00GB of RAM.

Relevant software components used were Microsoft SQL Sever 2005 as database management system and Windows XP SP3 as the operating system.

We created the `Employee` table which consists of five attributes: `EmpID` (the identity of the employee), `Name` (the fullname of the employee), `Birthday` (the day of birth of the employee), `Salary` (the salary of the employee) `and DepNo` (the identity of the employee's department). The datatypes of these attributes are integer (`int`), string (`nvarchar(30)`), datetime (`datetime`), float (`float`) and integer(`Int`) respectively. Data is generated randomly and inserted into the `Employee` relation. For each of the first two purposes of investigation, we conducted the tests five times with the increasing number of database's tuples: 1000, 2000, 5000, 10000 and 15000 with the capacity of 600KB, 640KB, 784KB, 1032KB and 1376KB respectively.

For the first purpose, we investigated the encryption time of our proposed encryption scheme. We ran the CRT algorithm and encryption algorithms suggested by Hacigümüs et al. [11] (such as RSA, AES or DES) on the original data with the increasing number of tuples and reported the encrypting time. Figure 3 (Fig. 3) shows the result of the test. It is shown that it needs more time for the CRT algorithm to complete the work. The first reason is that encrypting database using CRT is finding the common solutions for the congruences which costs much time. The second reason is that CRT algorithm encrypts the database at the column-level while the other algorithms do this work at the row-level. By using CRT to encrypt the database at the column-level, the data owner can restrict access to the more sensitive columns of his data. Although the encryption time of CRT algorithm is rather high, the respond time which includes the decrypting time is reasonable (see the next experimental result). In reality, database encryption is often to be an offline process. Reasonable respond time ensures system's availability. So our proposed encryption scheme is applicable in the real world.

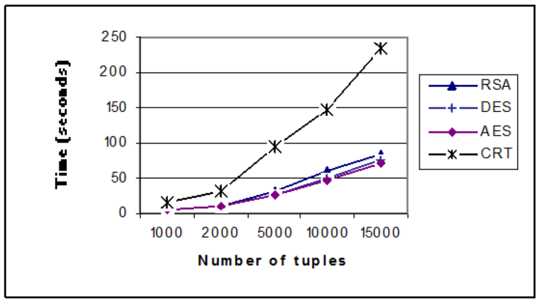


Fig. 3 Encryption time of CRT comparing with RSA, DES and AES

Secondly, we investigated the respond time of our proposed encryption scheme. We executed two queries with different set of selected attributes on the cipher data created by using different encryption algorithms and recorded respond time of each of them. The first query Q1 did a projection on one attribute (`EmpID`) over the database while the second query Q2 did a projection on five attributes (`EmpID, Name, Salary, Birthday, DepNo`).

```
Q1: Select EmpID
    From Employee

Q2: Select EmpID, Name, Salary, Birthday, DepNo
    From Employee
```

According to Hacigümüs et al. [11], the respond time to a query in the EHR scenario consists of query translation time, query execution time, communication time, decrypting time and final result selection time. This amount of respond time differentiates from the other one mainly on the decrypting time. Respond time to queries executed on the database encrypted by RSA, or DES, or AES is not different between Q1 and Q2. The reason is that they encrypt/ decrypt data the whole at all. The highest respond time is of the queries executed on the database encrypted by RSA. By using CRT, it costs more time for responding to the query which projects on more attributes because CRT encrypts and therefore, decrypts the database at the column-level. Each attribute appears more after the select statement needs more time to do one modulo operation according to the decryption algorithm of CRT. The respond time increases slowly between one and five-attribute projection: 0.594 to 0.692 (with 1000 tuples database), 1.906 to 2.750 (with 2000 tuples database), 7.063 to 8.101 (with 5000 tuples database), 23.203 to 23.637 (with 10000 tuples database) and 45.578 to 49.346 (with 15000 tuples database). The respond time to Q1 and Q2 are presented in tabular form in table 3 (Table 3 (a) and Table 3 (b) for Q1, Q2 respectively) and graphically in figure 4 (Fig. 4).

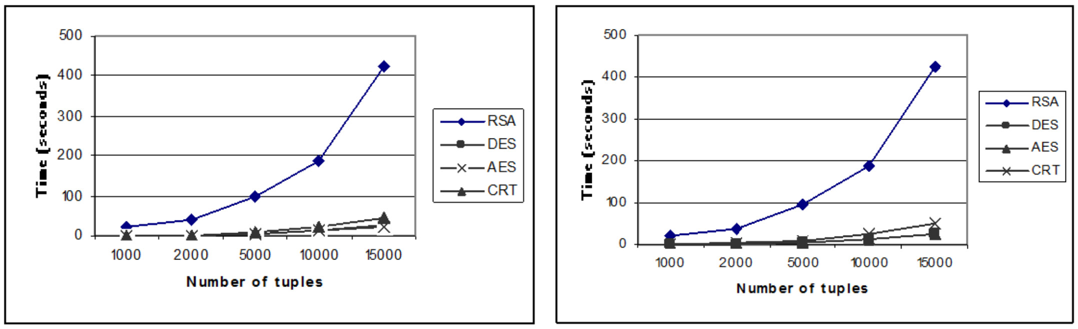
Table 3. Tabular representation of respond time to Q1 (a) and Q2 (b)

	RSA	DES	AES	CRT
1000	22.125	0.684	0.647	0.594
2000	38.519	1.606	1.531	1.906
5000	96.738	5.444	5.109	7.063
10000	186.122	12.793	11.538	23.203
15000	425.141	25.053	23.297	45.578

(a)

	RSA	DES	AES	CRT
1000	22.125	0.684	0.647	0.692
2000	38.519	1.606	1.531	2.750
5000	96.738	5.444	5.109	8.101
10000	186.122	12.793	11.538	23.637
15000	425.141	25.053	23.297	49.346

(b)



(a) **(b)**

Fig. 4 Graphical representation of the respond time of Q1 (a) and Q2 (b)

The next experiment, whose results are reported in table 4 (Table 4) and figure 5 (Fig. 5), was devoted to evaluate key derivation time. This tool received the access matrix as input and outputted the number of keys that will be used by the data owner to encrypt the database and to distribute to each user group. The data owner follows our proposed security mechanism to manage the keys. The users at a security class that is not the leaf-level of the binary trie will use the granted keys to derive all the necessary keys for accessing the granted data. We investigated the case that having 10 user groups and 10 resource categories. It means that the input matrix for constructing the binary trie having the size of 10 × 10. We generated the access matrices randomly with the probability of occurring 1 is 50%. Note that this access matrix is the one obtained after applying step 2 of our proposed key management mechanism. We regenerated the access matrix if there are at least two user groups having the same capability or two resources on which the users having the same privileges. (In that case, after applying step 2 of our proposed key management mechanism, the access matrix has one of its two sizes or both of them less than 10.) We then constructed the binary trie structure for this access matrix and assigned keys to each user group. We ran the tool 10 times on 10 different access matrices and got the average time of key derivation at each level of user group. In our experiment, the key consisting of three subkeys (which means that we considered the relation with three attributes). Each key k_i assigned to the none-leaf security class SC_i is of the form (k_{i1}, k_{i2}, k_{i3}) where k_{ij} is a large integer. Each key assigned to leaf -level security class SC_j has each subkey is a prime. The key derivation process is done by using one-way hash function, appendix.

Table 4. Average time of key derivation at each level of user group

	User group level									
	1	2	3	4	5	6	7	8	9	10
Average time (seconds)	11.71	7.43	21.96	16.49	5.57	6.49	8.43	5.67	4.65	0

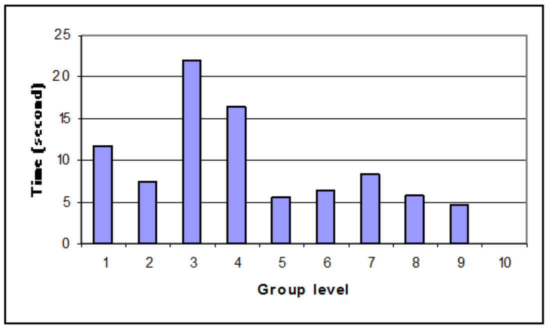


Fig. 5 Average time of key derivation (in seconds) at each level of user group

What we observed is that key derivation time of a user group not only depends on the number of keys that user group has to hold (for accessing the granted data) but also depends on the level of that user group on the binary trie. The key derivation time of the leaf-level user group is 0.

The final experiment was for investigating the reduction of a number of keys which are held by the users or the data owner in the system. Firstly, we investigated the number of keys held by each level of user group. This tool received an access matrix as input and outputted the number of keys held by a user at different level of the binary trie structure which is used for managing the keys. We generated five access matrices randomly, each is a 20 × 500 matrix. It means that there are 20 user groups and 500 row categories in the system. The probability of occurring 1 in each access matrix is 50%. These access matrices are generated by the same manner as in the third experiment. We ran the tool five times on five different access matrices and reported the number of keys held by each user group level of the binary trie. The graphic representation of this result is in figure 6 (a) (Fig. 6 (a)). What we observe is that the deeper the user group level that a user belongs to the more keys that user has to hold in order to access the data with granted permission. But the number of keys held by each user group seems to be stable when the user group level reaches the leaf level of the binary trie. The number of keys the data owner has to maintain is the number of keys held by all the user groups in the system in total. In our experiment, the data owner has to maintain 3083, 3005, 3010, 3059, 3046 keys for each time of running the tool, respectively. Because the input access matrix has 20 rows and 500 columns, with the probability of appearing 1 is 50 %, the number of keys which are necessary for accessing data is (20×500)×50% = 5000. As the result, the number of keys the data owner has to maintain is about 60% of all the necessary keys (see Fig. 6 (b)). This benefit is obtained because of our usage of the binary trie to manage the keys. A user belonging to some user group level in the binary trie has to hold only some keys and he is able to derive the remaining keys in order to access the data with granted permission. The number of key a user has to hold is always less than or equal to the number of key that user needs. Therefore, our approach can also reduce the number of keys held by each user group.

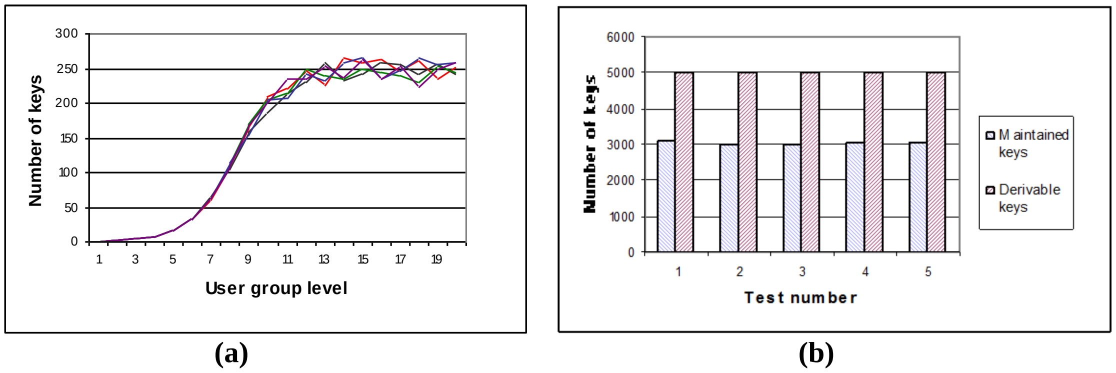


Fig. 6 Number of keys held by each user group level (a) and the reduction of number of keys maintained by the data owner (b)

Although the number of keys maintained by the data owner or the number of keys held by a user group is reduced by our proposed approach, the data owner or user still has to hold a rather high number of keys. Especially, the user groups at the low level of the binary trie often hold a high number of keys; but the nearer the bottom of the structure the groups are, the faster the derivation time to obtain the necessary keys is. We suggest a storage strategy for storing a number of keys securely and conveniently (section 6). Existing proposals can reduce the number of keys held by the data owner or users to single ([5], [6]), but they are very costly and inconvenient in the dynamic scenario.

8. Conclusion and Future Work

We have proposed a new column-level access control mechanism based on subkeys. It allows a data owner to further control the access to their data at the column-level. We also suggested a key management mechanism to reduce the number of keys held by a data owner or user in the system.

We devoted the experiment to investigate four aspects of our proposals: the encryption time, the query respond time, key derivation time and the reduction of number of keys which are held by the data owner or users. The experimental results show that our proposals ensure system's availability. Therefore, they are applicable in the real world.

As the future work, we will investigate the dynamic cases of access control in which the users or resources or privileges changed frequently. We will also improve the system's efficiency.

References

[1] Damiani, E., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, & S., Samarati, P.: Key Management for Multi-User Encrypted Databases. Proc. of the 2005 ACM Workshop on Storage Security and Survivability, pp.74--83 (2005)

[2] Davida, G.I., Wells, D.L, & Kam, J.B.: A Database Encryption System with Subkeys. ACM Transactions on Database Systems, vol. 6, No. 2, pp. 312--328 (1981)

[3] De Capitani di Vimercati, S., Foresti, S, Jajodia, S., Paraboschi, S., & Samarati, P.: Over-encryption: Management of Access Control Evolution on Outsourced Data, VLDB, pp. 123--134 (2007)

[4] El-khoury, V., Bennani, N., & Ouksel, A.M: Distributed Key Management in Dynamic Outsourced Databases: a Trie-based Approach, First Int. Conf. on Advances in Databases, Knowledge, and Data Applications, pp. 56--61 (2009)

[5] Zych, A., Petkovic, M., & Jonker, W. Efficient key management for cryptographically enforced access control, Elsevier Science, pp. 410--417 (2008)

[6] Atallah, M. J., Frikken, K. B., and Blaton, M.: Dynamic and Efficient Key Management for Access Hierarchies, ACM, (2005)

[7] Damiani, E., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, & S., Samarati, P.:An Experimental Evaluation of Multi-key Strategy for Data Outsourcing, Springer, IFIP, (2007)

[8] European Commission, Directive 95/46/EC of the European Parliament and of the Council of 24 Oct. 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, Official Journal of the European Communities, L 281, 395L0046, pp. 31--50 (1995)

[9] Google, Health Privacy Policy, http://www.google.com/intl/enUS/health/privacy.html

[10] Haas, S., Wohlgemuth, S., Echizen, I., Sonehara, N., & Müller, G.: On Privacy in Medical Services with Electronic Health Records, IMIA SiHIS, CoMHI (2009)

[11] Hacigümüs, H., Iyer, B.R., Li, C., & Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model, in SIGMOD, pp. 216--227 (2002)

[12] Damiani, E., Vimercati, S.D.C., Jajodia, S., Paraboschi, S., & Samarati, P.: Balancing confidentiality and efficiency in untrusted relational DBMSs. Proceedings of the 10th ACM Conference on Computer and Communication Security, Washington DC, USA, pp. 93-102 (2003)

[13] Health Insurance Portability and Accountability Act of 1996 (HIPAA) Privacy Rule.

[14] Hwang, M.S., & Yang, W.P.: A Two-Phase Encryption Scheme for Enhancing Database Security, J. Systems Software, Elsevier Science, pp. 257--265 (1995)

[15] Japanese Government: Act on the Protection of Personal Information, http://www5.cao.go.jp/seikatsu/kojin/foreign/act.pdf (2005)

[16] Lin, C.H., Chang, C.C., & Lee, C.T.: A record-oriented cryptosystem for database sharing, in Int. Computer Symposium, pp. 328--329 (1990)

[17] Microsoft, HealthVault Privacy Policy, https://account.healthvault.com/help.aspx?-topicid=Privacy-Policy, (2009)

[18] Sandhu, R.S.: Cryptographic implementation of a Tree Hierarchy for access control, Elsevier, pp. 95--98 (1988)

[19] Westin, A.F.: Privacy and Freedom. Atheneum, New York (1967)

Appendix

We present a method for constructing a one-way function that receives an integer as an input and gives a prime as an output. Using this algorithm, the users with the same privileges will generate the same keys at different times for deriving keys to access the specific data.

Theorem

Assume that (n -1) = F×R = $\left(\prod_{i=1,\dots,s} p_i^{a_i}\right) \times R$, where R < $\sqrt{n}$, and an integer b such that $b^{n-1} \equiv 1 \pmod n$ exists and $\gcd(b^{(n-1)/p_i} - 1, n) = 1$, i = 1,…, s. Then n is a prime.

Algorithm G: Generating a prime of d digits with a given seed s.
(1) Generate Q that has (d-2) digits using the consecutive hash values of seed s:
Q = h(s)||…||h(s), where || is the concatenation operator.
(2) Find the greatest R such that (R < Q) and (1 + RQ ≡ 1 (mod 6)).
(3) If p = RQ + 1 is a pseudo-prime with base 2 and 3
(4) Then return p.
(5) R = R+6.
(6) Goto (2).

Key deriving function

Let h be a one-way hash function h: $Z_n \rightarrow H$, i.e. given y∈H, it is hard to find $x \in Z_n$ such that h(x) = y, where H is the hash value space. For example, MD5 and SHA are two popular one-way hash functions. Let g be a generating prime function g: H → P, where P is a space of primes. Let f: $Z_n \rightarrow P$ be defined by f = goh, then f is a one-way function. Indeed, given y = f(x) = g(h(x)), because h is a one-way hash function, it is hard to find $x \in Z_n$ such that y = f(x) = g(h(x)). To generate a prime, we use the combination of the above generating prime function and the hash function SHA.

Algorithm F: Generating key based on a given integer p.
(1) Compute s = SHA (p).
(2) Generate a prime q using the algorithm G with seed s.

Authors

Pham Thi Bach Hue is now a PhD candidate at the University of Science, Vietnam National University – Ho Chi Minh City (VNU-HCMC). She is interested in information security, especially in the scenario of database outsourcing service. She has published several papers in this research area. She received the Bachelor and Master degrees in computer science in 1998 and 2002 at the University of Science, VNU-HCMC. She was an internship student at the National Institute of Informatics (NII), Japan in 2009. She is currently a lecturer of the Department of Information System, Faculty of Information Technology, University of Science, VNU-HCMC.

Sven Wohlgemuth has received his doctoral degree in computer science from the Albert-Ludwig University Freiburg, Germany in 2008. Since 2009 he has been a postdoctoral scholar at the National Institute of Informatics, Japan (NII) funded by German Academic Exchange Service (DAAD) and the Japan Society for the Promotion of Science (JSPS). In 2003 the German Federal State Baden-Württemberg has awarded his work on security and usability by identity management with a doIT Software Award. His paper on "On Privacy in Medical Services with Electronic Health Records" has been awarded with the Gerd Griesser Award 2009. His paper on "Tagging Disclosures of Personal Data to Third Parties to Preserver Privacy" has been selected as one of the best papers by the PC chairs of the IFIP SEC2010 conference. He was member of the organizing committees of several conferences and workshops regarding information systems for social innovation as well as on IT-Security and privacy, e.g. "International Conference on Emerging Trends in Information and Communication Security (ETRICS) 2006", "SICHERHEIT 2008", and of the ISSI series.

Isao Echizen received B.S., M.S., and D.E. degrees from the Tokyo institute of technology, Japan, in 1995, 1997, and 2003. He joined Hitachi, Ltd. in 1997 and until 2007 was research engineer in company's systems development laboratory. He is currently an associate professor of the National institute of informatics (NII) and a visiting professor of the University of Freiburg. He has been engaged in research on content security, copyright protection technologies, and multimedia application systems. He received the President's technology Award from Hitachi in 2000, the Best paper award from IPSJ in 2005, the Best paper award of IEEE IIHMSP in 2006, the Fujio frontier award and the Image electronics technology award in 2010, and the One of the best papers on IFIP SEC and the IPSJ Nagao special researcher award in 2011.

Ass. Prof. **NGUYEN Dinh Thuc** received the Ph.D. degree in computer science from the University of Science, Vietnam National University – HCMC, in 2000. He has been Professor of Cryptography and Computer Security at Faculty of Information Technology, University of Science, VNU– HCMC. His current research interests include cryptography, authentication and access control, database security and signal processing. He has been a Technical Program Committee member for many IEEE sponsored and co-sponsored workshops, symposiums and conferences. He has been Co-organizer of ICTACS2006, ICTACS2009 and ICTACS2010.

Dong Thi Bich Thuy received the B.S. in Management from the school of HEC - University of Lausanne- Switzerland (1975), then the Ph.D. in computer science from the University of Geneva (1986). Her research interests are business process pattern modelling, intelligent information systems including technologies used in recommender systems or question-answer systems, new approaches in querying databases such as mobile and secure queries. Some of her research activities are doing in collaboration with others research laboratories from European and Japanese Universities and Intitutions (Database group of the University of Geneva; CLIPS of the University Joseph Fourier, Grenoble – France; IRIT Toulouse, France; NII Tokyo, Japan). She is currently an associate professor in computer science at the University of Science, Vietnam National University – Ho Chi Minh City.