An Efficient Encryption Algorithm for Sensitive Data Using Numeric and Alphanumeric Format

Article *in* Journal of Computational and Theoretical Nanoscience · September 2018

DOI: 10.1166/jctn.2018.7561

2 authors, including:

Shikha Gupta
Netaji Subhas Institute of Technology
8 PUBLICATIONS **20** CITATIONS

Journal of
Computational and Theoretical Nanoscience
Vol. 15, 1–11, 2018

An Efficient Encryption Algorithm for Sensitive Data Using Numeric and Alphanumeric Format

Shikha Gupta[1,*] and Satbir Jain[2]

[1] *Netaji Subhash Institute of Technology, Dwarka, New Delhi*
[2] *Department of Computer Engineering, India*

In the contemporary world, with the growth of networking and increase in the volume of data storage capacity, a significant amount of personal information leakage accident occurs that leads to loss of personal data day by day. Hence, there arises a need to emphasize more on the security of data stored either in databases or transmitted over the web to ensure that user's personal data is stored at a safer place. To maintain the privacy and security of data a protective layer of encryption is applied around the sensitive data items focusing on encrypting only the sensitive data. Standard encryption techniques such as AES, DES or 3DES are used to encrypt the data, but these techniques are unsuitable for encrypting personal information as they lead to various shortcomings. In this paper to encrypt personal information a new and efficient encryption scheme is proposed that encrypt numeric data having a length ranging from 1 to 19 such Aadhaar card number, mobile numbers, SSN, IP address, credit card numbers, CVV, and alphanumeric data like email ids. We integrate the proposed solution with various encryption schemes such as Advanced Encryption Standard (AES) on the remote side database. Data transmission and data storage take place at the server database. The remote side and server side database will be secured with the same data formats. Hence, this paper describes an efficient encryption scheme for securing numeric and alphanumeric data of databases.

Keywords: Database, Format Preserving Encryption (FPE), Data Security, Encryption, Advanced Encryption Standard (AES), Numeric and Alphanumeric Data.

RESEARCH ARTICLE

1. INTRODUCTION

In today's data-centric world security is becoming an essential factor for most of the organizations as a significant amount of data stored in their database systems. The database is a storage area used to ensure that the personal information of the user safe. The data stored may be either sensitive or non-sensitive. Hence, the security of data becomes the primary concern for most of the organizations as it contains particular categories of data which includes credit card number, social security number, passport number, the account number that should not be revealed. Damage and misuse of sensitive data do not only affect a single user but possibly an entire organization.[1] It implies that security and privacy of data are most important for protecting the data from unauthorized users or attackers. Lots of research is being done in the field of data security to protect the personal data to maintain the privacy, integrity, and availability of data at the database level. Cryptography plays a vital role in securing data in a variety of contexts. For example, encryption algorithms are used to encrypt sensitive data such as credit card numbers, account numbers, and other personal information. Encryption has proved to be the best solution to achieve a high level of security, especially for sensitive data. Symmetric key algorithm and asymmetric key algorithm are two categories of encryption shown in Figure 1. In the symmetric key algorithm, a private key is used both for encryption and decryption and the key is distributed via key exchange protocol or offline. In asymmetric key algorithms, two keys are used one for the encryption and other for the decryption of data. A key is kept private, and the other is divided publicly among the users. Asymmetric key algorithms solve the problem of key distribution, but it is complicated and consumes more time as compared to symmetric key algorithms.

Symmetric key algorithms work on the basis of two modes either as stream ciphers or as block ciphers. Stream ciphers, typically divide and encrypt a single character, bit by bit from a long sequence of characters and then apply bitwise modulo 2 to the plaintext to produce the ciphertext.

*Author to whom correspondence should be addressed.

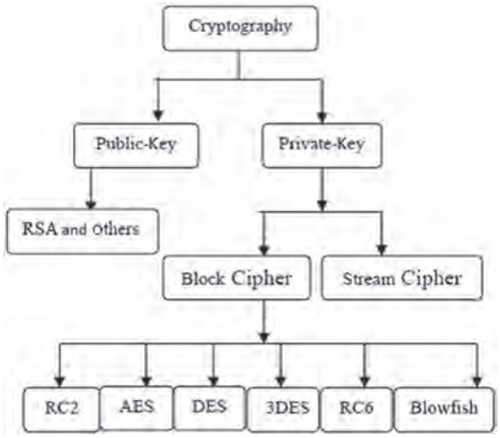

Fig. 1. Modern block ciphers.[3]

Stream ciphers have faster processing speed, but weaker in the context of integrity protection and authentication.[3] On the other hand, block ciphers play an important in securing database systems.[4, 6] It works by dividing the plaintext into blocks of a fixed length cipher, which are then encrypted into blocks of ciphertexts using the same key.

Block ciphers are slow in processing but used as they provide high efficiency and low implementation complexity. Modern block ciphers are widely used in cryptography because they help in maintaining the data confidentiality, message authentication codes (MACs), hash functions, pseudorandom number generators (PRNG) and authentication protocols.[4, 7] AES, DES, 3DES, Blowfish are most successful symmetric key block ciphers. These ciphers are applied to 64 bit or 128-bit blocks of data. However, at the same time, encrypting data using modern block ciphers is also the biggest challenge because these ciphers will encrypt data in bytes or bits and results in binary strings which are no more in its original form as it increases the data length and tampers the format of encrypted data obtained as shown in Figure 2.

In Figure 2 it is shown that encrypting a numeric string of size 16 bytes by using standard encryption algorithms such as AES may produce a series of ciphertext that contains alphanumeric characters. It implies that the encryption process alters the format and hence, it loses its data format. The modified data format of encrypted string disrupts the software applications as well as it forces to make changes in database schema where the encrypted data to be stored. The above encryption schemes cannot meet the availability demand for sensitive data security. The solution to the above problems is defined by using an effective

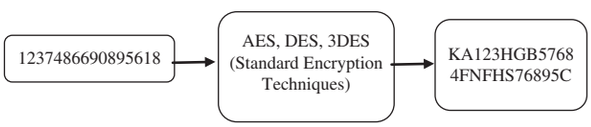

Fig. 2. Encrypting numeric data using standard encryption technique.

cryptographic solution that encrypts and decrypt the data without altering its original format. This paper finds a solution that transforms the unformatted data of numeric and alphanumeric format into a sequence such that the encrypted data has the same size and format as the original plaintext. It works by encrypting a plain text of n bit string into a ciphertext of n bit string. It uses feistel network for encrypting sensitive data.

Formulation. The paper formulation is done by dividing it into the following four sections, which starts from Section 2: Section 2 focuses on existing encryption techniques for ensuring data security. Section 3 introduces the methodology used in the proposed algorithm and explains its procedure, while Section 4 presents the proposed solution and its implementation along with its suggested structure. Section 5 defines the comparative study of various data formats and calculate the performance of each format, and discusses them. Section 6 provides the conclusion with some suggestions for future research work.

2. RELATED WORK

Encryption is a cryptographic technique that plays a vital role in securing sensitive data. Encrypting entire files of sensitive data can be a useful technique for securing a significant amount of data. However, sometimes encrypting bulk of records can be inefficient as it is not possible to decrypt a selected portion of data that is needed by an application. If an application wants to access a small part of the data, the entire file needs to be decrypted. Hence, it would be difficult to design any database system by keeping this idea in mind. To avoid such problem researchers have proposed and implemented various cryptographic techniques. Hence, encrypting sensitive data, that is transmitted over a communications channel or stored in a database is secured, even if it moves over an unsecured communications channel. It prevents data from illegal stealing and avoiding the risks such as misuse of sensitive data[1] to achieve the better security that assumed hard to break by an intruder. In Ref. [8] Yong-Xia describes classical encryption schemes such as substitution and transposition along with traditional encryption techniques such as AES, DES and how these techniques could help in securing the database systems. In Ref. [9] Bouganim and Pucheral proposed a chip-based smart card solution to protect data confidentiality, and a security model is proposed to avoid tampering of data such that only the owner of the database can access the data using a client terminal that is supported by smart card devices.[4] This solution proposed is considered as a secure and a practical solution, but it was complicated and expensive.[1] To execute a significant query data must be decrypted because data encryption generally affects the performance of the database. Yand and Sesay[10] suggested that encrypting sensitive data can only provide the desired security

RESEARCH ARTICLE

to the database without affecting its performance. Numerous encryption algorithms were proposed depending on the sensitivity of data. In Ref. [11] Kaur proposed an encryption technique to encrypt numeric data of database using 3 kdec algorithm having a fixed data field type and length. Though the algorithm is easy to use and computationally very fast, still it lacks behind as it does not support the encryption of character data. In Ref. [12] Agrawal also proposed an encryption technique to encrypt numeric data called as Order Preserving Encryption (OPE). It allows a large number of SQL queries or any comparison operations to be processed directly over the encrypted data without decrypting them. The OPE technique allows to create indexes of the database over encrypted data and can easily be integrated with existing databases. The method supports range queries as it can be applied only to numeric data and also reveals the order of existence of data. A technique named Format Preserving Encryption[13, 14] is gaining attention to handle the problem arising due to Order preserving encryption. Since 1981, researchers have done plenty of research on Format Preserving Encryption Technique. FPE uses pseudo-random numbers and permutations.

In Ref. [15] BPS proposed an enhanced FPE scheme based on FFX that describes the usage of tweak function.[16] It encrypts the string of any length by using CBC mode[18, 19] having AES,[17] TDES,[20] or SHA-2[22] as inner block ciphers. In Ref. [23] Vidhya and Chitra proposed an FPE algorithm based on AES for encrypting numeric data of 16 digits only. In Ref. [25] Mallaiah proposed an encryption scheme that encrypts the numeric data only. It also discussed the overhead of the proposed technique over a (FIPS-74-8) standard. It is a NIST standard based on DES algorithm and that analyses the performance of FPE over the modern block ciphers such as AES, Blowfish, 3DES, DES with different key sizes. The FIPS 74-8 standard technique was applied over the modern block ciphers to map digits using CFB mode[18, 19] of operation which improves the security of data. In Ref. [27] Naor and Reingold proposed a technique that performs various operations based on Galois Field GF (2^N). The usage of pairwise independent permutation function to be used in LR constructions proves to provide additional security as well as randomness to data. The problem with this technique was that the GF (2^N) representation for a variable length of inputs is complicated in practice. In Ref. [28] Luby and Rackoff proposed and analyzed the construction of a secure block cipher by using it with Feistel function[28] and tried to prove that whether the round function is a secure pseudorandom function regarding cryptography or not.

The proposed technique uses three round function to make block cipher a useful and safe pseudorandom function. Black and Rogaway[29] proposed three procedures for arbitrary domains. The prefix cipher, Cycle walking method and Feistel + Cycle method. First two techniques mentioned in their work proved to be very expensive in practice, and Feistel and cycle method use DES[20] as a pseudorandom function. In Ref. [30] Mihir and Bellare proposed a variable input length block cipher.[32] The problem with their technique is that it requires constructions of multiple application of original block ciphers to design an arbitrary length block cipher and which makes it inefficient and computationally hard.

3. METHODOLOGY

This section provides a logical description of the proposed approach to countermeasure data security issues in designing of a database system.

3.1. Encryption Solution for Achieving Data Security

The underlying problem that occurs in applying data encryption to a database system is that whether it should be implemented inside the database or outside the database. A large number of encryption algorithms are available for encrypting the data at the database level. The most successful approach adopted to secure data is symmetric key encryption techniques.

DES[20] is one of the most widely used iterated block cipher encryption that uses symmetric key encryption to operate on a plaintext block of 64-bit to results into a ciphertext block of 64-bit, using a key of 56-bit and 16 rounds.[8] DES becomes the basis for secure communication but now is considered to be insecure for many applications, due to the large amount of data availability and the size of the key, which is too small.[35] Therefore, DES is replaced by a new block cipher named as Advanced Encryption Standard (AES). It is a replacement for the DES algorithm as a standard for data encryption. AES[17] is the most frequently used encryption algorithm due to its simplicity and efficiency. It also works on symmetric-key encryption algorithm that operates on 128-bit plain text and uses keys of various lengths 128, 192, or 256-bit, the key length specifies the number of rounds in the algorithm.

AES and DES are most commonly used encryption algorithms for encrypting data files of fixed length having a specified format. The problem that occurs in adopting the standard encryption algorithms is that they cannot fulfill the requirements of encrypting the sensitive data because these algorithms change the format of the ciphertext obtained and hence cause changes to the database schema to the stored encrypted data back to it. A solution is proposed in this paper to overcome these problems by combining a symmetric encryption algorithm with Feistel cycle walking[26] method to encrypt numeric and alphanumeric data to maintain the original format of data.

3.2. AES (Advanced Encryption Standard)

AES[17] is a modern block cipher encryption technique, which takes 128 bits data as input and encrypts it with a key of size 128, 192 and 256 bits. In AES the regular byte structure is maintained throughout the cipher that is the

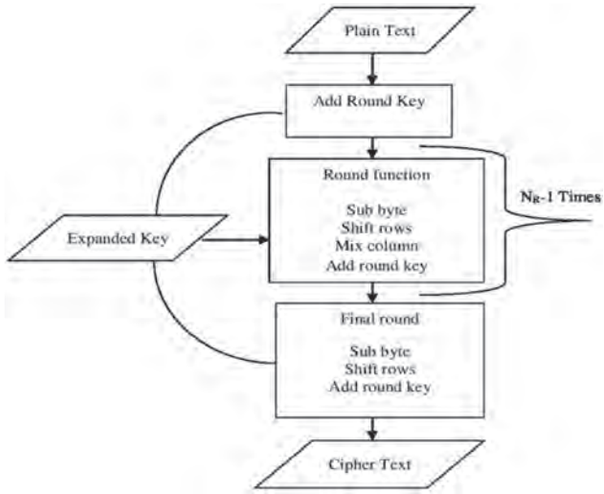

Fig. 3. AES encryption flowchart.

most striking feature of it. The encryption process is the same in AES. Therefore, iterative operations are performed many times on fixed size input bytes. The second most striking feature of the AES is that to reuse it whenever a component is required, and whenever different options are available, the simplest and most efficient option is chosen.[6] The final result obtained is a cipher that is very attractive to implementers.

In AES the key expansion process takes an input of 16-words and gives an expanded key of 44-words (First 4 words denote master key, and rest are expanded keys for the next ten rounds).

The following keyword evaluates the expanded key for each round—

- *Word_Rotate*: In word rotate a word is shifted circularly, i.e., left by one byte per word. For [W0, W1, W2, W3] input word, the changed output obtained would result in [W1, W2, W3, W0].
- *Word_Sub*: In word sub, substitute a word with corresponding elements of Rijndael S_Box matrix.
- The results of the above operations are exclusive OR with constant Rcon (i/4). [Rijndael Rcon matrix].

All operations in a round of AES can be easily broken down to these essential functions:

1. *Add Round Key*: In this operation, every 16 bytes of data is XORed with 16 bytes of expanded keys of the current round. The expanded keys never reused.
2. *Sub Byte*: This operation replaces the input bytes with corresponding Rijndael S-Box matrix value.
3. *Shift Rows*: This operation shifts the rows of input, circularly left by ⟨current row number⟩ bytes and row number always start with zero.
4. *Mixed Column*: This operation performed on every column. Each value multiplied with every value of matrix (16 total multiplications). The result of these multiplications is XORed to produce 4-byte output for the next state.

3.3. FPE (Format Preserving Encryption)

FPE is a symmetric block cipher encryption technique, which takes n bits of input (plaintext) and encrypts it with a 16-word key (128 bits). The resultant ciphertext obtained is not only having the same length of bits as input but also having the same format as that of the plaintext. It is also useful for internet security because it can protect data in transmission process without changing the form of the datagram. Prefix cipher, Cycle walking and Generalized Feistel Network[27] are three commonly used encryption and decryption scheme used to encrypt data.

3.3.1. FPE Using Prefix Cipher

Prefix cipher is one of the easiest methods of implementing FPE in which system generates random weights to create a permutation table in memory and then perform the encryption and decryption based on the permutation table. Random weights are calculated using modern block cipher methods. The table contains the input data of 6 digits and the desired encrypted value; then the table will be sort according to the encrypted value. This scheme is suitable for the small size of plain text, i.e., within 6 digits only. Suppose, to implement FPE on domain $N = \{0, 1, 2, 3, 4, 5\}$ having five possible input values. Under the control of a key K, of size 128 bits, we compute $X(0) = \mathrm{AES}_K(0)$, $X(1) = \mathrm{AES}_K(1)$, $X(2) = \mathrm{AES}_K(2)$, $X(3) = \mathrm{AES}_K(3)$, $X(4) = \mathrm{AES}_K(4)$, $X(5) = \mathrm{AES}_K(5)$. Use the relative ordering of $X(0)$, $X(1)$, $X(2)$, $X(3)$, $X(4)$ and $X(5)$ to determine the desired permutation. For larger values, the size of the table is too huge to be practical.

3.3.2. FPE Using Cycle Walking

Cycle walking is applied repeatedly to modern block ciphers such as AES, 3DES, etc. until results fall in the desired domain. The method is suitable for integer set in the range of 54 to 64 bits and will not reduce the security of the block cipher,[29] but it always has a poor performance. However, if the size of the integer set to be encrypted is much smaller than the length of a block cipher, it will consume much time.

3.3.3. FPE Using Generalized Feistel Network

In this method encryption of input data is done in two stages.

a. Firstly, it will generate a block cipher E of length 2 m; 2 m is block cipher size and using it with any of encryption techniques such as AES, 3DES, etc. to encrypt the data.
b. Secondly, using the cycle walking technique to ensure that the desired output falls in the correct domain.

The generalized-feistel is most widely method that can be applied to encrypt the integer set of size 40 to 240 bits.

The security of it proved by Black and Rogaway in Ref. [29].

4. PROPOSED SOLUTION FOR ENCRYPTING NUMERIC AND ALPHANUMERIC DATA

In this section, the proposed encryption method is defined and implemented by using Advanced Encryption Standard (AES) and Generalized Feistel Network. The algorithm implemented can be used to encrypt both numeric and alphanumeric data. Figure 4 shows the proposed encryption scheme for encrypting numeric and alphanumeric data. Numeric data is encrypted by applying XOR operation to adjacent bytes of the output of the AES technique. In algorithm 1 input is given by input_data which obtained from AES encryption. If the length of the plaintext is more than 16 bytes, then the length of ciphertext will be multiple of 128 bits. Hence, we need to apply the XOR operation again on obtained XORed output. After repetitive XORing, output will be of 16 bytes. Now, a translation method applied on every byte of the ciphertext that will result in a 16-digit number or 16-byte number.

4.1. Mapping Procedure for Numeric Data Encryption

Numeric data having a length ranging from 1 byte to 15 bytes would need a reduction in length of the output. The reduction can be accomplished by looping the output for truncation of the required number of bytes. We keep on applying XOR operation on the last two bytes and pre-appending the result in the number. This procedure will keep on reducing the length of the output number by one. Hence, the output obtained will achieve the required length of encrypted data. Algorithm 1. shows the encryption process for 16-byte numeric data.

ALGORITHM 1 (NUMERIC DATA ENCRYPTION).
1: **Input:** AES 128 encrypted data (input_data)
2: **Output:** Encrypted data of the desired length (fpe_data)
3: **Start:**
4: if length(input_data) = 128 then
5: fpe_data ← input_data //we get 16-byte encrypted data
6: else
7: //xor adjacent bytes to get 16-byte encrypted output
8: fpe_data ← 16-byte data
9: //apply 5211 translation on the encrypted data
10: fpe_data ← translation_5211(fpe_data)
11: if length(fpe_data) < 16 then

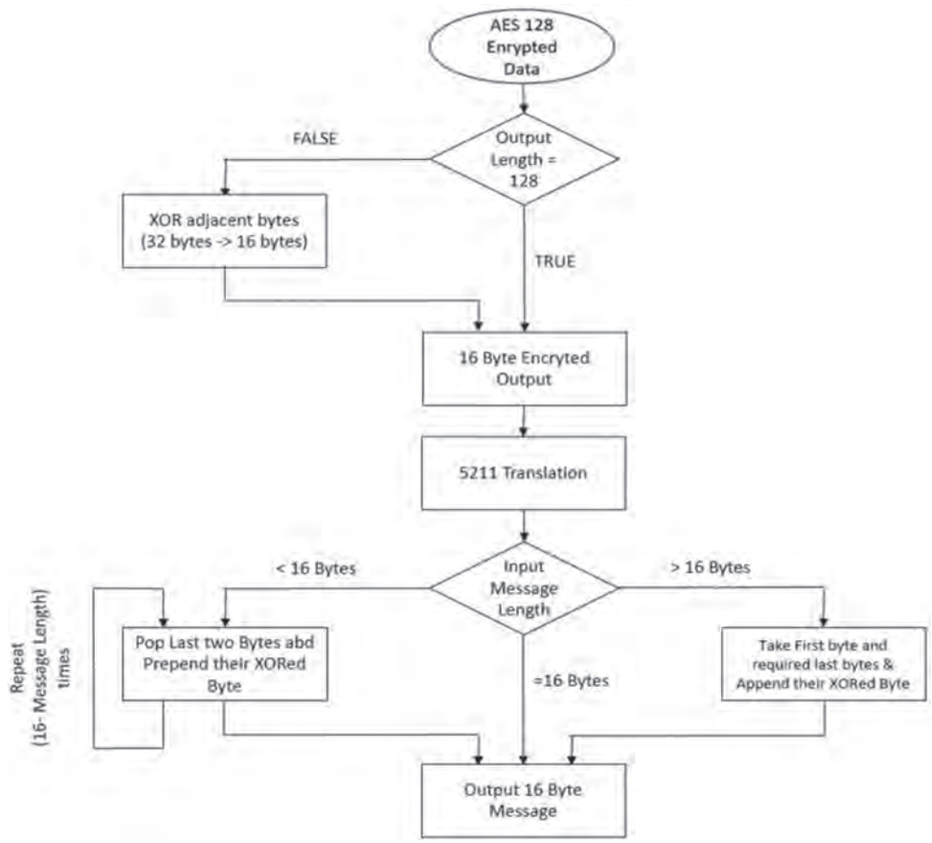

Fig. 4. Flowchart of proposed encryption scheme.

12: $i \leftarrow 1$
13: while $i < (16 - \text{length(fpe\_data)})$
14: //append the XORed output of last two bits at the start
15: $i \leftarrow i + 1$
16: else if length (fpe_data) > 16 then
17: //take a first byte and additional bytes from last and append the Xored byte
18: return fpe_data

Algorithm 1 defines the numeric data encryption in which the output of AES encryption is fed as input to obtain the desired output of the same format. In this algorithm variable input_data is given by AES encrypted output data, followed by fpe_data which is then XORed 16-byte AES encrypted data. Then again, the variable fpe_data get updated with a translation vector (fpe_data). Finally, the fpe_data will undergo a bitwise XOR operation and appending it to give the updated fpe_data.

The operations performed on numeric data are broken down into the following three functions:-

1. Block Cipher Generation
2. Length Synthesis and Translation
3. Length Preserving.

Algorithm 1 is implemented on two set of numeric data as shown in the example: A numeric data block having a size of 9 bytes taken as input. This numeric data block will undergo the following three functions.

(a) Block Cipher Generation

$$I = [7, 4, 5, 4, 5, 2, 6, 3, 4]$$

$$C = [0 \times 00, 0 \times 01, 0 \times 02, 0 \times 03, 0 \times 04, 0 \times 05, 0 \times 06, 0 \times 07, 0 \times 08, 0 \times 09, 0 \times 0a, 0 \times 0b, 0 \times 0c, 0 \times 0d, 0 \times 0e, 0 \times 0f]$$

Input (I) is encrypted using an AES-128 encryption algorithm to produce the ciphertext (E) as shown in the equation.

$$E = I \oplus C$$

$$E = [b6, 61, 46, 7c, ae, 69, 81, eb, 1f, b3, 1b, 3d, cd, cf, 98, fc] \quad (1)$$

(b) Length Synthesis and Translation method

In this step, XOR operation is performed by repetitively applying it on adjacent bytes of output E until the length of E reaches 16 bytes and then a 5211-translation is performed on 16 bytes of output data (E).

$$E = [13, 7, 2, 11, 4, 15, 9, 5, 14, 8, 10, 14, 1, 3, 1, 3]$$

After 5211-translation,

$$E = [8, 4, 1, 7, 2, 9, 6, 3, 8, 5, 6, 8, 1, 2, 1, 2]$$

(c) Length Preserving Function

This function works by repetitively applying the XOR operation on the starting and ending bytes of input data and then appending the result until the length of output E become equals to the length of the input data.
After first iteration,

$$E = [3, 8, 4, 1, 7, 2, 9, 6, 3, 8, 5, 6, 8, 1, 2]$$

After seven iterations,

$$E = [6, 9, 5, 7, 4, 3, 3, 8, 4]$$

The resultant ciphertext obtained by applying these three functions is given by 695743384 as shown in Figure 5.

4.2. Steps Required for Encrypting Input Data of Length Ranging from 17 to 19 Bytes

1. The increment in the length of ciphertext is needed, and the increase can be accomplished by looping the output for the addition of a required number of bytes.
2. For the first and the last byte, an XOR operation is defined. The result obtained will be appended to the end of the output and will lead to an increase in the length of output by one.
3. Hence, a loop operation is applied to it that will help to achieve the required result. A data block of 17 bytes is taken as input I as shown in Figure 6.

$$I = [1, 2, 3, 4, 5, 6, 7, 8, 9, 8, 7, 4, 5, 2, 6, 3, 4]$$

4.3. Encryption Solution for Encrypting Alphanumeric Data are Broken Down into the Following Six Functions

1. ASCII to Decimal Conversion
2. Block Cipher Generation (E)
3. Length Synthesis and Translation
4. Length Preserving Function
5. Mapping Function
6. Decimal to ASCII Conversion.

1. *ASCII to Decimal Conversion*: Alphanumeric input data mapped with its corresponding decimal data using ASCII Table.
2. *Block Cipher Generation* (E): By using a cipher key of 16 bytes a feistel block encryption and the AES-128 encryption algorithm is generated and evaluated.
3. *Length Synthesis and Translation*: In length synthesis, XOR operation is used repetitively on adjacent bytes of ciphertext E until the length of E reaches to 16 bytes. A 5211-translation method is applied to 16 bytes of the ciphertext (E).
4. *Length Preserving Function*: It is performed by repetitively applying XOR operation on starting and ending bytes and appending the result until the length of E becomes equal to the length of input data.
5. *Mapping Function*: Encryption for alphanumeric data follows the same procedure as of numeric data by using some more mapping functions. To convert an alphabet to

RESEARCH ARTICLE

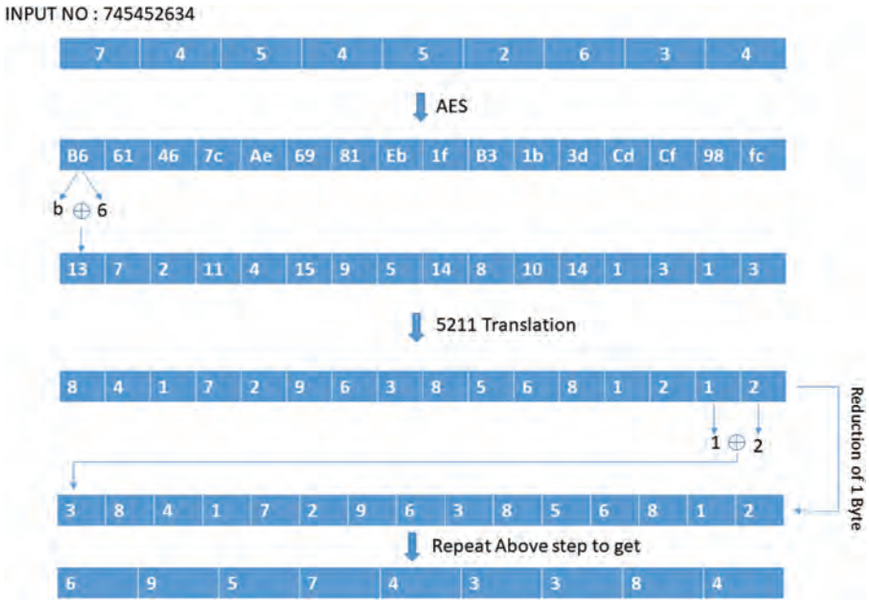

Fig. 5. Encrypting numeric data of 9 bytes.

its ASCII value and then to encrypt it using the above described numeric encryption. After encryption, it undergoes mapping using the Eq. (1) which provides a mapped 2-digit number.

$$\text{Floor}(\text{absolute}((\sin E[i]) * 100) \tag{2}$$

A new function is applied to the same word if the same character repeats in the iteration as shown in Eq. (2) again on the result of Eq. (1) which will differentiate the two same occurring characters.

$$\text{Floor}(\text{absolute}((E[i]/(1+E[i])) * 100) \tag{3}$$

This procedure applies to each character and finally mapped to the range of defined ASCII value for alphabets, numbers and special characters. All these results will be concatenated to get the final encrypted output with the same length as that of the input and with the same format.

6. *Decimal to ASCII Conversion*: Bytes of E are converted back to ASCII using ASCII chart.

ALGORITHM 2 (ALGORITHM FOR MAPPING).

1: **Input:** Encrypted ASCII values vector
2: **Output:** Encrypted ASCII mapped vector
3: **Start:**
4: $i \leftarrow 0$

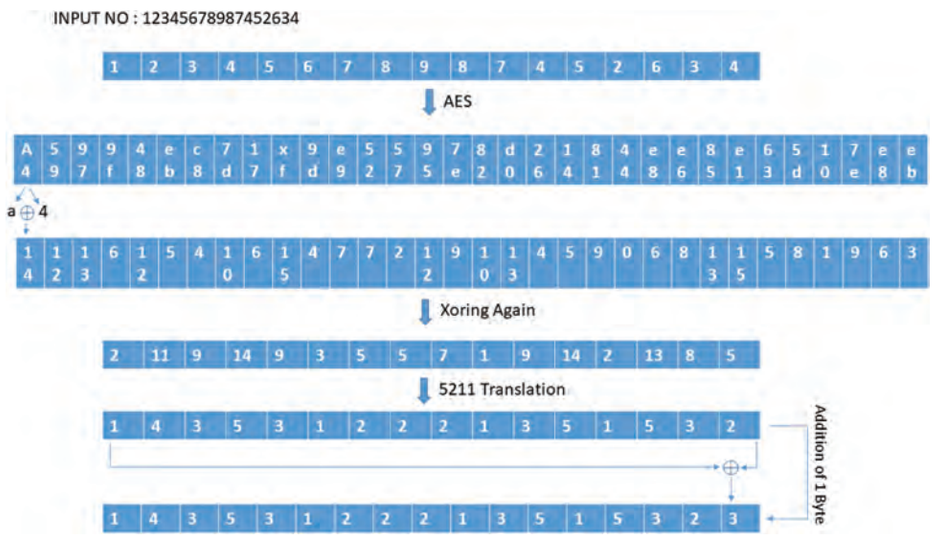

Fig. 6. Encrypting numeric data of 17–19 bytes.

RESEARCH ARTICLE

5: while $i <$ length(input)
6: input[i] $\leftarrow$ absolute(floor(sin(input[i] $*$ 100)))
7: $i \leftarrow i+1$
8: $i \leftarrow 0$
9: while $i <$ length(input)
10: if input[i] in input[:$i-1$] + input[$i+1$:]
11: input[i] $\leftarrow$ absolute(floor(input[i]/(1 + input[i])))
12: else
13: Continue
14: $i \leftarrow i+1$
15: $i \leftarrow 0$
16: while $i <$ length(input)
17: if input[i] < 33
18: input[i] $\leftarrow$ input[i] + 37
19: elseif input[i] >= 48 && input[i] <= 57
20: input[i] $\leftarrow$ input[i] + 13
21: else
22: Continue
23: $i \leftarrow i+1$
24: return input

In Algorithm 2 ASCII FPE encrypted vector is taken as input and results in ASCII mapped vector as output. The digits mapped in such a way that the resulting digits will come in desired ASCII range. We apply the algorithm to a plaintext given by governg@gmail.com. The steps shown below are used to obtain the desired ciphertext.

(a) *ASCII to Decimal Conversion*

$$\text{Converted Input } I = [103, 111, 118, 101, 114, 110, 103]$$

(b) *Block Cipher Generation + Length Synthesis + Translation + Length preserving function*

$$E = \text{Numeric FPE } (I[i])$$
$$= [433, 443, 300, 307, 777, 745, 433]$$

(c) *Mapping Function*

$$E = [98, 41, 100, 77, 86, 43, 65]$$

(d) *Decimal to ASCII Conversion*

$$E = [\text{'}b\text{'}, \text{')'}, \text{'}d\text{'}, \text{'}M\text{'}, \text{'}V\text{'}, \text{'+'}, \text{'}A\text{'}]$$

The resultant ciphertext obtained by applying the steps defined above =b)dMV+A@gmail.com.

5. ENVIRONMENTAL SETUP AND RESULT ANALYSIS

5.1. Environment Setup

The paper uses the python scripts for evaluating the performance of newly proposed algorithms. A database of size 467 KB is created containing test cases and their encrypted results. The database is running on the SQL server, and SQL server runs on a 64-bit operating system with 4 GB of RAM. SQL queries are used to write and read data from SQL tables through Python script by using Python 3.6 Anaconda Distribution. A database is created containing the numeric data of length 1 to 19 that includes Aadhaar card number, CVV number, mobile number, Credit/Debit/Master Card (all possible cards), social security number, IPV4 (IP Address), pin codes, email addresses. The proposed algorithm is used to encrypt the numeric and alphanumeric data stored in the database.

Table I. Encryption timings for various data formats.

	Data type	Encryption time (seconds)
1	AADHAR numbers (12-digit)	155.6895
2	CCV numbers (3-digit)	248.6263
3	IPv4 addresses (8-digit)	249.9849
4	SSN numbers (9-digit)	251.7890
5	Email ids (alphanumeric) (variable length)	242.0550

5.2. Result Analysis

Encryption time for various data formats is analyzed and calculated using the algorithm described above. The graph shows the time required to encrypt and store the data in remote databases. 1350 number of iterations are used to calculate a considerable amount of lapse time. Different data formats are encrypted having the same number of iterations.

Reading Data from Database: "SELECT ⟨Data Name⟩ from ⟨Table Name⟩."

Writing Data to Database: "INSERT INTO ⟨Table Name⟩ (Data Names) VALUES (Encrypted Data)."

It has observed that aadhar card number takes about 155.6895 seconds to encrypt and to store the encrypted data in the database for encrypting 1350 Aadhaar card numbers. Similarly, timing analysis for other data types is formulated in the table and plotted.

Table II. Encryption time analysis for different length of data.

	Data length (digits)	Encryption time (seconds)
1	3	6.182545185089111
2	4	6.094269037246704
3	5	6.089296579360962
4	6	6.041971921920776
5	7	6.079653978347778
6	8	6.032282829284668
7	9	6.022290706634521
8	10	6.023602247238159
9	11	5.922979831695557
10	12	5.902327775955278
11	13	5.942899703979492
12	14	5.742136955261230
13	15	6.431885719299316
14	16	5.702696561813354
15	17	9.852843999862671
16	18	10.04500222206150

Table III. Encryption time analysis for numeric and alphanumeric data.

	Iterations	Numeric data encryption time (in seconds)	Alphanumeric data encryption time (seconds)
1	1024	1.780965089797973	6.5635595321655271
2	2048	1.210676908493042	12.997919321060183
3	3072	1.810962438583374	19.772315263748174
4	4096	2.370525360107422	26.323584794998170
5	5120	3.020329475402832	32.848187446594242
6	6144	3.541921615600586	39.279468297958374
7	7168	4.132498025894165	45.895224809646606
8	8192	4.741972446441650	52.778207778930664
9	9216	5.332676410675049	60.232902765274050
10	10240	5.952418565750122	65.806844472885131

5.3. Encryption Time Analysis for Numeric Data Types of Different Length

The graph in Figure 8 shows the encryption time for format preserved numeric data of variable length ranging from (3–18) digits. It evaluates a total number of 10240 sample iterations for calculating a considerable amount of span time. The obtained result analyses that the size of the number increases from 3-digit to 16-digit, encryption time decreases. Also, it analyses that the size of the input digit is risen from 16-digit to 18-digit, encryption time increases. It happens due to the following reason as we know AES outputs the encrypted data of 128 bits, which on applying XOR operation to adjacent bits provide 16 bytes data and which on further applying continuous looping of XORed bits provide an output of numbers with length less than 16.

However, in the case of numbers having a length higher than 16, encryption time from AES increases because output increases its size to 256 bits by using the key of 128 bits. Hence, it arises a need to implement XOR operation twice and then looping of XORed bits will provide the result of the desired length (digits) number. Therefore, the algorithm explains this anomaly.

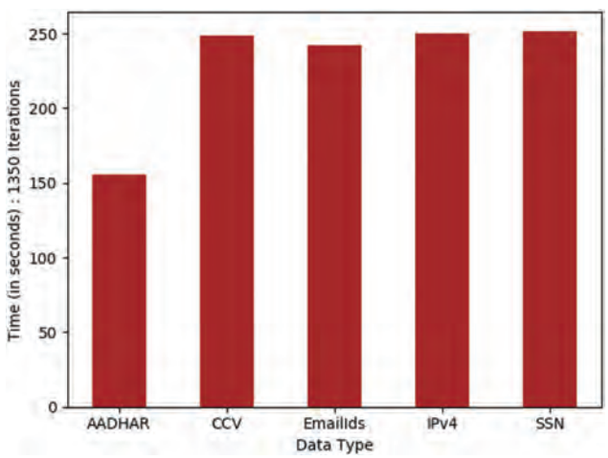

Fig. 7. Encryption time analysis for various data formats.

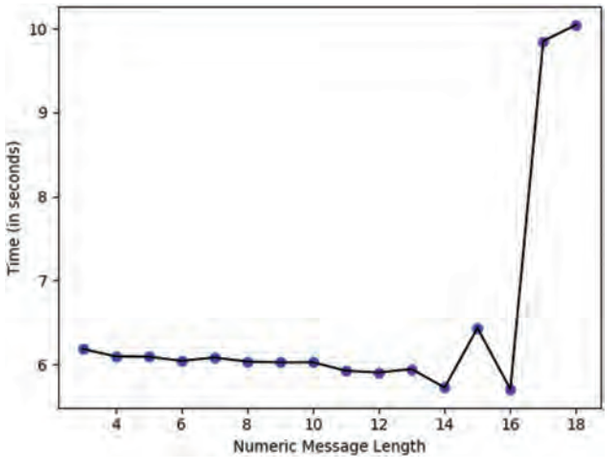

Fig. 8. Encryption time analysis for numeric data types of different length.

5.4. Comparing Encryption Time of Numeric and Alphanumeric Data Over Different Input Size

The graph shows the comparison between encryption time required to encrypt a numeric data and alphanumeric data. After analysis, we found that alphanumeric data takes more time than a numeric data for the same number of input size and both data types have an increase in their encryption time as input size increase. The algorithm can quickly explain the reason why alphanumeric data is taking more time to encrypt. As, we can infer from the method of encryption of alphanumeric data which includes conversion of alphanumeric data to its ASCII form, then implementing numeric encryption on them. After encryption, mapping of data takes place so that output should not exceed ASCII values outside a range of ASCII values of alphabets, numbers and special characters. After mapping, inverse conversion method is applied back to alphanumeric data to obtain encrypted format preserved data.

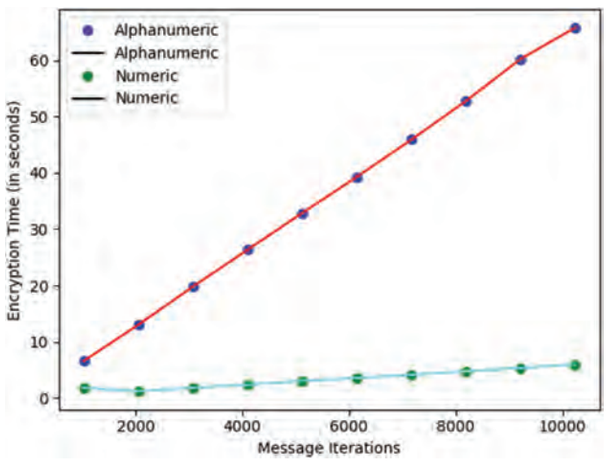

Fig. 9. Comparing encryption time of numeric and alphanumeric data over different input size.

RESEARCH ARTICLE

5.5. Analyzing the Complexity of Proposed Encryption Algorithm

The complexity of an algorithm is defined regarding time complexity and space complexity of that algorithm. Space complexity deals with the amount of space required by the algorithm as a function of input during execution. Similarly, Time complexity deals with the time needed by the algorithm as a function of input during execution. Evaluating these complexities with advanced mathematics is a tedious job. Hence, Big O Notation is used in computer science to describe the performance of the algorithm's complexity. Big O Notation describes the worst-case scenario, explaining the worst-case space and time complexity.

5.6. Space Complexity

The proposed algorithm works on AES encryption technique followed by a Feistel-cycle walking algorithm that undergoes a worst-case space allocation of 32 bytes (256 bits). The given input varies from 1 to 19 for numeric data and 1 to 3 in alphanumeric data (Each alphanumeric character's ASCII value can have a maximum value of 127, i.e., a 3-digit number). Hence, after AES encryption, the output can have maximum space allocated to be of 32 bytes (for input of range 17–19). Therefore, space complexity as a function of input will be ordering O(1).

5.7. Time Complexity

In the AES algorithm, execution time remains the same for all inputs in range (1–16) and double for those in range (17–19). XORing in AES to reduce the output back to 16 bytes will lead to the time complexity of $O(N/2)$. 5211 Translation executed on 16-byte output (XORed AES output) hence does not affect the Time complexity. Mapping of numeric data will again take an execution time to format the output length according to input length. Therefore, worst-case Time Complexity for the proposed algorithm for numeric data will be $O(N)$ where N is a number of input bytes. For evaluating the values of alphanumeric data, further mapping takes place through a mathematical expression that will add an equal time to all the inputs and depend on input byte size. Hence, for alphanumeric input, time complexity will be ordering $O(N)$ where N is a number of input bytes.

5.8. The Strength of the Proposed Algorithm

As the encryption is from numeric to numeric and alphanumeric to alphanumeric; now it becomes hard for an adversary to crack the information because it appears as if the encrypted data is itself in original data format. It happens because encrypted data retains its original form as of the plain text. Hence, the algorithm maintains the data length of the data to be stored in data fields and doesn't require any change in the encrypted data fields of the numeric and alphanumeric form.

6. CONCLUSION

Data Security is becoming a primary concern of today's growing technology. Intruders are a significant threat for the organizational data stored in server databases and transmitted over the network. The aim of the proposed research presented in the paper is to provide a practical solution that can not only offer secure database system but also reduce the output data size such that we can transmit more data in limited bandwidth channel. Adding security to remote database eases the purpose and lead to secure transmission between remote and server database. Encrypting data on remote side database leads to fast retrieval of information with least risk of data loss. In this paper, we implement an irreversible algorithm for encrypting data from the remote database with the same format as that of input data. Preserving data format leads to high security and reduction in the size of output as compared to standard encryption techniques such as AES, DES, 3DES. The proposed algorithm is an efficient example of Generalized Feistel Network that implements AES followed format preservation. The algorithm proposed in this paper can encrypt all the numerical data of size ranging from 1 byte to 19 bytes. Numeric data such as credit card numbers, Aadhaar card numbers, SSNs, CVV keys, IPv4 (IP Addresses) and alphanumeric data such as email ids are implemented in the paper for secure data transmission. For remote side databases, SQL is used. In this paper, encryption time is compared to different input length numeric data. Execution time for numeric data and alphanumeric data at a constant iteration value, are also examined in the paper. Hence, the research paper provides a new and efficient encryption algorithm by maintaining the format of plaintext data as well as enhances the security of data stored in the databases.

References

1. D. Manivannan and R. Sujarani, Lightweight and secure database encryption using TSFS algorithm, *2010 International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, July **(2010)**, pp. 1–7.
2. D. S. A. Elminaam, H. M. A. Kader, and M. M. Hadhoud, *IJCSNS International Journal of Computer Science and Network Security* 8, 280 **(2008)**.
3. W. Trappe, Introduction to Cryptography with Coding Theory, Pearson Education India **(2006)**.
4. I. D. Gorbenko and Y. I. Gorbenko, Applied Cryptology, Theory, Practice, Application: A Monograph, Fort, Kh. **(2012)**.
5. H. C. Van Tilborg and S. Jajodia (eds.), Encyclopedia of Cryptography and Security, Springer Science and Business Media **(2014)**.
6. L. R. Knudsen and M. Robshaw, The Block Cipher Companion, Springer Science and Business Media **(2011)**.
7. J. Katz, A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography, CRC Press **(1996)**.
8. Z. Yong-Xia, The technology of database encryption, *2010 Second International Conference on Multimedia and Information Technology*, IEEE, April **(2010)**, pp. 268–270.
9. L. Bouganim and P. Pucheral, Chip-secured data access: Confidential data on untrusted servers, *Proceedings of the 28th International*

Conference on Very Large Data Bases, VLDB Endowment, August **(2002)**, pp. 131–142.
10. S. Sesay, Z. Yang, J. Chen, and D. Xu, A secure database encryption scheme, *2005 Second IEEE Consumer Communications and Networking Conference, 2005, CCNC*, IEEE, January **(2005)**, pp. 49–53.
11. K. Kaur, K. S. Dhindsa, and G. Singh, Numeric to numeric encryption of databases: Using 3 Kdec algorithm, *IEEE International Advance Computing Conference, 2009, IACC 2009*, IEEE, March **(2009)**, pp. 1501–1505.
12. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, Order preserving encryption for numeric data, *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, ACM, June **(2004)**, pp. 563–574.
13. M. Brightwell and H. Smith, Using datatype-preserving encryption to enhance data warehouse security, *20th National Information Systems Security Conference Proceedings (NISSC)*, October **(1997)**, pp. 141–149.
14. T. Spies, Format preserving encryption, Unpublished white paper, www. voltage. com Database and Network Journal (December 2008), Format preserving encryption: www. voltage. com **(2008)**.
15. E. Brier, T. Peyrin, and J. Stern, BPS: A format-preserving encryption proposal, http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-spec.pdf **(2010)**.
16. M. Liskov, R. L. Rivest, and D. Wagner, Tweakable block ciphers, *Annual International Cryptology Conference*, Springer, Berlin, Heidelberg, August **(2002)**, pp. 31–46.
17. The pub, N. F. 197: Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, US Department of Commerce/NIST, November **(2001)**, Available from the NIST website.
18. M. Dworkin, Recommendation for block cipher modes of operation. Methods and techniques (No. NIST-SP-800-38A). National Inst of Standards and Technology Gaithersburg Md Computer Security Div. **(2001)**.
19. P. Rogaway, Evaluation of some block cipher modes of operation, *Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan* **(2011)**.
20. P. FIPS, *National Institute of Standards and Technology* 25, 1 **(1999)**.
21. W. C. Barker, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Revised 19 May 2008. NIST Special Publication, 800-67.
22. National Institute of Standards, and Technology (US), Technology Administration, *Secure hash standard*, Vol. 180, No. 1, US Department of Commerce, Technology Administration, National Institute of Standards and Technology **(1993)**.
23. K. Chitra and S. Vidhya, *IOSR Journal of Computer Engineering* **(2013)**.
24. K. Mallaiah, S. Ramachandram, and S. Gorantala, Performance analysis of Format Preserving Encryption (FIPS PUBS 74-8) over block ciphers for numeric data, *2013 4th International Conference on Computer and Communication Technology (ICCCT),* IEEE, September **(2013)**, pp. 193–198.
25. R. Agbeyibor, J. Butts, M. Grimaila, and R. Mills, Evaluation of format-preserving encryption algorithms for critical infrastructure protection, *International Conference on Critical Infrastructure Protection*, Springer, Berlin, Heidelberg, March **(2014)**, pp. 245–261.
26. V. T. Hoang and P. Rogaway, On generalized Feistel networks, *Annual Cryptology Conference*, Springer, Berlin, Heidelberg, August **(2010)**, pp. 613–630.
27. M. Naor and O. Reingold, *Journal of Cryptology* 12, 29 **(1999)**.
28. M. Luby and C. Rackoff, *SIAM Journal on Computing* 17, 373 **(1988)**.
29. J. Black and P. Rogaway, Ciphers with arbitrary finite domains, *Cryptographers' Track at the RSA Conference*, Springer, Berlin, Heidelberg, February **(2002)**, pp. 114–130.
30. M. Bellare and P. Rogaway, On the construction of variable-input-length ciphers, *International Workshop on Fast Software Encryption*, Springer, Berlin, Heidelberg, March **(1999)**, pp. 231–244.
31. P. Chandrashekar, S. Dara, and V. N. Muralidhara, Efficient format preserving encrypted databases, *2015 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT),* IEEE, July **(2015)**, pp. 1–4.
32. S. Patel, Z. Ramzan, and G. S. Sundaram, Efficient constructions of variable-input-length block ciphers, *International Workshop on Selected Areas in Cryptography*, Springer, Berlin, Heidelberg, August **(2004)**, pp. 326–340.
33. B. Cui, B. Zhang, and K. Wang, A data masking scheme for sensitive big data based on format-preserving encryption, *2017 IEEE International Conference on Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC)*, IEEE, July **(2017)**, Vol. 1, pp. 518–524.
34. K. Chitra and S. Vidhya, Format preserving encryption for small domain algorithm, *Proceedings of International Conference on Computing and Intelligence Systems* **(2015)**, pp. 1379–1383.
35. S. K. Bhatnagar, Securing data-at-rest, Literature by Tata Consultancy Services.
36. A. J. Elbirt, W. Yip, B. Chetwynd, and C. Paar, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 9, 545 **(2001)**.
37. B. C. Fung, K. Wang, and P. S. Yu, *IEEE Trans. Knowl. Data Eng.* 19, 711 **(2007)**.
38. N. H. Arshad, S. N. Tahir Shah, A. Mohamed, and A. M. Mamat, *International Journal of Applied Mathematics and Informatics* 1, 15 **(2007)**.
39. A. Sachdev, and M. Bhansali, *International Journal of Computer Applications* 67 **(2013)**.
40. J. J. Hwang, H. K. Chuang, Y. C. Hsu, and C. H. Wu, A business model for cloud computing based on a separate encryption and decryption service, *2011 International Conference on Information Science and Applications (ICISA)*, IEEE, April **(2011)**, pp. 1–7.
41. E. Thambiraja, G. Ramesh, and D. R. Umarani, *International Journal of Advanced Research in Computer Science and Software Engineering* 2 **(2012)**.
42. D. S. A. Minaam, H. M. Abdual-Kader, and M. M. Hadhoud, *IJ Network Security* 11, 78 **(2010)**.
43. J. Cui, L. Huang, H. Zhong, C. Chang, and W. Yang, *International Journal of Innovative Computing, Information, and Control* 7, 2291 **(2011)**.
44. Z. Liu, X. Chen, J. Yang, C. Jia, and I. You, *Journal of Network and Computer Applications* 59, 198 **(2016)**.
45. M. Nagendra and M. C. Sekhar, *International Journal of Software Engineering and Its Applications* 8, 287 **(2014)**.

Received: 13 September 2018. Accepted: 19 September 2018.

RESEARCH ARTICLE