The Design and Implementation of Database Encryption

Noor Habibah Arshad, Saharbudin Naim Tahir Shah, Azlinah Mohamed, Abdul Manaf Mamat

***Abstract*—** Information inside the database is shared by multiple parties such as internal users, partners, contractors and others. Sensitive data stored in database could be a target to attackers. The attacker for data stored in database not only from external but also from within the organization. Adding the database encryption, valuable information in database becomes more secure since the encrypted data ensure the confidentiality of the data. A new affine block cipher named Enhanced Affine Block Cipher technique is proposed for database encryption. This algorithm improves the weakness of the original affine cipher. The new encoding schema and modification Cipher Block Chaining (CBC) mode of operation for block cipher is designed for the new algorithm and then the prototype of the system is built and implemented into existing system for protecting user password. The result has shown that the algorithm is working properly, where the decryption process produced similar output as the original plaintext and it ran through specified configuration and evaluated thoroughly with respect to database approach and algorithm technique to prove the design.

***Keywords*—**Database, Enhanced Affine Block Cipher, Encryption, Decryption.

I. Introduction

In today's enterprise environment, database systems are distributed and used in various applications such as e-Business and e-Commerce. These applications are examples of real-time online resources that need to deliver value-added services through high confidentiality and availability of databases. With a networked database in the complex multi-tiered applications, multiple parties such as partners, contractors, internal users and others will share the information inside the database. The sensitive data could be a target to attackers. The attacker for data stored in database not only from external parties but also from internal. However, their vulnerability to external attack increases and critical business data stored in databases are obviously vulnerable for attackers. Therefore, to properly maintain the integrity and confidentiality of the data, database security becomes one of the most urgent challenges in database research. Database security is a wide research area and includes topic such as statistical database security, intrusion detection and most recently privacy preserving data mining [4].

There are some techniques have been done to protect data stored in database such as firewall, intrusion detection system (IDS) and access control but this is not enough. Databases are still vulnerable to be attacked from internal and external threats. Firewall and IDS only provide network layer protection. Access control is based upon the concept of privilege and it is basic for many security features [2]. One of the requirements for database security is database encryption. With database encryption, the valuable information in database becomes more secure since the encrypted data ensure the confidentiality of the data.

Thus, this paper will focus specifically on some of the details on cryptographic algorithm technique used to implement the database encryption. Throughout this paper, the cryptography algorithm that will be used to provide security and confidentiality of data in the database are discussed and elaborated.

II. Database Encryption

In general, database sharable resource among many user or applications. A multiuser application in distributed system complicates the data security problem imposed upon a database. Hence, security is becoming one of the most urgent challenges in database research and industry. Past studies reviewed that database security is the most common architectures and methodologies for designing secure database [7, 5]. One of the important aspects of database security is database encryption [1, 2, 10].

The original data that is readable and understood is called plaintext or cleartext. Method that used to code a plaintext that can conceal its meaning is called encryption. Once a message has been transformed with an encryption algorithm, the resulting message is called ciphertext. The encryption is used to ensure that information is hidden from unintended person, even from those who can see the encrypted data. In order to be able to read ciphertext, the other process is needed to decipher the ciphertext. The study of encryption and decryption is called cryptography [9]. According to Menezes *et al.* [6] and

Authors are with Faculty of Information Technology & Quantitative Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, MALAYSIA, habibah@tmsk.uitm.edu.my, naim@tmsk.uitm.edu.my, Manuscript received April 18, 2007; Revised Received September 23, 2007

Russell and Gangemi [9] cryptography provides security in the areas of confidentiality, data integrity, authentication, and non-repudiation.

The goal of encryption is to make data unintelligible to unauthorized users and extremely difficult to decipher when it is attack. Symmetric key cryptography is the most commonly used technique to encrypt data in the storage or database. This ciphers use the same key when to encrypt and decrypt the data. There are two types of symmetric ciphers; block ciphers and stream ciphers. Stream ciphers are generally twice as fast as block ciphers but they require the use of unique keys. Block ciphers on the other hand, allow keys to be reused. There are some encryption features of block cipher technology were included in Database Management System (DBMS). The recommended minimum key length for all symmetric key ciphers is 128 bits.

A block cipher is a type of symmetric key cryptography that transforms a fixed length block of plaintext (unencrypted text) data into a block of ciphertext (encrypted text) data of the same length. This transformation takes place under the action of a user-provided secret key. The fixed length is called the block size, and for many block ciphers, the block size is 64 bits. Meanwhile, Cipher Block Chaining (CBC) mode adds a feedback mechanism to the encryption scheme. In CBC, the plaintext is exclusively-ORed (XORed) with the previous ciphertext block prior to encryption. In this mode, two identical blocks of plaintext never encrypt to the same ciphertext.

Decryption is performed by applying the reverse transformation to the ciphertext block using the same secret key. An early and highly influential block cipher design was the Data Encryption Standard (DES), developed at IBM in 1974, and published as a standard in 1977. A successor to DES, the Advanced Encryption Standard (AES), was adopted in 2001 [12].

The affine block cipher [11] is one of the symmetric key cryptography that was known as classical cryptography and it is easier to break by ciphertext-only cryptanalysis. Some improvements have been done on affine cipher. Instead of using single letter, Koblitz [3] shows digraphs in his works but it is still not enough because of the second letter of each ciphertext digraph depends only on the second letter of the plaintext digraph. Thus, one could obtain a lot of information keys from a frequency analysis of the even numbered letters of the ciphertext. In this paper, enhanced affine block cipher algorithm with its encoding schema was designed to overcome affine cipher and it was implemented in securing data stored in database.

III. Database Encryption Approach

There are two main approaches for database encryption which is whether performing encryption and decryption inside the database or performing encryption and decryption outside the database [4, 8]. After reviewing the database encryption, the best ways to secure the information stored in database is database encryption and apply it at outside the database i.e. at application level encryption. This approach was selected because it provides good end-to-end data protection. By using this approach, encryption will be on the column and row basis. Hence, not all data stored in the database will be encrypted. Only sensitive information such as user identification, credit card number and password will be encrypted. By applying this approach, it will be more efficient in reducing the overhead of reading data. The cryptographic algorithm used for the database encryption is designed and implemented in java programming language and it acts as application server whereby the encryption and decryption processes are done at the application level.

This approach applied end-to-end encryption between client and applications server. For encryption process, the data is encrypted at application server and then inserted into the appropriate fields or columns in the database. For decryption process, the encrypted information is retrieved from the database and then decrypts it at application server so that only authorized user can see the information. The keys used to encrypt and decrypt the data in this approach is stored in file storage at application server not in the database. Hence, this approach will add one security layer in securing the data stored in the database. The keys must be found before the attacker can see and know the contents of data. Figure 1 depict database encryption outside the database.

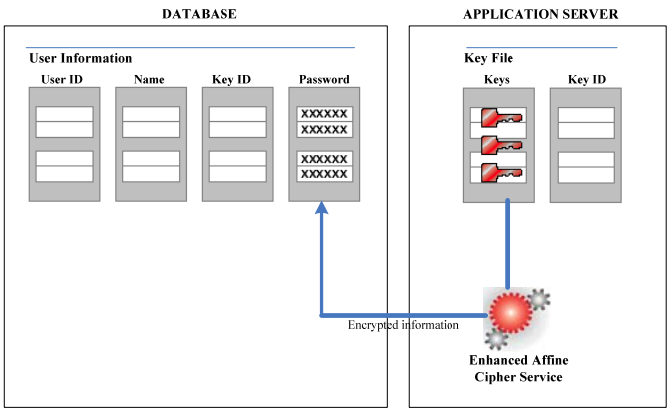

Fig. 1 applying database encryption outside the database

IV. Enhanced Affine Block Cipher

The analysis on affine block cipher was done and revealed that some new features can be added into its cipher such as the encoding schema and mode operation of block cipher. Therefore the new affine block cipher was designed and called enhanced affine block cipher to overcome the weaknesses of the original affine cipher. For implementation of these algorithms, the activity diagram was used to model the workflow behind the implemented system. The activity diagram is useful in understanding work flow analysis of synchronous behaviours across the process.

Figure 2 shows the process flow of encryption and decryption using Enhanced Affine Block Cipher. As seen in figure 2, the process started with either plaintext or ciphertext format as an input.

When plaintext is taken as an input, the Encoding activity is performed and followed by the Encryption activity and next the DecodingHex activity. The DecodingHex activity

indicates that both Display Result activity and the Store Result in Database activity occur at the same time.

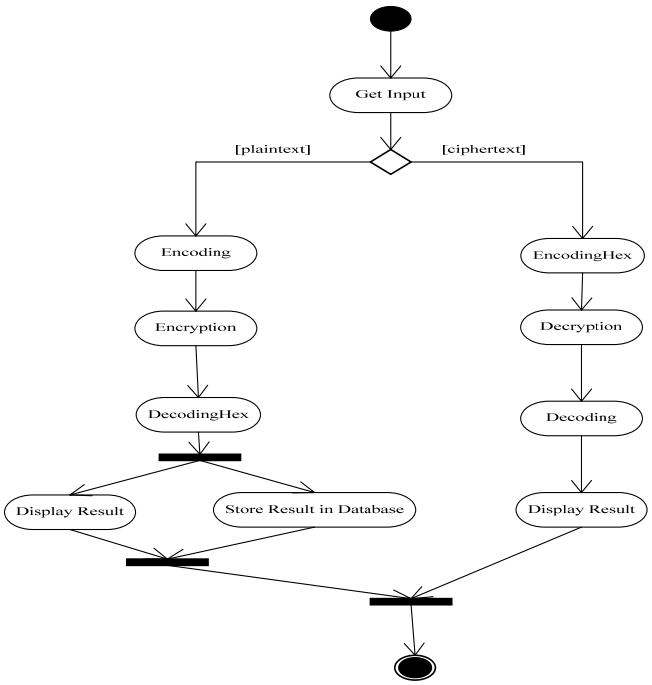

Fig. 2 activity diagram for database encryption using enhanced affine block cipher

Meanwhile if ciphertext is the input, the EncodingHex activity is performed and would then indicate the Decryption activity and next the Decoding activity. The Decoding activity indicates the Display Result activity. Finally the parallel activities are combined to end the activity.

The inputs that have been used in the encryption process are plaintext, key, block length and initial vector. In the decryption process, the ciphertext, key, block length and initial vector are its input. The plaintext was divided into simple and long plaintext. The main purposes of the testing are to validate the functionality of the algorithm and also to ensure that the database encryption is working properly. From the result, it was found that the algorithm is working properly where the decryption process produced a similar output to the original plaintext.

A. The Design of Enhanced Affine Block Technique

The design of enhanced affine block technique would be described in the next sections.

i) Encoding Schema

The encoding schema designed and developed was based on ASCII format. The plaintext and ciphertext is code and decode into certain number or value before encryption or decryption process. Hence, the encoding schema was used to enhanced affine block cipher.

The total of the ASCII characters set is 128. Therefore, the encoding schema is used based on these numbers where it contains encode and decode schema. In this encoding schema, during encryption process, the number will be converted into hexadecimal code whereas during decryption process, the number will be converted into characters.

Before plaintext and ciphertext is encrypted or decrypted, it was broken up into message units (block size). A message unit might be a single letter, a pair of letters (digraphs), a triple of letters or any number of letters. The encoding schema of message unit is done by an enciphering transformation function where it takes any plaintext message unit and transformed into a ciphertext message unit. In other words, it is a map from the set of P all possible plaintext message units to a set of C all possible ciphertext message units. The encoding schema of message unit is also done by deciphering transformation function where it takes any ciphertext message unit and transformed into an original plaintext message unit. In other words, it is also a map from the set of C all possible ciphertext message units to a set of P all possible plaintext message units.

ii) Encode and Decode Schema of Plaintext Message Unit

First, let start with encode schema and the case of a message unit (block size of plaintext message) is single letter in ASCII character (128 characters) was labeled by integer 0, 1, 2, 3... , 128-1.

For block size = 1, the message unit of plaintext is $p = x_1$. The formula of encoding schema is as follows:

$$p = x_1$$
$$= \sum_{i=1}^{1} 128^{1-i} x_i \quad \text{so for every p of plaintext}$$
$$p \in \{0,1,2,3,.. .128^{1} - 1\} = Z_{128}$$

With the same techniques, it can be applied for block size equal to two.
For block size =2, the message unit of plaintext is $p = x_1x_2$

$$p = 128 x_1 + x_2$$
$$= \sum_{i=1}^{2} 128^{2-i} x_i \quad \text{so for every p of plaintext}$$
$$p \in \{0,1,2,3,.. .128^{2} - 1\} = Z_{16384} = Z_{128^2}$$

Therefore, with the same techniques it could be used for block size = n, the message unit of plaintext is $p = x_1x_2\ldots x_n$

$$p = 128^{n-1} x_1 + 128^{n-2} x_2 + 128^{n-3} x_3 + \ldots + x_n$$
$$= \sum_{i=1}^{n} 128^{n-i} x_i \quad \text{so for every p of plaintext}$$
$$p \in \{0,1,2,3,...128^{n} - 1\} = Z_{128^n}$$

In decode schema of plaintext, the value or number was obtained from encrypting process is converted into

appropriate code. The process of converting a number (decimal numbers) into digits y_n, y_{n-1}...y_1 and y_0 such that

$$y = 128^{n-1} y_1 + 128^{n-2} y_2 + ... + y_n$$

It can be obtained by successively dividing y by 128 until quotient is 0. So the values are the remainders y_n, y_{n-1}…y_1, y_0. In case of encryption, the combination of these values is in hexadecimal number and is called ciphertext message.

iii) Encode and Decode Schema of Ciphertext Message Unit

First, let start with encode schema and the case of a message unit (block size of ciphertext message) is single letter in ASCII character (128 characters) was labeled by integer 0, 1, 2, 3... , 128-1.

For block size = 1, the message unit of ciphertext is $c = y_1$
The formula of encoding schema is as follows:

$$c = y_1$$
$$= \sum_{i=1}^{1} 128^{1-i} y_i \quad \text{so for every c of ciphertext}$$
$$c \in \{0,1,2,3,.. .128^1 - 1\} = Z_{128}$$

With the same techniques, it was also apply for block size equal to two.
For block size =2, the message unit of ciphertext is $c = y_1y_2$

$$c = 128y_1 + y_2$$
$$= \sum_{i=1}^{2} 128^{2-i} y_i \quad \text{so for every c of ciphertext}$$
$$c \in \{0,1,2,3,...128^2 - 1\} = Z_{16384} = Z_{128^2}$$

Therefore, with the same techniques, it was also concluded that as follows;
For block size = n, the message unit of ciphertext $c=y_1y_2…y_n$

$$c = 128^{n-1} y_1 + 128^{n-2} y_2 + 128^{n-3} y_3 + ... + y_n$$
$$= \sum_{i=1}^{n} 128^{n-i} y_i \quad \text{so for every c of ciphertext}$$
$$c \in \{0,1,2,3,.. .128^n - 1\} = Z_{128^n}$$

In decode schema of ciphertext, the value or number was obtained from decrypting process is converted into appropriate code. The process of converting a number (decimal numbers) into digits x_n, x_{n-1}...x_1 and x_0 such that

$$x = 128^{n-1} x_1 + 128^{n-2} x_2 + ... + x_n$$

It can be obtained by successively dividing x by 128 until quotient is 0. So the values are the remainders x_n, x_{n-1}…x_1, x_0. In case of decryption, the combination of these values is in decimal number and is called plaintext message.

B. Design Enhanced Affine Block Cipher

The Affine cipher works by transforming the letters of the alphabet to their corresponding numerical value (which is from 0 to 25), then utilize the encryption formula as follows;

$$e_{a,b}(x) = (ax + b) \bmod 26$$

This encryption function must be bijective, and a must have a multiplicative inverse mod 26 (gcd(a,26) is equal 1). For decryption function

$$d_{a,b}(y) = a^{-1}(y - 1) \bmod 26$$

The invertible integers mod 26 are set of {1,3,5,7,9,11,15,17,19,21,23,25}.

New affine cipher namely enhanced affine block cipher was designed based on the encoding schema as mentioned earlier. For the first step, recall the affine cipher as follows:

Let $P = C = Z_{128^n}$ and n is block size.

$K = \{(a,b) \in Z_{128^n} \times Z_{128^n} : \gcd(a,128^n) = 1\}$

for $K = (a,b) \in \kappa$,

the encryption functon is defined as

$$e_\kappa(x) = ax + b \bmod 128^n$$

and the decryption function is defined as

$$d_\kappa(y) = a^{-1}(y - 1) \bmod 128^n$$

where $(x,y \in 128^n)$

The second step for enhancement of affine cipher is done by adding modes of operation into the block cipher algorithm. . This technique is similar to cipher block chaining (CBC) mode. Figure 3 and figure 4 show that the processes of encryption and decryption of enhanced affine block cipher with its modes of operation.

It was discovered that, by applying CBC mode; during the operation XOR, certain values are more than the value of the modulo 128^n. Due to the basic properties that is congruence between the value and the modulo; it cannot give exact value during the decryption process. Based on analysis and initial testing, instead of using XOR as mode of operation, this algorithm was used as an additional operation for encryption and subtraction operation for decryption. The mathematical formula for this mode of operation is as follows:

$$C_i = e_\kappa(P_i + C_{i-1}),\ C_0 = IV \text{ for encryption}$$

and

$$P_i = d_\kappa(C_i) - C_{i-1},\ C_0 = IV \text{ for decryption}$$

In encryption process, each plaintext block is added with previous ciphertext block, and then encrypted. An initialization vector (IV) is used as a seed for the process. In decryption process, each decrypted ciphertext block is subtracted with the previous ciphertext.

This proposed enhanced affine block cipher could be used for any application systems which needs the sensitive data to be protected.

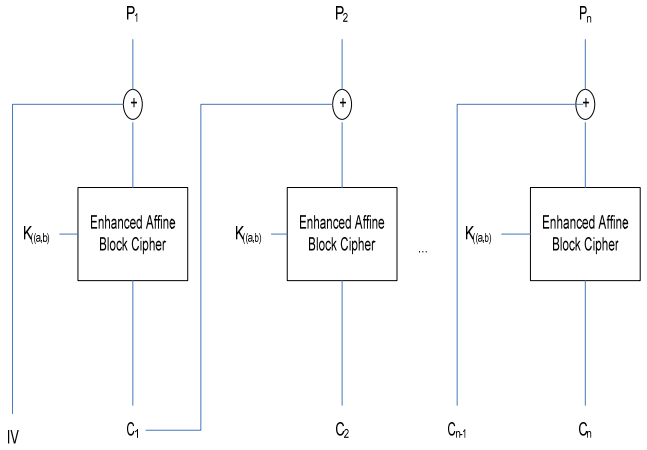

Fig. 3 modes of operation during encryption process

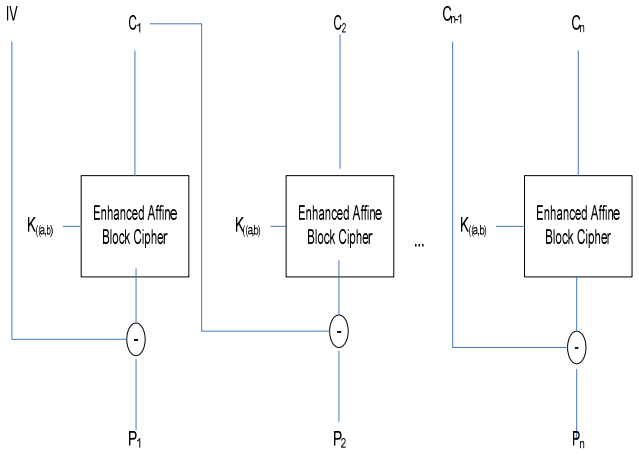

Fig. 4 modes of operation during decryption process

VI. The Implementation

In this phase, the development of the system was implemented using java and java server pages code. The system was developed in Netbeans IDE version 4.1 and the database used is PostgreSQL8.1. The Aqua Data Studio 4.5 was used to create table and to query the data in database PostgreSQL8.1. Apache Tomcat5.5 was used as web server and java server pages as its web component language. The enhanced affine block cipher was coded in java beans class where java.math.BigInteger, a class that represents arbitrary precision integers was used to handle the large integer and modular arithmetic involves in this algorithm. Java servlet was used to interact with user input pages and the algorithm in java beans and database. An additional driver such as postgresql-8.0-314.jdbc3.jar is also needed in order to ensure smooth and successful connection between the database and the application of the system.

The Enhanced affine block cipher was implemented in the existing Fraud Management System (FMS). Currently, users can logon to the system by entering user ID and their password. The password stored in database is in cleartext (not encrypted). The password can be obtained by other users if they have privileged to access the database and execute the query. The password also can be intercept by someone else in the network using certain tools.

Performing encryption for securing password within FMS will add more security and prevent unauthorized users to logon to the system. The implementation was done by performing encrypted password in table user_account. The architecture of the encrypted password for FMS is showed in Figure 5. The modification was made based on the changes of password data in table user_account. The FMS modules were added by the enhanced affine block cipher are login module, insert new password module and change password module. The modification was also made to handle the reading and store the keys stored in file storage.

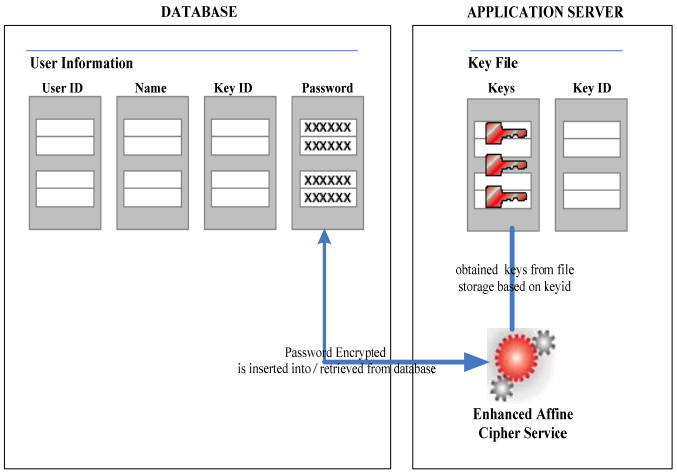

Fig. 5 applying encrypted password for fms

The implementation of encrypted password for FMS runs through the specified configuration and evaluated thoroughly with respect to database approach and cryptographic algorithm technique to prove the design.

A. Enhanced Affine Block Cipher Class Diagram

The enhanced affine block cipher was converted into class diagram as shown in Figure 6. It shows set of classes, attributes, operations and their relationships between them. Class diagram is used to model the static design view of the system. There are three classes for the enhanced affine block cipher; EncodingSchema, EnhancedAffineCipher and UserInfo.

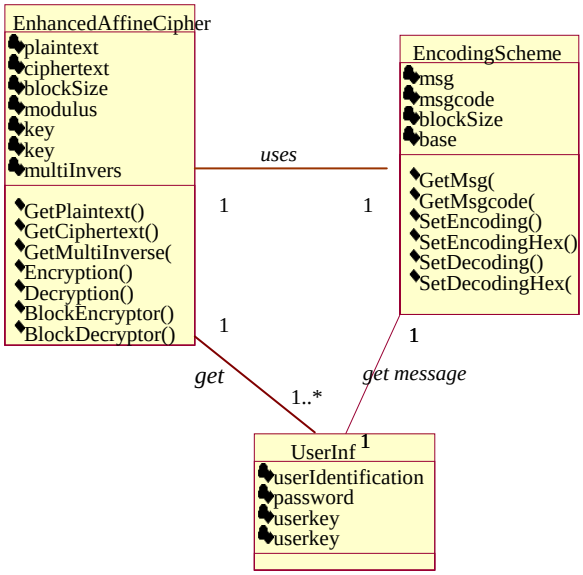

Fig. 6 class diagram enhanced affine block cipher

B. Input and Output of Encryption and Decryption Process

This section discusses the input and output of the encryption and decryption process by using enhanced affine block cipher algorithm. Several results are shown based on the different of the inputs.

i) Input and Output of Encryption and Decryption Process

There are two samples of same plaintext with different block size and keys. The two type of block size used are 4 and 64 characters with difference key A and key B and also initial vector.

Table I presents the input of the process using this algorithm for a cipher input block size of 4 characters (32 bits length) and
key A = 777771; key B = 123456789; initial vector (IV) = 654321.

Table I The Result of Encryption and Decryption Process of Simple Plaintext with Block Size=4

Cipher key	A =777771 , B=123456789, IV=654321
Block size	4 Characters (32 bits length)
Input	Plaintext : Enhanced affine block cipher
Output	Ciphertext: 17760d5b212b442a603818457101670c344 3353a311d0c76686d430d3a
Input	Ciphertext : 17760d5b212b442a603818457101670c344 3353a311d0c76686d430d3a
Output	Original message: Enhanced affine block cipher

Meanwhile, in Table II presents the input of the process using this algorithm for a cipher input block size of 64 characters (512 bits length) and
key A = 7211111111111111111; key B = 123456789999999999999999999999999999999999999999999999999; initial vector = 654321.

Table II The Result of Encryption and Decryption Process of Simple Plaintext with Block Size=64

Cipher key	A =7211111111111111111 , B=123456789999999999999999999999999999999999999999, IV=123456
Block Size	64 Characters (512 bits length)
Input	Plaintext : Enhanced affine block cipher
Output	082d38043150762c6d762b1b5e1b233b5e2a0 8290a007d06684140783a3a3a3a3a3a3a3a3a 3a3d04365e433405311b7d496d004e260344 2c5549245f441f443
Input	ciphertext : 082d38043150762c6d762b1b5e1b233b5e2a0 8290a007d06684140783a3a3a3a3a3a3a3a3a 3a3d04365e433405311b7d496d004e260344 2c5549245f441f443
Output	Original message: Enhanced affine block cipher

The following tables (Table III and IV) are 2 samples of half paragraph plaintext with different block size and keys. The two types of block size used are 4 and 64 characters with difference key A and key B and also initial vector.

Table III and IV present the input of the process using the algorithm for a cipher input block size of 4 characters (32 bits length) and
key A = 7; key B = 19; initial vector (IV) = 123456.

Table III Result of Encryption Process of Half Paragraph of Plaintext with Block Size=4 and Key A=7, B=19

Cipher key	A =7 , B=19, IV=123456
Block Size	4 Characters (32 bits length)
Input	Plaintext : PostgreSQL has a rich set of native data types available to users. Users may add new types to PostgreSQL using the CREATE TYPE command. Table 8-1 shows all the built-in general-purpose data types. Most of the alternative names listed in the Aliases column are the names used internally by PostgreSQL for historical reasons. In addition, some internally used or deprecated types are available, but they are not listed here.
Output	Ciphertext: 36440a7f550014510e167622104721285a16 4d6051044a7665061b1729326b600723371 37425333f14383d5c5832153e2a0b7b39530

	759652b677f366c02452b1b554a136c202b3 d5b625e0d675f542a3c25177a0c367c79202 7510e6c796d7e2606592351782c1c2403603 6036d0939784025770d2855204201256201 6d6d761c0201744b4c501a5b695d777c2874 5c2377173c59516b5c58696c0b5416522332 5278675c730e430e0b3c2b6a1949675d741b 1735434d593b3e612951600f5e1136374a3d 6126225b56720a4d46503163535f5e6b1342 271b3a355e047e7f4c234224077235277069 327d02654b1c790b721756374c50302b7d6 d6c2e2a02092953321509793c692b59566d7 b1a464109022b3a2418616c2134597b64415 b1e1030346e1e04493319693d0c1a323b051 e3f025b0748366f6f27757c2e5f671c56267a 1940155b226d594060315754351041 1b2b0 249042077074f36653c59014079554d711c1 50e276479315f58752d581b6711272904592 86847624a7f3b213a3b4b59124d457647544 e7b2e7e0f1c263c3b373a4a6474221a701f69 213d18292812136a3c445a

Table IV The Result of Decryption Process of Half Paragraph of Plaintext with Block Size=4 and Key A= 7, B=19

Cipher key	A =7 , B=19, IV=123456
Block Size	4 Characters (32 bits length)
Input	Ciphertext : 36440a7f550014510e167622104721285a16 4d6051044a7665061b1729326b600723371 37425333f14383d5c5832153e2a0b7b39530 759652b677f366c02452b1b554a136c202b3 d5b625e0d675f542a3c25177a0c367c79202 7510e6c796d7e2606592351782c1c2403603 6036d0939784025770d2855204201256201 6d6d761c0201744b4c501a5b695d777c2874 5c2377173c59516b5c58696c0b5416522332 5278675c730e430e0b3c2b6a1949675d741b 1735434d593b3e612951600f5e1136374a3d 6126225b56720a4d46503163535f5e6b1342 271b3a355e047e7f4c234224077235277069 327d02654b1c790b721756374c50302b7d6 d6c2e2a02092953321509793c692b59566d7 b1a464109022b3a2418616c2134597b64415 b1e1030346e1e04493319693d0c1a323b051 e3f025b0748366f6f27757c2e5f671c56267a 1940155b226d594060315754351041 1b2b0 249042077074f36653c59014079554d711c1 50e276479315f58752d581b6711272904592 86847624a7f3b213a3b4b59124d457647544 e7b2e7e0f1c263c3b373a4a6474221a701f69 213d18292812136a3c445a
Output	Original message: PostgreSQL has a rich set of native data types available to users. Users may add new types to PostgreSQL using the CREATE TYPE command. Table 8-1 shows all the built-in general-purpose data types. Most of the alternative names listed in the Aliases column are the names used internally by PostgreSQL for historical reasons. In addition, some internally used or deprecated types are available, but they are not listed here.

Table V and table VI present the input of the process using this algorithm for a cipher input block size of 64 characters (512 bits length) and key A = 5, key B = 177771 and initial vector (IV) = 123456.

Table V The Result of Encryption Process of Half Paragraph of Plaintext with Block Size=64 and Key A= 5, B=177771

Cipher key	A =5 , B=177771, IV=123456
Block Size	64 Characters (512 bits length)
Input	Plaintext : PostgreSQL has a rich set of native data types available to users. Users may add new types to PostgreSQL using the CREATE TYPE command. Table 8-1 shows all the built-in general-purpose data types. Most of the alternative names listed in the Aliases column are the names used internally by PostgreSQL for historical reasons. In addition, some internally used or deprecated types are available, but they are not listed here.
Output	004e1a2b36050b350e286f46440d6e377105 59683c71181020704c167038442b5c51036e 7a442a37712e0b5f100d6e4450425b124156 1303712c41714127253d1b3c61191a705f7f 002e0c3b7c492d1654253b1a761456140069 185f1a1e130c7a2e7f412347344e5b2b776a6 b31254340772d65000d49100d3a2d13530e 366757295c2237510b6551362e495175425 d1f6f56180d574b097e116a196368317a035 43533386f4e327632187d31384d455c43473 f08623b4a03435439554a02480677691e0f7 02759252965432b2a1f326138285f2f10187 6207e3e3a6c2f6d516b5404414c246b72572 a4a295a246c73616d31420013040e50742e1 46f3d5809722854401f01392435524c25473 04604533b77035a6f0456791b6572210d116 a7d28562c43673808065e1241225e6312326 609002f37536f6865757d4b214d3b493f553 338166f223d1c252577760008776b24600b6 35c635a4b3b0a7b32160463045a0e203b1a0 94d774e532369606b2c5756147612675826 204f405a034348390a192c3d741822106233 28266272552c073a10236b6656736803403c 153d05674a1f557869752241233b130d3e14 5c35664972294a16723b504b321327683f

Table VI The Result of Decryption Process of Half Paragraph of Plaintext with Block Size=64 and Key A= 5, B=177771

Cipher key	A =5 , B=177771, IV=123456
Block Size	64 Characters (512 bits length)
Input	ciphertext : 004e1a2b36050b350e286f46440d6e377105 59683c71181020704c167038442b5c51036e 7a442a37712e0b5f100d6e4450425b124156 1303712c41714127253d1b3c61191a705f7f 002e0c3b7c492d1654253b1a761456140069 185f1a1e130c7a2e7f412347344e5b2b776a6 b31254340772d65000d49100d3a2d13530e 366757295c2237510b6551362e495175425 d1f6f56180d574b097e116a196368317a035 43533386f4e327632187d31384d455c43473 f08623b4a03435439554a02480677691e0f7 02759252965432b2a1f326138285f2f10187 6207e3e3a6c2f6d516b5404414c246b72572 a4a295a246c73616d31420013040e50742e1 46f3d5809722854401f01392435524c25473 04604533b77035a6f0456791b6572210d116 a7d28562c43673808065e1241225e6312326 609002f37536f6865757d4b214d3b493f553 338166f223d1c252577760008776b24600b6 35c635a4b3b0a7b32160463045a0e203b1a0 94d774e532369606b2c5756147612675826 204f405a034348390a192c3d741822106233 28266272552c073a10236b6656736803403c 153d05674a1f557869752241233b130d3e14 5c35664972294a16723b504b321327683f
Output	Original message: PostgreSQL has a rich set of native data types available to users. Users may add new types to PostgreSQL using the CREATE TYPE command. Table 8-1 shows all the built-in general-purpose data types. Most of the alternative names listed in the Aliases column are the names used internally by PostgreSQL for historical reasons. In addition, some internally used or deprecated types are available, but they are not listed here.

VII. Conclusion

This paper focused on the design of database encryption at application level using enhanced affine block cipher. This improvement has been made because of the weakness found in the original affine cipher. In this paper, the improvement is made by using a new encoding schema and mode of the operation for the encryption and decryption process. The enhanced affine block cipher is developed and implemented where the selected sensitive data is encrypted outside the database (application level) and then it is inserted into database.

There are two types of samples used namely the plaintext and ciphertext for validation purpose. Two types of block size have been used, i.e. 4 and 64 characters. Each of block size has been testified by using different keys and two types of samples; plaintext and ciphertext. This algorithm is then applied in existing system involves in database.

The enhanced affine block cipher can be used to explore other existing symmetric cryptography algorithms or combine it to other techniques. The mode of operations used in enhanced affine block cipher also can be extended into others approaches. The database encryption can also be applied in hybrid cryptography techniques. This technique can be applied by combining the symmetric key cryptography and asymmetric key cryptography.

References

[1]Chen, G., Chen, K., Dong, J., "A Database Encryption Scheme for Enhanced Security and Easy Sharing,"*Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design*, 2006

[2]He, J. and Wang, M., "Cryptography and Relational Database Management Systems," IEEE, 2001.

[3]Koblitz,N., *A Course in Number Theory and Cryptography*. New York: Springer-Verlag, 1988.

[4]Mattsson, U.T. "A Practical Implementation of Transparent Encryption and Separation of Duties in Enterprise Databases: Protection against External and Internal Attacks on Databases," *IEEE International Conference*, 2005.

[5]Maurer, U.,"The Role of Cryptography in Database Security," ACM SIGMOD, 2004.

[6]Menezes, van Oorschot. P.C., and Vanstone, S.A., *Handbook of Applied Cryptography*. CRC Press, 1999.

[7]Piattini, M.G., Ferntindez-Medina, E., "Secure databases: State of The Art," *IEEE*, 2000.

[8] RSA Security, Inc., "Securing Data at Rest; Developing a Database Encryption Strategy," White Paper, 2002.

[9]Russell, and Gangemi,G.T, *Computer Security Basics*. O'Reilly, 1991.

[10]Sesay, S., Yang, Z., Chen, J., and Du Xu, " A Secure Database Encryption Schema," *IEEE*, 2004.

[11]Stinson, D.R., *Cryptography;Theory and Practice*.CRC Press, 1995.

[12]Tropical Software, 2007. http://www.tropsoft.com/strongenc/des3.htm