Research Journal of Applied Sciences, Engineering and Technology 7(21): 4476-4489, 2014
DOI:10.19026/rjaset.7.824
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2014 Maxwell Scientific Publication Corp.
Submitted: December 01, 2013 Accepted: January 07, 2014 Published: June 05, 2014

Research Article

A Novel Dynamic Data Blocking Mechanism for Symmetric Cryptosystems

[1, 2]Ijaz Ali Shoukat, [1]Kamalrulnizam Abu Bakar and [1]Subariah Ibrahim
[1]Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia, 81310, Johor Bahru, Malaysia
[2]Computer Science Department, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia

Abstract: This study contributes a dynamic data blocking mechanism to replace the fixed (static) data blocking mechanism in symmetric cryptosystems. The robustness of cryptosystems relies on dynamicity and probability to provide sufficient randomness. Any encryption method is considered as secure as it retains randomness properties. Current cryptographic algorithms (AES, DES) utilize fixed (static) data blocking mechanism that enable practical cracking of DES and academic cracking of AES-256 up to full 14 rounds with almost practical complexity of (q.2^{67}) queries through related-key distinguisher attack which works for 1 key out of 2^{35} keys with 2^{120} data and time complexity trials. Fixed (static) data blocking mechanism triggers the applicability of this kind of security attacks on symmetric cryptosystems by offering computable probability range. In this study, we proposed a dynamic (variable) data blocking mechanism to achieve robust probabilistic randomness in computing of number of data blocks and their number of bits in symmetric cryptosystems for the enhancement of security strength. The ultimate objective behind this proposed dynamic data blocking mechanism is to produce different number of data blocks with different number of bits in order to lead the complexity (probability) of block partitioning as a NP-hard problem ($P \neq NP$ - widely believed for which no efficient algorithm exists). In our proposed scheme the diversity of both parameters (block numbers, block bits) is dependent on a plaintext data and key. We applied superlative tactics of digital logic and mathematics in order to evaluate and justify our proposed Dynamic Data Blocking Mechanism (DDBM).

Keywords: Brute force attack, cryptographic algorithms, decryption, dynamic data blocking mechanism, symmetric encryption

INTRODUCTION

The art of secret writing can be applied by either utilizing the steganography or cryptographic schemes. But cryptographic practices are robust to stenographic methods as reported by Shoukat *et al*. (2011). Recent trend of encrypting confidential information is getting potential fame under hybrid encryption schemes with full of security and privacy. A generic Hybrid Encryption System (HES) has been proposed (Shoukat *et al*., 2013) which works optimally with joint combination of symmetric and asymmetric algorithms and offers a complete set of security objectives as recommended by National Institute of Standard and Technology (NIST). In any case of encrypting the information either with just symmetric or hybrid cryptosystems, the security of symmetric algorithms cannot be compromised. Cryptography deals with symmetric, asymmetric and hybrid schemes, however, symmetric block ciphers are significantly preferred in order to save battery power, CPU time and memory requirements (Elminaam and Abdul-Kader, 2010). Due to these prominent advantages, symmetric encryption is the key choice of governmental bodies for storing or communicating confidential information especially related to banking transactions, ministry of finance, police departments, scientific laboratories, Army units, Navy Force units, Air Force bodies, hospitals and assurance companies. The replacement of fixed data blocking is an active issue with present encryption standards (AES, DES). The widely implemented block cipher AES uses fixed sized data block (128 bits) and fixed 8 bits substitution boxes (Young and Yang, 2010). The other foremost Data Encryption Standard (DES) uses 64 bits fixed block sizes, permutations and fixed substitution for encrypting data (Singh and Bansal, 2010).

The successful philosophy of differential cryptanalysis attacks is its larger probability than the randomly occurred permutations on fixed size (length) data blocks (Lu, 2010; Lu, 2008). At present, advanced cryptanalysis techniques are creating critical situation against the security satisfaction of standard-cryptosystems (AES, DES). Data Encryption Standard

Corresponding Author: Ijaz Ali Shoukat, Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia, 81310, Johor Bahru, Malaysia

(DES) is insecure as it had been cracked earlier just within 4.5 days (Wobst, 2001) even the security satisfaction of Triple DES (TDES) is also vulnerable as witnessed by its practical cracking within 800 days on a machine that can execute 50 billion keys/sec (Alanazi *et al*., 2010). Moreover, Biryukov *et al*. (2010) reported that, TDES has already been cracked with key complexity of 2^{56} bits as claimed by Biryukov *et al*. (2010). Similarly, the highly reputed algorithm (AES-192/156 bits key) (Biryukov *et al*., 2010) has been academically considered as insecure as claimed by Biryukov and Khovratovich (2009). Furthermore, the authors of study (Ferguson *et al*., 2001) performed significant effort with a success of 2^{44} rather to 2^{72} trials in order to crack the 6 rounds of AES. The 7 rounds of 192 and 256 key AES has also been cracked with 2^{32} plain text choosing probability under the complexity of 2^{140} encryption tries (Gilbert and Minier, 2000). The 5 rounds of AES with 128 bits key have been cracked with 2^{46} chosen plain texts under 2^{46} encryption tries and 6 rounds with 2^{78} encryption tries with utilization of 2^{78} chosen plain texts when Boomerang attack is implemented by Biryukov (2004). Effort remained continue and consequently, in 2009 Biryukov and his fellows successfully cracked the AES with 192 and 256 bits key as claimed in studies (Biryukov and Khovratovich, 2009; Biryukov *et al*., 2009). In 2009 meet-in-the-middle attack is applied by Huseyin Demirci and his fellows on 7-round AES which seems to be superior in time complexity rather than the previous reported attacks (Demirci *et al*., 2009). Against 9 rounds of AES-256 2^{39} encryption tries can crack it by calculating the XOR of encrypted plaintext under 2 related keys in many ways and 10 rounds can be cracked academically in the same way with 2^{45} encryption tries (Biryukov *et al*., 2010). The most shocking situation is the academic cracking of AES-256 up to full 14 rounds with almost practical complexity of (q.2^{67}) queries through related-key distinguisher attack which works for 1 key out of 2^{35} keys with 2^{120} data and time complexity trials (Biryukov and Khovratovich *et al*., 2009).

Therefore, the security satisfaction of Advanced Encryption Standard (AES) is minimizing its realistic attraction to select it as a confident verdict due to its academic cracking. Furthermore, the Biryukov and Dunkelman *et al*. (2010) also claimed that AES security strength is not as strong as it is believed or conveyed. The actual cause of this kind of cracking with AES and DES is the utilization of fixed data block partitioning because permutation on fixed length data blocks is not an optimal case against hefty probability of differential cryptanalysis attacks (Lu, 2010; Lu, 2008). The authors of study (Shoukat and Baker, 2013) claimed that new design of symmetric cryptosystem must be evaluated under dynamicity constraints like dynamicity in data blocking step, dynamic selection of encryption operations, pseudo random based key state's updating and operational pseudo randomness in encryption algorithm (s). The implementation of dynamic data blocking mechanism under some probabilistic randomness can provide sufficient resistance with series of large optimizations which may also be robust to provide initial safe-guard against these kinds of powerful attacks. Therefore, such kind of situation necessitates the timely development of dynamic data blocking mechanism in order to create sufficient resistance against modern cryptanalysis of symmetric cryptosystems.

This study aims to convey a dynamic data blocking mechanism for symmetric cryptosystems in order to create dynamic data blocks rather to fixed sized data blocks. The ultimate objective behind this proposed dynamic data blocking mechanism is to produce different number of data blocks with different number of bits against different set of plaintext and secret key in order to achieve the complexity of probability computation as likely NP-hard problem ($P \neq NP$ - widely believed for which no efficient algorithm exists). For hard problems, this hypothesis ($P \neq NP$) is broadly believed but it has not been proven yet (Gomes and Williams, 2005; Taslaman, 2012). In proposed scheme, for a selected plaintext and key, the all blocks contain same number of bits but without this knowledge that how many bits are in a block and how much are total blocks? Similarly for another plaintext and key the number of blocks and their bits will be different but all blocks contain same number of bits which means number of data blocks and the length of data blocks are dependent on the set of plaintext and secret key. Most significantly, this study has putt-forwarded the presented idea of dynamic block creation for future cryptosystems, however this scheme can also be utilized to input such dynamic data blocks to any widely known symmetric cryptosystem with just minor changes. Our proposed idea is enriched with novelty of dynamic data block creation with vital scope in the development of forthcoming cryptographic algorithms to kick off fixed block creation philosophy.

METHODS AND PROCEDURES

The proposed dynamic data blocking mechanism relies on a secret number that is generated from the initial 160 bits long secret key. Encryption key should not be too lengthy or too short as agreed by Shoukat *et al*. (2011). On the other hand, Institute of Standard and Technology (SP800-57, NIST 2005a) also recommends minimum key length ranged from 112 to 128 bits up to year 2030 (Une and Kanda, 2007). In order to find this secret number cracker needs a secret key (160 bits) that is quite secure against brute force attack according to the specified length of ISO and NIST. This secret number is just like a probabilistic random number that is considered as computationally hard problem in cryptography. The data blocking is dependent on this random number is sufficiently

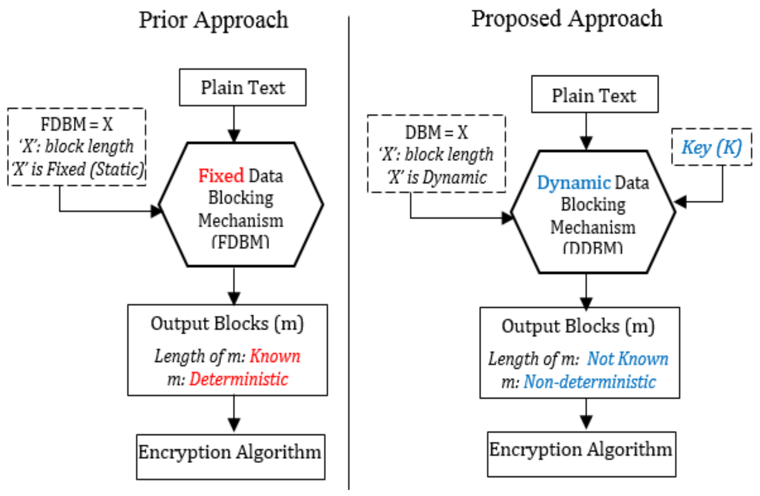


Fig. 1: Overview of proposed and prior schemes with comparison

dynamic. The proposed dynamic data blocking mechanism can generate two types of block ranges. The first block range lies between 80 to 128 bits but this is not a recommend range. The other block range lies between 128 to 256 bits block sizes, is recommended range to get optimal security in any block cipher. The chosen 160 bits binary key requires the following conditions to get optimal security:

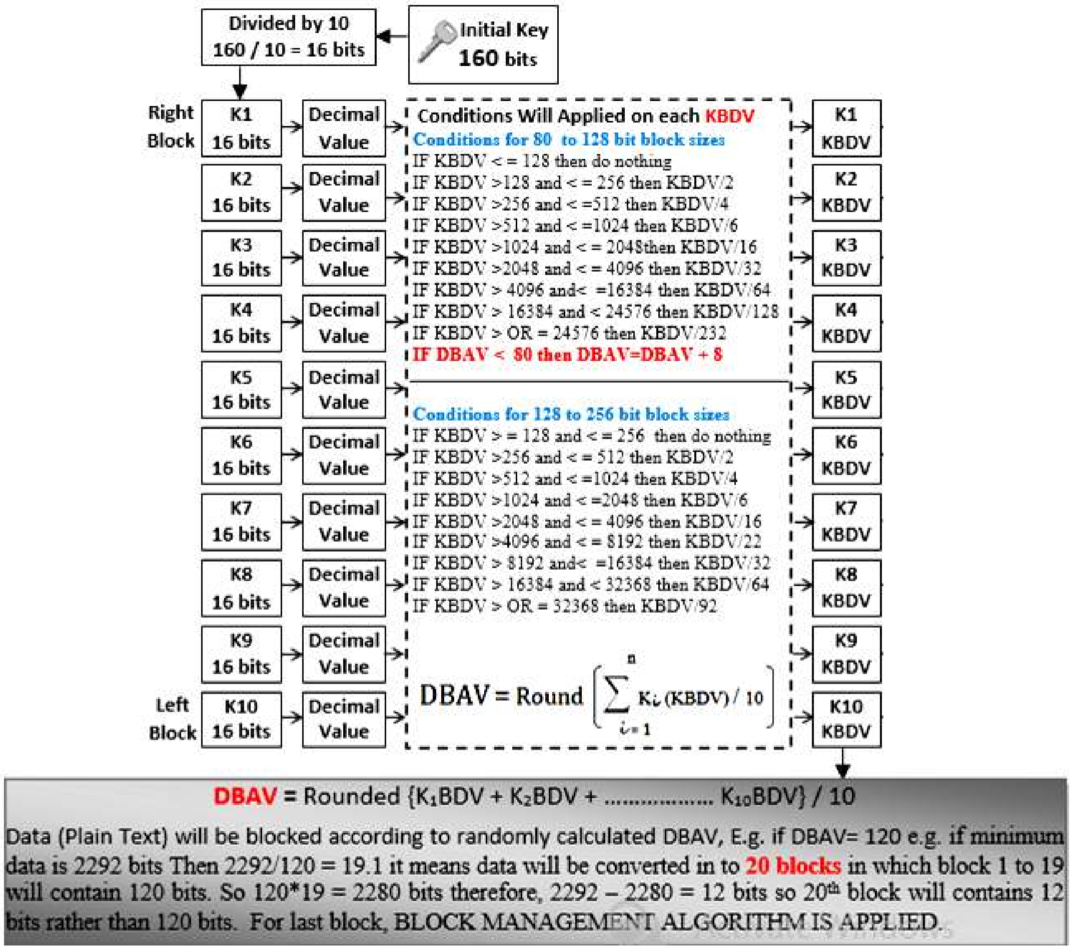
- Key length must be 160 bits in binary form and should be selected by the user
- Key cannot contain all zeroes
- Key cannot contain all ones
- It is too good if no more than six continuous pattern of 0's or 1's should be repeated in initial 160 bits key. This is the preferred condition to get highest security satisfaction but it is up to user whether to follow or not

Figure 1 shows the flow and comparison between prior and proposed data blocking mechanisms which start working by receiving of plaintext. After receiving the plain text value of block length (*X*) is input to the mechanism in order to partition the plaintext into output blocks as depicted in Fig. 1 through arrow flow. In prior approach, plain text is converted according to the fixed (static) value which is known to the cracker. However in proposed approach the value of *'X'* is dynamically calculated with the involvement of Key (*K*) and this value of *X* is dynamic (random) and un-known for the cracker. Moreover, in prior approach, parameter m is deterministic but in proposed approach *m* is non-deterministic. Proposed idea of dynamic block creation revolves around the novel philosophy, how dynamic (randomized) value of *X* and *m* can be generated and how *m* can be non-deterministic in block partitioning step of symmetric encryption algorithm.

Logical key blocking mechanism: Dynamic data blocking follows a secret (random) digit that can be calculated from initial 160 bits key. In order to find out this probabilistic random digit the initial 160 bits key is converted into 10 key blocks having 16 bits each. This key partitioning is logical and its mutual security strength is not affected due to this logical partitioning. The dynamic data blocking procedure with logical partitioning of initial 160 bits key is discussed step by step with their operational flow in Fig. 2. Logical key blocking mechanism follows various steps. In step 1, the initial 160 bits key is divided by 10 to get 10 key blocks each having 16 bits length. The first right side key block is designated with K1, the second right side key block is designated with K2 and so on up to 10th key block designated with K10. In 2nd step, a decimal value is calculated against each key block (K1, K2, K3…… K10). This calculated decimal value is referred as Key Block Decimal Value (KBDV). Each KBDV always lies under the given conditions to manage block sizes. Two ranges of block sizes can be created under this dynamic data blocking mechanism through given block size management conditions. The block size management conditions are as follows.

Conditions for (80 to 128) bit's block sizes:

IF KBDV < = 128 then do nothing
IF KBDV >128 and < = 256 then KBDV/2
IF KBDV >256 and < = 512 then KBDV/4
IF KBDV >512 and < = 1024 then KBDV/6
IF KBDV >1024 and < = 2048 then KBDV/16
IF KBDV >2048 and < = 4096 then KBDV/32
IF KBDV >4096 and < = 16384 then KBDV/64
IF KBDV >16384 and <24576 then KBDV/128
IF KBDV >OR = 24576 then KBDV/232
IF DBAV <80 then DBAV = DBAV + 8 (1)


Divided by 10
160 / 10 = 16 bits
Initial Key
160 bits
Right Block
Left Block
K1 16 bits
K2 16 bits
K3 16 bits
K4 16 bits
K5 16 bits
K6 16 bits
K7 16 bits
K8 16 bits
K9 16 bits
K10 16 bits
Decimal Value
Conditions Will Applied on each KBDV
Conditions for 80 to 128 bit block sizes
IF KBDV < = 128 then do nothing
IF KBDV >128 and < = 256 then KBDV/2
IF KBDV >256 and < =512 then KBDV/4
IF KBDV >512 and < =1024 then KBDV/6
IF KBDV >1024 and < = 2048then KBDV/16
IF KBDV >2048 and < = 4096 then KBDV/32
IF KBDV > 4096 and< =16384 then KBDV/64
IF KBDV > 16384 and < 24576 then KBDV/128
IF KBDV > OR = 24576 then KBDV/232
IF DBAV < 80 then DBAV=DBAV + 8
Conditions for 128 to 256 bit block sizes
IF KBDV > = 128 and < = 256 then do nothing
IF KBDV >256 and < = 512 then KBDV/2
IF KBDV >512 and < =1024 then KBDV/4
IF KBDV >1024 and < =2048 then KBDV/6
IF KBDV >2048 and < = 4096 then KBDV/16
IF KBDV >4096 and < = 8192 then KBDV/22
IF KBDV > 8192 and< =16384 then KBDV/32
IF KBDV > 16384 and < 32368 then KBDV/64
IF KBDV > OR = 32368 then KBDV/92
DBAV = Round [∑ⁿᵢ₌₁ Kᵢ (KBDV) / 10]
K1 KBDV
K2 KBDV
K3 KBDV
K4 KBDV
K5 KBDV
K6 KBDV
K7 KBDV
K8 KBDV
K9 KBDV
K10 KBDV
DBAV = Rounded {K₁BDV + K₂BDV + ... K₁₀BDV} / 10

Data (Plain Text) will be blocked according to randomly calculated DBAV, E.g. if DBAV= 120 e.g. if minimum data is 2292 bits Then 2292/120 = 19.1 it means data will be converted in to 20 blocks in which block 1 to 19 will contain 120 bits. So 120*19 = 2280 bits therefore, 2292 – 2280 = 12 bits so 20th block will contains 12 bits rather than 120 bits. For last block, BLOCK MANAGEMENT ALGORITHM IS APPLIED.

Fig. 2: Logical key blocking mechanism

Conditions for 128 to 256 bit's block sizes:

IF KBDV > = 128 and < = 256 then do nothing
IF KBDV >256 and < = 512 then KBDV/2
IF KBDV >512 and < = 1024 then KBDV/4
IF KBDV >1024 and < = 2048 then KBDV/6
IF KBDV >2048 and < = 4096 then KBDV/16
IF KBDV >4096 and < = 8192 then KBDV/22
IF KBDV >8192 and< = 16384 then KBDV/32
IF KBDV >16384 and <32368 then KBDV/64
IF KBDV >OR = 32368 then KBDV/92 (2)

Figure 2, represents the logical key blocking procedure step by step. In first step 160 bits key is selected and divided by 10 that results the bit length (16 bits) of sub logical key block. In this way, in 2nd step, the key converts to 10 logical blocks and designates with unique variables K_1, K_2, K_3.... K_n, respectively. Where the right most 16 bits of binary key are assigned to K1 and the left most 16 bits are assigned to K10. In next step, a decimal value is calculated against each logical key blocks (K_1, K_2, K_3.... K_n) and filters by Round function [Round $\{\sum^{n}_{i=1} K_i\ (KBDV/10)\}$] After that, any desired set of conditions Eq. (1) or (2) is applied on K_1, K_2, K_3.... K_n. This function returned the Data Block Average Value (DBAV) which is used to partition the plain-text data into blocks. The first block size management condition (1) is not recommended; it is an optional trait for small devices. However, the second block management condition (2) is recommended for getting optimal security that creates block sizes from 128 bits up to 256 bits. It means the length of each block remains in between this specified limit but it is not known to the cracker; what is the bit length of each block. The bit length is dynamic and varies from key to key and data to data so called dynamic. After applying desired block management condition, each Key block gets a KBDV. In next step 3, KBDV against (K_1, K_2, K_3.......... K_{10}) follows the following mathematical formula Eq. (3) to get a secret (probabilistic random) number. This random digit is referred as Data Block Average Value (DBAV). The round function is applied on this random DBAV to get a fraction free number. Data (Plain Text) will be blocked according to randomly calculated DBAV, e.g., if DBAV = 120 and let suppose, minimum

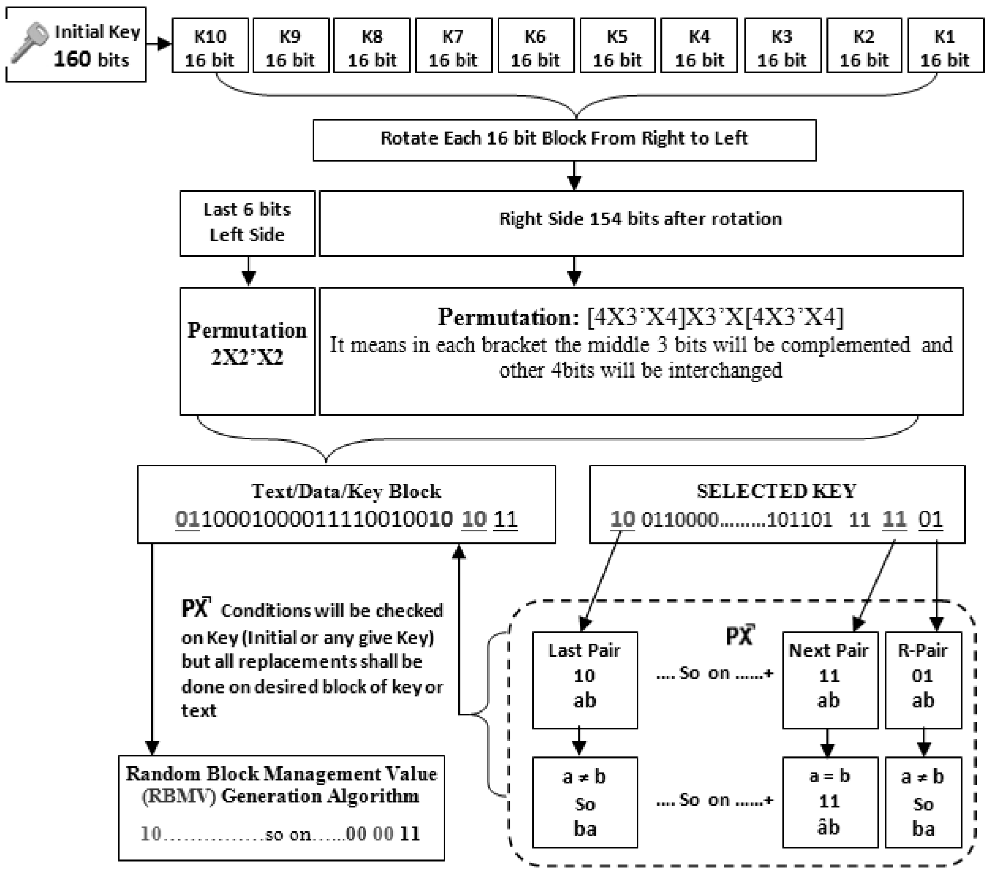

Initial Key 160 bits
K10 16 bit
K9 16 bit
K8 16 bit
K7 16 bit
K6 16 bit
K5 16 bit
K4 16 bit
K3 16 bit
K2 16 bit
K1 16 bit
Rotate Each 16 bit Block From Right to Left
Last 6 bits Left Side
Right Side 154 bits after rotation
Permutation 2X2'X2
Permutation: [4X3'X4]X3'X[4X3'X4]
It means in each bracket the middle 3 bits will be complemented and other 4bits will be interchanged
Text/Data/Key Block
01100010000111100100101011
SELECTED KEY
10 0110000………101101 11 11 01
PX̂ Conditions will be checked on Key (Initial or any give Key) but all replacements shall be done on desired block of key or text
PX̂
Last Pair 10 ab
.... So on ......+
Next Pair 11 ab
R-Pair 01 ab
a ≠ b So ba
a = b 11 âb
a ≠ b So ba
Random Block Management Value (RBMV) Generation Algorithm
10……………so on……00 00 11

Fig. 3: Random block management value generation algorithm

data is 2292 bits, then 2292/120 = 19.1. This means data will be converted into 20 blocks in which block 1 to 19 will contain 120 bits. So 120*19 = 2280 bits therefore, 2292-2280 = 12 bits so 20th block will contains 12 bits rather than 120 bits. To cope small number of bits of last block a Last Block Management Algorithm (LBMA) is applied as discussed in below section. Before to apply LBMA, there is need to generate random block management value through the employment of Random Block Management Value (RBMV) algorithm.

Random block management value algorithm: In dynamic data block generation procedure, the last data block may be smaller in number of bits as compared to the all other data blocks. Usually, the small sized data-block is considered as a weak in security strength of block ciphers. Therefore, in order to make its bit length equal to the other data blocks, we proposed a Random Block Management Value (RBMV) algorithm. This algorithm generates a random 160 bit value from initial key that will be used to manage the last data block size as depicted in Fig. 3. In this algorithm, permutations and $P\hat{X}$ operation are used. This $P\hat{X}$ operation is specifically designed for proposed Dynamic Data Blocking Mechanism. In this function ($P\hat{X}$), back tracking is not possible without knowing the original key string. This function is an alternative to XOR function. The utilization of $P\hat{X}$ is mutually binded with RBMV algorithm as depicted in Fig. 3 for further discussion. The RBMV generation algorithm's working flow is illustrated in Fig. 3. Firstly, the 160 bits initial key is blocked into 10 equal blocks (K_1, K_2, K_3,……, K_{10}) each having 16 bit length. In the next step, each 16 bits block is rotated from right to left. E.g., B1 = 10110 after rotation will be B1 = 01101. After rotation the all 10 blocks are considered as single block. In the next step, this single key block (160 bits) is converted into 2 blocks in which the first block contains 154 bits and 2nd block contains 6 bits. The permutation operation: [4*X*3*'X*4] *X*3*'X* [4*X*3*'X*4]. It means in each bracket value middle 3 bits will be complemented and other 4 bits will be interchanged. The permutation operation: [4*X*3*'X*4] *X*3*'X* [4*X*3*'X*4] can be computed as follows:

Suppose B1 = 0011011101010110001101101

In [4X3'X4] X3'X [4X3'X4] format it will become:

B1 = [0011X011X1010] X101X [1000X110X1101]

The 3' (Complement of middle 3 bits in red color) will be applied as follows:

B1 = [0011X100X1010] X010X [1000X001X1101]

The 4 underlined bits in each bracket will be interchanged to each other as follows:

B1 = [1010X100X0011] X010X [1101X001X1000]
Result (B1) = 1010100001101011010011000

Similarly, Permutation operation [2X2'X2] can be applied on 6 bit left side 2nd block. After applying these both kinds of permutation operations, the $P\hat{X}$ operation is applied as follows:

Suppose, Text Block = 01 10 10 11 and
Selected Key = 10 11 11 01

Before applying $P\hat{X}$ operation each Text Block and selected Key shall be converted into pairs of two binary bits. If any of block contains odd number of bits then the right most bit of selected key will be concatenated:

$$\text{DBAV} = \text{Round}\left[\sum_{i=1}^{n} K_i\,(\text{KBDV}) / 10\right] \tag{3}$$

to the left of any odd number bits block either text or key. After completing this procedure of pair conversion, the $P\hat{X}$ conditions (Fig. 3) shall be checked on selected key but all replacements shall be done on text block. Firstly, the $P\hat{X}$ condition will be checked on first pair of selected key and replacement will be done on the first pair of text block, then $P\hat{X}$ condition will be checked on 2nd pair of key but replacement will be done of 2nd pair of text block and so on up to last pair:

Selected Key bits = *10* 11 *11* 01

In this block in red color pair left side digit is "a" and right side digit is called b, similarly, in each blue pair left side digit is "a" and right side digit is called b. In the same way a and b values are assigned against each pair of text block. According to $P\hat{X}$ condition in each pair if a = b in selected key pair it will result $\hat{a}b$ on text block and if a ≠ b in key pair it will result ba on text block. These notations can be written as follows:

$P\hat{X}$(ab) → if a = b in Key : $\hat{a}b$ on Text Block
$P\hat{X}$(ab) → if a ≠ b in Key : ba on Text Block

where, $\hat{a}$ means complement of "a". After applying $P\hat{X}$ operation the text block can be written as follows:

$P\hat{X}$(Text Block) = PTB = *10* 00 *00* 11

In order to re-compute the original value from PTB = *10* 00 *00* 11, there is need to apply the same $P\hat{X}$ conditions on Selected Key = *10* 11 *11* 01 but all replacements shall be done on PTB. After applying the $P\hat{X}$ conditions the following original value is generated from PTB = *10* 00 *00* 11

$P\hat{X}$(PTB) = 01 10 10 11 = Text Block

This $P\hat{X}$ (PTB) is equla to the original value of Text Block. Hence, the $P\hat{X}$ operation can be recomputed with 100% accoracy.

Last block management algorithm: After generating the random block management value through RBMV algorithm, the next step is to adjust this random value in the last data block that contains small number of bits. This task is necessary in order to make the last data block size (bits) equal to the all other blocks. In cryptographic algorithms the small sized block possesses greater chance to crack rather to the bulky block having more number of bits. The recommend block size is 128 bits or more. The proposed Dynamic Data Blocking Mechanism (DDBM) recommends that block size should be between 128 to 256 bits for getting optimal security. However DDBM has the option to generate block size in between 80 to 128 bits but this block range is for small devices. For managing the size of last block this study proposes a Last Block Management Algorithm which is as follows.

Last block management algorithm:

```
Declare B, NLB, R2, LB as string and all having initial value = " "
LDBBV = 010010010110 // Supposed last block's bits
B = Calculate_Bits (LDBBV);
IF B<80 bits AND First-Last bit (RBMV) = Same // For Block size (128-256), 80 bits will be replaced by 128
{
NLB = Select "LEFT Side" 128 bits (from left side) of RBMV;
}
Else IF B<80 bits AND First-Last bit (RBMV) = Different // For Block size (128-256), 80 bits will be replaced by 128
{
NLB = Select "FIRST RIGHT Side" 128 bits (from right side) of RBMV;
}
NLB = Latest NLB value;
LB = Select DBAV bits from right side (NLB || LDBBV); // DBAV means its decimal value
End if;
End if;
IF B>80 bits AND<DBAV AND Not Multiple of 8 // For Block size (128-256), 80 bits will be replaced by 128
{
R2 = [Select only left side of decimal point value of (LDBBV bits /8) + 1] * 8;
LB = Select first R2 no. of bits from LB starting from right to left;
}
Else
Do nothing;
```

```
LB = Select Right Side DBAV No. bits From (LB);
LB = latest LB value;
End if;
```

At receiver's end the dynamic data blocking procedure will be done in the same way as it has done at sender's side. However, the last block management algorithm will automatically be omitted at receiver's end because the cipher text has already gone through the blocking phase therefore, it will satisfy the block size conditions. The bit counts value of last data block will be calculated at sender' side before the implementation of last block management algorithm. This block count will be sent with key to other party. After completing the decryption phase the string will be compared with last block count's value. The string up to last block count will be preserved and other extra string will be discarded.

RESULTS AND EVALUATION

In order to evaluate the working accuracy of proposed dynamic data blocking mechanism, we applied all of its steps (algorithms) upon a supposed encryption key and plain text data. All proposed algorithms have been tested through digital logic and mathematical proofs. We utilized the same evaluation approach which has been followed in the book of digital logic and design. The mathematical evaluation is as follows.

Initial 160 bit key in string form: *ijazshoukat@pakistan*

Initial key 160 bit key in binary form:

[Leftside]0110100101101010011000010111101001110 0110110100001101111011101010110101101100001011101000100000001110000011000010110101101101001011100110111010001100001011011110 [Right Side]

According to the proposed dynamic data blocking mechanism, the initial key can be partitioned in to 10 blocks having 16 bits each by applying this mathematical formula:

Initial Key (K) = 160 bits: 160/10 = 16 bits of each sub key.

After partitioning the master 160 bits key, we got the following sub key blocks:

Right Side Key Block-1
(R-K1): 0110000101101110
K2: 0111001101110100
K3: 0110101101101001
K4: 0111000001100001
K5: 0111010001000000
K6: 0110101101100001
K7: 0110111101110101
K8: 0111001101101000
K9: 0110000101111010
Left Side Key Block- 10
(L-K10): 0110100101101010

Similarly, we choose a supposed plain text data which is as follow.

Plain text data in string form: In the name of Allah the most beneficent the merciful. There is not God but Allah, Muhammad is the Prophet of Allah. This data is going to be implemented with newly developed symmetric encryption algorithm named as Random Encryption Method (REM). This data will be encrypt and decrypt with REM practically.

Plain text data in binary form (2292 bits):
{Left Side} 01001001 01101110 0001 01110100 01101000 01100101 0001 01101110 01100001 01101101 01100101 0001 01101111 01100110 0001 01000001 01101100 01101100 01100001 01101000 0001 01 110100 01101000 01100101 0001 01101101 01101111 01110011 01110100 0001 01100010 01100101 01101110 01100101 01100110 01101001 01100011 01100101 01101110 01110100 0001 01110100 01101000 01100101 0001 01101101 01100101 01110010 01100011 01101001 01100110 01110101 01101100 00101110 0001 01010100 01101000 01100101 01110010 01100101 0001 01101001 01110011 0001 01101110 01101111 01110100 0001 01000111 01101111 01100100 0001 01100010 01110101 01110100 0001 01000001 01101100 01101100 01100001 01101000 00101100 0001 01001101 01110101 01101000 01100001 01101101 01101101 01100001 01100100 0001 01101001 01110011 0001 01110100 01101000 01100101 0001 01010000 01110010 01101111 01110000 01101000 01100101 01110100 0001 01101111 01100110 0001 01000001 01101100 01101100 01100001 01101000 00101110 0001 01010100 01101000 01101001 01110011 0001 01100100 01100001 01110100 01100001 0001 01101001 01110011 0001 01100111 01101111 01101001 01101110 01100111 0001 01110100 01101111 0001 01100010 01100101 0001 01101001 01101101 01110000 01101100 01100101 01101101 01100101 01101110 01110100 01100101 01100100 0001 01110111 01101001 01110100 01101000 0001 01101111 01110101 01110010 0001 01101110 01100101 01110111 01101100 01111001 0001 01100100 01100101 01110110 01100101 0110110001101111 01110000 01100101 01100100 0001 01110011 01111001 01101101 01101101 0110010101110100 01110010 01101001 01100011 0001 01100101 01101110 01100011 01110010 01111001 01110000 01110100 01101001 01101111 01101110 0001 01100001 01101100 01100111

Table 1: Logical key blocking mechanism

Sub key block #	Blocks (right to left)	Key Block Decimal Value (KBDV)
K1	0110000101101110	24942>24576 so 24942/232 = 107.50 = 107
K2	0111001101110100	7374>4096 and <16384 so 7374/64 = 115.21 = 115
K3	0110101101101001	27497>24576 so 27497/232 = 118.52 =118
K4	0111000001100001	7061>4096 and <16384 so 7061/64 = 110.32 = 110
K5	0111010001000000	29760>24576 so 29760/232 = 128.27 = 128
K6	0110101101100001	27489>24576 so 27489/232 = 118.48 = 118
K7	0110111101110101	28533>24576 so 28533/232 = 122.98 = 123
K8	0111001101101000	29544>24576 so 29544/232 = 127.34 = 127
K9	0110000101111010	24954>24576 so 24954/232 = 107.56 = 107
K10	0110100101101010	15282>4096 and <16384 so 15282/64 = 238.78 = 239>128 and <256 so 239/2 = 119.5 = 119
AVG	Data Block Avg. Value (DBAV)	1172/10 117.2 = 117 For getting each block as multiple of 8 this rule will be followed: 117/8 = 14.62 = 15 and again 15*8 = 120 It means each data block will contains 120 bits. This value is dynamic. IF DBAV<80 then DBAV = DBAV + 8 (this condition is not for block sizes of 128 to 256 bits)

01101111 01110010 01101001 01110100 01101000
01101101 0001 01101110 01100001 01101101
01100101 01100100 0001 01100001 01110011 0001
01010010 01100001 01101110 01100100 01101111
01101101 0001 01000101 01101110 01100011
01110010 01111001 01110000 01110100 01101001
01101111 01101110 0001 01001101 01100101
01110100 01101000 01101111 01100100 0001
00101000 01010010 01000101 01001101 00101001
00101110 0001 01010111 01100101 0001 01110111
01101001 01101100 01101100 0001 01110000
01110010 01100001 01100011 01110100 01101001
01100011 01100001 01101100 01101100 01111001 00
01 0100010101101110 0110001101110010
0111100101110000 01110100 01100101 / 01100100
0001 0110000101101110 01100100 000101000100
0110010101100011 01110010 01111001 0111000 0
01110100 0001011101000 11010000110 /
1001011100110001011001000110000101110100011 00
0010001011101110110100101110100011010000000101
0100100100010101001101001011 10 {Right side}

Implementation of logical key blocking mechanism: The mathematical transformation Eq. (3) with logical key blocking mechanism as depicted in Fig. 2, can be applied and calculated as described in Table 1. If Data (Plain Text Message) will be less than 2048 bits then it will be converted to 2048 bits at least before staring encryption. It is known that, 1 charter = 8 bits in binary representation. So approximately 256 char message is needed. As 256*8 = 2048 bits. For every message the chars will be count, if chars <256 then Message-Chars/256 = R (Result) and same message is repeated by R times after putting "~" symbol at the left end of original message. For minimizing the risk of known symbol, there is another solution for this problem that is in case of not putting any symbol, the message (Plain text Data) binary bits can be calculated before converting it into 2048 bits. This binary counter can easily be exchanged with symmetric key. In 160 bits key, the number of 1 and 0 cannot be repeated 5 times continuously. It means 0 or 1 can repeat 4 times but not 5 times continuously. By following this pre condition, the KBDV column cannot be less than 73 decimal value. Moreover, minimum Key Block Decimal Value (KBDV) will not be less than 127 bits and maximum KBDV will not be greater than 16260 (if all last 7 bits of each key block will be on). In case of odd number of plain text binary bits, the number of blocks will follow the rule as exampled here, e.g., if minimum data is 2292 bits, Then 2292/120 = 19.1 it means data will be converted in to 20 blocks in which block 1 to 19 will contain 120 bits. So 120*19 = 2280 bits therefore, 2292 - 2280 = 12 bits so 20th block will contains 12 bits rather to 120 bits. The recommended data block size is 128 to 256 bits. However, there is a choice of 80 to 128 bits block size too. These block sizes are dependent on the conditions discussed in Eq. (1) and (2). Here, in this practical proof, the conditions of Eq. (1) has been applied on column 3 of Table 1 which means the data block size will remain between 80 to 128 bits long. According to Table 1, in the last left side 20th block contains only 12 bits.

Left Side Last Data Block Binary Value (LDBBV): 010010010110.

In order to make this LDBBV equal to 120 bits as all other 19 data blocks are, there is need of random bits. These random bits can be derived from initial 160 bits key by applying Random Block Management Value (RBMV) Algorithm as discussed in Table 2.

Implementation of Random Block Management Value (RBMV) algorithm: RBMV requires the following three operations on the initial 160 bit key in order to get the probabilistic random string that can be used to manage last data block:

- **Rotation:** Rotate each 16 bit block (R→L) (Table 3)
- **Permutations operations:** [4X3'X4] and [2X2'X2]
- **PX operation:** It means select pairs of string (s) from right to left if pair are different just shift rotate if pair is same then select the 2nd bit of this pair and replace it with the complement of first bit of same pair

Table 2: Random Block Management Value (RBMV) algorithm with implementation

4X4 conversion of permutated 160 bit key	After applying PX operation	Random Block Management Value (RBMV)
[Right side] 1101	0110	RBMV =
0111	1001	[Left side]
1011	0101	1001101010010110100110101010011010010101
1000	0110	100101101010010110100
0001	1010	1101001011010100110101001101010100110010
0011	1001	1100101010110100110101010100101101010011
0001	1010	0100110010110010110 [right side]
1010	0101	
0001	1010	
0001	1010	
1101	0110	
0101	1010	
1011	0101	
1110	0101	
1101	0110	
0111	1001	
0011	1001	
0001	1010	
1100	0110	
0001	1010	
1001	0110	
0000	1010	
1001	0110	
0110	1001	
1101	0110	
0001	1010	
1011	0101	
0101	1010	
1101	0110	
0011	1001	
1111	0101	
0010	1001	
1001	0110	
0001	1010	
0001	1010	
0010	1001	
1001	0110	
0010	1001	
0101	1010	
[Left side] 0110	1001	

Table 3: Rotation operation implementations

160 bits initial key blocks	After applying rotation (R→L)	160 bits rotated key
R-K1: 0110000101101110	Rotate (R→L)	R-K1: 0111011010000110
K2: 0111001101110100	Rotate (R→L)	K2: 0010111011001110
K3: 0110101101101001	Rotate (R→L)	K3: 1001011011010110
K4: 0111000001100001	Rotate (R→L)	K4: 1000011000001110
K5: 0111010001000000	Rotate (R→L)	K5: 0000001000101110
K6: 0110101101100001	Rotate (R→L)	K6: 1000011011010110
K7: 0110111101110101	Rotate (R→L)	K7: 1010111011110110
K8: 0111001101101000	Rotate (R→L)	K8: 0001011011001110
K9: 0110000101111010	Rotate (R→L)	K9: 0101111010000110
L-K10: 0110100101101010	Rotate (R→L)	L-K10: 0101011010010110

Permutation operation: In next step, the permutation operation [4X3'X4] X3'X [4X3'X4] is applied on 160 bits rotated key. However, for last 6 bits in the left part of key at left side the permutation operation [2X2'X2] is implemented. The operation [4X3'X4] X3'X [4X3'X4] means, select first 4 bits from right side of key (string) as it is, then take the complement of next 3 bits and then again select the next 4 bits as it is, then exchange both left and right side 4 bits with each other and so on. Moreover, the middle 3 bits of large bracket pairs will just be complemented in whole string. However, at the last, the left most 6 bits of the left part of rotated key are permutated by using [2X2'X2] permutation operation which means select 2 bits of key as it is then take the complement of next 2 bits and then again select the 2 bits as it is then exchange both 2 bits

Table 4: Right side, half part of 160 bits rotated key after applying permutations (permutation operation part a)

000	1000	101	1101	000	0110	000	0111	010	0101	101	1010	110	0010	111	0110	011	1001	110	1101	000	0110
╳				╳				╳				╳				╳				╳	
111	0000	**010**	0110	**111**	1101	**111**	0101	**101**	0111	**010**	0010	**001**	1010	**000**	1001	**100**	0110	**001**	0110	**111**	1101

Left side, half part of 160 bits rotated key after applying permutations (permutation operation part b)

01	01	01	101	0010	110	0101	111	0100	001	1000	010	1101	100	1110	101	0111	011	1101	101	0000	110	1101	011	0000
	╳				╳				╳				╳				╳				╳			
01	**10**	01	**010**	0101	**001**	0010	**000**	1000	**110**	0100	**101**	1110	**011**	1101	**010**	1101	**100**	0111	**010**	1101	**001**	0000	**100**	1000

Table 5: Last block management algorithm with implementation

Last block management algorithm	Practical implementation
Declare B, NLB, R2, LB as string and all having initial value = “”	LDBBV = 010010010110;

LDBBV = 010010010110
B = Calculate_Bits (LDBBV);
	B = 12; // this value will indicate the length of last data block at the time of decryption and

IF B<80 bits AND first-last bit (RBMV) = same // for block size (128-256), 80 bits will be replaced by 128 {	So B<80 and first-last bit = same therefore,

NLB = select “LEFT side” 128 bits (from left side) of RBMV;
}
Else IF B<80 bits AND first-last bit (RBMV) = different // for block size (128-256), 80 bits will be replaced by 128
{
NLB = select “FIRST RIGHT Side” 128 bits (from right side) of RBMV;
}
NLB = latest NLB value;
	NLB = [left side]
	100110101001011010011010101001101001010110010110101001011010011010010110100
	11010010110101001101010011010101001100101100101010110100110101
	101010 [right side]

LB = select DBAV bits from right side (NLB || LDBBV); // DBAV means its decimal value
End if;
End if;
IF B>80 bits AND<DBAV AND not multiple of 8 // for block size (128-256), 80 bits will be replaced by 128
{
R2 = [select only left side of decimal point value of (LDBBV bits / 8) + 1] * 8;
	NLB = NLB;
	DBAV = 20; // so right side 120 bits are selected from NLB
	LB =
	[left side]
	10010110100110101010011010010101100101101010010110100110100 10
	11010100110101001101010100110010110010101011010011010101010 ||
	010010010110 [right side]

LB = Select first R2 no. of bits from LB starting from right to left;
}
Else
Do nothing;
LB = Select right side DBAV No. bits from (LB);
LB = latest LB value;
End if;
	LB =
	[left side]
	1001011010011010101001101001010110010110101001011010011010010
	11010100110101001101010100110010110010101011010011010101010
	010010010110 [right side]
	From above 132 bits long LB string right most 120 bits are selected because DBAV = 120 bits lengthy.
	LB = [left side]
	1010101001101001010110010110101001011010011010010110101001101
	01001101010100110010110010101011010011010101010 010010010110
	[right side]

pair with each other. The permutation operations can be applied as Table 4.

The ultimate objective this kind of permutation is to produce uniform dynamic modification in the key that will further be mixed through the utilization of $P\hat{X}$ operation. It is well known that the change of bit in key can produced enough complexity for the cracker. Our reflect with the opinion of authors who claim, if key state is updated on some random bases then it can be resulted as quiet hard for adversary (attacker) to get useful secret information (Kocher *et al*., 2011). In our case many bits of key have been dynamic modified without having the knowledge of their identity (0 or 1) and the remaining bits are permutated. After applying the rotation and permutation operations, the permutated key becomes:

[Leftside]01100101001010010010000100011001001011110011110101011011000111010110100100001001000111000001001101111101111010110101110100010001101000001001100011000101101111101 [Right Side]

Now, in next step the P$\hat{X}$ operation is applied that requires 4 by 4 bits conversion which is applied in Table 2. In next step, the Last Block Management Algorithm (LBMA) is applied and evaluated mathematically as discussed in Table 5.

Hence, the Last Block Binary Value (LDBBV): 010010010110 which was 12 bits long before the employment of last Block Management value (RBMV) algorithm now has become 120 bits long as the all other 19 data blocks are. The proposed dynamic data blocking mechanism can generated dynamic number of blocks with dynamic number of bits as follows:

```
R-DB1:
[Left Side]
1001011100110001011001000110000101110100011000
0010001011101110110100101110100011010000000101
0100100100010101001101001011110 [Right Side]
DB2:
[Left Side]
0110010000001011000010110111001100100000101000
1000110010101100011011100100111100101110000001
1101000000101110100011010000110 [Right Side]
.
+
.
.
+
.
L-DB20:
[Left Side]
1010101001101001010110010110101001011010011010
0010110101001101010011010101001100101100101011
0110100110101010100100100101110 [Right Side]
```

The original value counter of last data block will exchange with private key and after decrypting the data only the right most value of last data block will be considered as a plain text. In the light of mathematical evaluation, our proposed Dynamic Data Blocking Mechanism (DDBM) with its sub algorithms gives 100% correct results. This Dynamic data blocking mechanism requires to perform on both sides (sender and receiver) with the same logic.

DISCUSSION AND ANALYSIS

The proposed Dynamic Data Blocking Mechanism (DDBM) is substantially significant to fixed data blocking mechanism. Fixed length blocks support the cracker in matching of hypothetical decrypted string (block) with the targeted chosen plaintext string to get accurate final results. The accuracy of final result is actually dependent on matching of hypothetical decrypted string with any chosen string of plaintext which is easy in case of known fixed length data blocks. Fixing block length can help the cracker in mapping/matching step of hypothetical decrypted text with known (chosen) block of plaintext. The other noteworthy problem with fixed and known length block is the brute force attacking approximation because any chosen fixed sized block means that the same sized key is implemented on it. Therefore, in this case, the brute force attack is more likely and effectively to be applicable as discussed in Table 6. Moreover, Permutation is less effective in case of fixed substitution and it just means to increase processing computation ability which can easily be defeated in current era's based high computing processors. Fixed parameters trigger the applicability of modern cryptanalysis attacks on symmetric cryptosystems by offering computable probability range because the permutations happed against fixed sized data blocks is less vigorous against the larger probability of differential attacks as witnessed in studies (Lu, 2010; Lu, 2008). This cause is reasonably notable in case of AES and DES because both utilize fixed (static) data blocking mechanism (Young and Yang, 2010) (Singh and Bansal, 2010). Proposed scheme is fully able to create dynamic data blocks. This is more convincing that DES utilized fixed data blocks and fixed 8 bits substitution and practically cracked in just 4.5 days (Wobst, 2001) even the security satisfaction of Triple DES (TDES) is also vulnerable as its practical cracking requires 800 days on a machine that can execute 50 billion keys per second (Alanazi *et al*., 2010). Moreover, Biryukov *et al*. (2010) reported that, TDES has already been cracked with key complexity of 2^{56}

Table 6: Comparison of randomness between dynamic and static data blocking

Parameters	With prior fixed data blocking schemes (DES, AES-128)	With proposed DDBM scheme
Probabilistic randomness calculation	Suppose, plain text data (Đ) = 2048 bits. Blocking with AES-128 = 2048/128 = 16 blocks, so No. of data blocks (β) = 16 No. of bits of each block (μ): 128 bits Probabilistic randomness calculation formula: $\beta * 2^{\mu \text{ bits}}$ where For each single block = $1 * 2^{\mu \text{ bits}}$ If: β is fixed = 16 in supposed case μ is fixed = 128 in supposed case So, $16 * 2^{128 \text{ bits}}$ Therefore:- It is static and easily computable	Suppose plain text data (Đ) is, Đ = 2048 bits No. of data blocks (will be done randomly) = ₱ (β) No. of bits of each block (will be selected Randomly) = ₱ (μ) bits Probabilistic randomness calculation formula: $₱(\beta) * 2^{₱(\mu) \text{ bits}}$ Where ₱ (β) is random ₱ (μ) is random As these are unknown to the cracker. So it is quite confusing for the cracker to guess it. Therefore: It is dynamic and not easy to compute
Result	Weak and guessable	Robust and hard to guess
NP-hard property for block partitioning	×	√ (see discussion section debate on P and NP)

bits as claimed by Biryukov *et al*. (2010). Similarly, AES utilizes fixed substitution and fixed data blocks and has been victim under chosen key distinguisher attack which was verified by the authors of study (Biryukov *et al*., 2009) with almost applied complexity of ($q.2^{67}$) queries to crack full 14 rounds of AES-256 which works for 1 key out of 2^{35} keys with 2^{120} data and time complexity trials. Moreover, these threats possibly becomes practical under some modes of operations that take chosen inputs block as a key (Biryukov and Dunkelman *et al*., 2010). This is commonly known that fixed parameters in creation of portability and randomness are not considered robust that's why proposed approach aims to create dynamic data blocks rather to fixed data blocks. In Table 6 and in below section, we have discussed how, fixed data blocking cannot create sufficient probability and leads the situation to non-NP Completeness.

Thus, fixed data blocking approach did not create sufficient randomness (Table 6) and is not adequately secure against brute force attack. Our opinion quite reflects the prior research analysis of different authors. In Ritter (1995), has been reported that Dynamic (Variable) Data blocking is architecturally simple but computationally stronger and faster than fixed data blocking. Dynamic data blocking facilitates vigorous pseudorandom-permutation that is the superlative principle in designing of block ciphers (Cook, 2006). Dynamicity can create sufficient randomness. Randomness and mathematical robustness are good heuristic to resist cryptanalysis attacks (Cook, 2006).

Discussion on P, NP and (P ≠ NP): P class belongs to (∈) set of deterministic (not-harder) based all decision problems (yes or no) which can be solved in polynomial time (O (n^k) where n is problem size and k is constant. In other words for P class problems, efficient algorithms persist to solve desired problem practically by using deterministic Turing machine. NP class ∈ set of non-deterministic (hard) natured all optimization problems (series of possible guesses) which require infinite time to predict possible solution so called hard (non-deterministic). In other words the problem belongs to NP Class has not an efficient algorithm with practical solution. In class of *P*, mostly a solution is find out and then it required to verify against NP within time span *T*. where, the Polynomial Time (*T*) term is used for an efficient algorithm which can solve any problem quickly in reasonable time. For hard problems, it is broadly believed ($P \neq NP$) but it has not been proven yet (Gomes and Williams, 2005; Taslaman, 2012).

For this widely believed hypothesis ($P \neq NP$ - widely believed for which no efficient algorithm exists), it is assumed that the solution against the related problem is intractable which means, there persist no efficient algorithm that guarantees an optimal solution for such hard problems as witnessed in a book (Gomes and Williams, 2005). Therefore, generally but unproved hypothesis ($P \neq NP$) has no known polynomial algorithm to solve the NP-Complete problem optimally and this hypothesis is famously known as hard problem ($P \neq NP$) as agreed by Taslaman (2012). NP-Complete problems require too bulky set of tries to guess the solution of a problem which further required large optimizations (serious of guess) to verify the exact solution in polynomial Time (*T*). Therefore, problem is said to be NP-Hard if all the supplementary problems in NP class can be reduce in time T which means if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$ satisfy the reduction properties through functions *f* and *h*. These functions (f_{xz}, h_{xy}) and (f_{yz}, h_{YZ}) can be reduced as X to Y and Y to Z respectively through a reduction composition $f_{Yz}.f_{xy}$ maps the instance of *X* to an instance of *Z* and similarly $h_{xy}.h_{yz}$ can track the solution of *Z* back to *X* which satisfies, problem *X* is NP-Complete and any other problem like *X* will also be *NP* complete (Dasgupta *et al*., 2008). This reduction can be represented with a notation $X \leq_p Z$ (Polynomial reducible to Z). Mathematically, it can be written as:
A problem X is NP-complete if:

- $X \in NP$
- $X \leq_p Z$ for every $X \in NP$: polynomial reducible from $X \rightarrow Z$ Similarly
- $Z \leq_p X$ for every $Z \in NP$: polynomial reducible from $Z \rightarrow X$

This proof satisfy the properties of NP-Completeness, i.e., *P* is NP-hard (polynomial reducible) and $P \in NP$ in accordance with general *P* and *NP* completeness transformation.

By taking the probabilistic randomness computation example of Table 6 against AES and proposed scheme, we can write the probabilistic computation complexity equation as follows:

$$₱ = \beta * 2^{\mu \, bits} \tag{4}$$

where,
₱: Probability computational complexity
β: The no. of blocks
μ: The bit-length of each block

Lemma: we prove NP-Completeness of probabilistic randomness equation ₱ = β * $2^{\mu(\text{bits})}$ in case of AES and proposed case of DDBM by supposing Đ is known in both cases. The purpose of declaring Đ as known is only to evaluate the robustness of probabilistic randomness complexity for which generally at least one parameter should be known in both cases for comparison.

First case: blocking partitioning under AES-128: In this case: $\mu \notin NP$ as:- μ is known (as discussed in Table 6) which means it is deterministic because it is fixed (128 bit).

Where $\notin$ means "not belongs to" and $\Rightarrow$ means "implies that".

Similarly, we can compute $\beta = Đ. \mu^{-1}$

Therefore, $\beta \notin NP$:- as β can easily be computed through simple function ($\beta = Đ. \mu^{-1}$) having both known parameters (Đ , μ) $\Rightarrow \beta$ is deterministic.

So $\beta \in P$ and $\mu \in P$:- as P is a deterministic class

According to NP-Complete transformation as discussed earlier, the Eq. (4) is hard if-and-only-if: $\beta \in NP$ and $\mu \in NP$ which is false here. It means Ᵽ = $\beta * 2^{\mu \text{ bits}}$ is deterministic and is not a hard problem against NP-Completeness.

Second case: block partitioning under proposed DDBM: $\mu \in NP$ as:- μ follows a series of possible guesses (Table 6) which means it belongs to optimization problem class (NP) having non-deterministic series of instances of $\mu = \mu_1, \mu_2, \mu_3 \ldots\ldots \mu_n$ ($\mu_1, \mu_2, \mu_3 \ldots\ldots \mu_n$) $\subset \mu \in NP \Rightarrow \mu_n \in NP$: as μ_n is a subset ($\subset$) of μ

Similarly, $\beta \in NP$ as:- β follows a series of possible guesses which means it belongs to optimization problem class (NP) having non-deterministic series of instances of $\beta = \beta_1, \beta_2, \beta_3 \ldots\ldots \beta_n$ ($\beta_1, \beta_2, \beta_3 \ldots\ldots \beta_n$) $\subset \beta \in NP \Rightarrow \beta_n \in NP$: as β_n is a subset ($\subset$) of β.

Now let we try to compute:

$$\beta = Đ. \mu^{-1} \tag{5}$$

In this computation we only know Đ but we do not know β and μ which both parameters have non-deterministic property as both $\in$ NP, even its subsets (sub-instances) also $\in$ NP $\Rightarrow$ Eq. (5)$\in$ NP.

Therefore,

Eq. (5) $\leq_p$ Eq. (4):- as instances of Eq. (5) $\in NP$
Eq. (4) $\leq_p$ Eq. (5):- as instances of Eq. (4) $\in NP$

Where Eq. (5) $\subset$ Eq. (4) $\Rightarrow$ Eq. (4) $\in NP$ which links to optimization problem (series of possible guesses) that are hard to guess which means no polynomial efficient algorithm persists which gives optimal or guaranteed solution for such problems belong to NP. This hypothesis is widely believed as hard problem ($P \neq NP$) because no efficient algorithm exists to find practical solution for all problems which satisfy this hypothesis as agreed by the authors of studies (Gomes and Williams, 2005; Taslaman, 2012).

In our proposed scheme the diversity of both parameters (block numbers, block bits) is dependent on a plaintext data and key. Proposed DDBM scheme is enriched with dynamicity to create sufficient randomness as for one set of plaintext data and key {$Đ_1$, $Ꝁ_1$} the block length (β) will be β_x but all data blocks will contain equal bits and β_x will not be known to the cracker. Similarly, for other set of data and key {$Đ_2$, $Ꝁ_2$} the block length (β) will be β_y but all data blocks will contain equal bits and β_y will not be known to the cracker. In both cases the block length ($\beta_{(x,\ y)}$) and number of blocks will vary from key to key and data to data but for cracker these both variables will be dynamic and unknown. The comparison of randomness computation of proposed scheme with prior methods used in DES and AES as explained in above section and Table 6, which clearly appeal that proposed method of data blocking is robust rather to prior schemes. The complexity and probability of proposed method is computationally hard ($P \neq NP$) as discussed in above section and Table 6. On the other hand in case of prior cryptographic methods (DES, AES) the same computation is mathematically computable and un-adequately follows the condition ($P \neq NP$). Hence, the objective of creating dynamic data blocking in any cryptographic is superior to fixed data blocking mechanism.

CONCLUSION

Dynamic data blocking is a superlative philosophy to stimulate probabilistic randomness in encryption heuristics rather to employ any sort of static (fixed) data partitioning policy. Fixed data blocking is an actual cause of academic cracking of AES and practical cracking of DES because fixed block partitioning approach supports the cracker in *matching* of hypothetical decrypted string (block) with the targeted chosen plaintext string to get accurate final results. Static (fixed) data blocking mechanism generates weak randomized probability against exhaustive searching and computationally weak to follow the condition (*$P \neq NP$* - widely believed for which no efficient algorithm exists) in block partitioning step as discussed in above section and Table 6, however, in contrast with fixed data blocking, our proposed Dynamic Data Blocking Mechanism (DDBM) is architecturally simple but computationally robust in probabilistic randomness to offer sufficient resistance with series of large optimizations which is adequately robust to provide initial safe-guard against powerful cryptanalysis attacks. Proposed idea is enriched with dynamicity that triggers more probabilistic randomness. Moreover, in our proposed scheme (DDBM) the diversity of both parameters (block numbers, block bits) is dependent on a plaintext data and key. According to our best knowledge our idea is novel and timely significant to enhance the security of symmetric cryptosystems. Moreover, it is enriched with robust scientific contribution to replace the fixed data blocking mechanism with dynamic tactics in future cryptosystems to achieve higher degree of security satisfaction.

REFERENCES

Alanazi, O.H., A.A. Zaidan, H.A. Jalab, M. Shabbir and Y. Al-Nabhani, 2010. New comparative study between DES, 3DES and AES within nine factors. J. Comput., 2(3): 152-157.

Biryukov, A., 2004. Boomerang attack on 5 and 6-round AES. Proceeding of the 4th Conference on Advanced Encryption Standard.

Biryukov, A. and D. Khovratovich, 2009. Related-key cryptanalysis of the full AES-192 and AES-256. In: M. Matsui (Ed.), ASIACRYPT 2009. Lecture Notes in Computer Science, Springer, Heidelberg, Vol. 5912: 1-18.

Biryukov, A., D. Khovratovich and I. Nikolic, 2009. Distinguisher and Related-key Attack on the Full AES-256 (Extended Version). In: Halevi, S. (Ed.), CRYPTO 2009. LNCS, Springer, Heidelberg, 5677: 231-249.

Biryukov, A., O. Dunkelman, N. Keller, D. Khovratovich and A. Shamir, 2010. Key Recovery Attacks of Practical Complexity on AES-256 Variants with up to 10 Rounds. In: Gilbert, H. (Ed.), EUROCRYPT 2010. Lecture Notes in Computer Science, Springer, Heidelberg, 6110: 299-319.

Cook, D.L., 2006. Elastic Block Ciphers. Ph.D. Thesis, Graduate School of Arts and Science, Columbia University.

Dasgupta, S., C.H. Papadimitriou and U.V. Vazirani, 2008. Book-Algorithms. 1st Edn., Published by McGraw Hill Book Publishers, pp: 336, ISBN-10: 0073523402.

Demirci, H., I. Taskın, M. Coban and A. Baysal, 2009. Improved Meet-in-the-middle Attacks on AES. In: B. Roy and N. Sendrier (Eds.), INDOCRYPT 2009. Springer-Verlag, Berlin, Heidelberg, LNCS, 5922: 144-156.

Elminaam, D.S. and H.M. Abdul-Kader, 2010. Evaluating the performance of symmetric encryption algorithms. Int. J. Network Secur., 10(3): 213-219.

Ferguson, N., J. Kelsey, S. Lucks, B. Schneier and M. Stay, 2001. Improved Cryptanalysis of Rijndael. In: Schneier, B. (Ed.), FSE 2000. LNCS, Springer, Heidelberg, pp: 213-230.

Gilbert, H. and M. Minier, 2000. A collision attack on 7 rounds of Rijndael. Proceeding of the 3rd AES Candidate Conference (AES3), pp: 230-241.

Gomes, C.P. and R. Williams, 2005. Approximation Algorithms. In: Burke and Kendall (Eds.), Introduction to Optimization, Decision Support and Search Methodologies, Kluwer, pp: 557-58.

Kocher, P., J. Jaffe, B. Jun and P. Rohatg, 2011. Introduction to differential power analysis. J. Cryptography Eng., 1: 5-27.

Lu, J., 2008. Cryptanalysis of block ciphers. Ph.D. Thesis, the University of London, UK (2008). A Copy is Available Online as Technical Report RHUL-MA-2008-19, Department of Mathematics, Royal Holloway and University of London, UK. Retrieved form: http://www.ma.rhul.ac.uk/static/techrep/2008/RHUL-MA-2008-19.pdf.

Lu, J., 2010. The (related-key) impossible boomerang attack and its application to the AES block cipher. Published in DES, Codes Cryptography-Springerlink.com, DOI 10.1007/s10623- 010-9421-9.

Ritter, T., 1995. Variable Size Block Ciphers. Cryptography Software Analogy Digital. Retrieved form: http://www.ciphersbyritter.com/VSBC.HTM.

Shoukat, I.A. and K.A. Bakar, 2013. Effective evaluation metrics for the assessment of cryptographic algorithms and key exchange tactics. Inform. Tokyo (Japan), 16(5): 2801-2814.

Shoukat, I.A., K.A. Bakar and M. Iftikhar, 2011. A survey about the latest trends and research issues of cryptographic elements. Int. J. Comput. Sci., 8(3): 140-149.

Shoukat, I.A., A.B. Bakar and S. Ibrahim. 2013. A generic hybrid encryption system (HES). Res. J. Appl. Sci. Eng. Technol., 5(09): 2692-2700.

Singh, A. and M. Bansal, 2010. FPGA implementation of optimized DES encryption algorithm on Spartan 3E. Int. J. Sci. Eng. Res., 1(1), ISSN: 2229-5518.

Taslaman, N., 2012. Exponential-time algorithms and complexity of NP-hard graph problems. Ph.D. Thesis, IT University of Copenhagen, Section of Theoretical Computer Science.

Une, M. and M. Kanda, 2007. Year 2010 Issues on Cryptographic Algorithms. Institute for Monetary and Economic Studies, Japan. Retrieved form: http://www.imes.boj.or.jp.

Wobst, R., 2001. The Advanced Encryption Standard (AES): The successor of DES. Inform. Secur. Bull., pp: 31-40.

Young, J.O. and D. Yang, 2010. A selective encryption algorithm based on AES for medical information. Health. Inform. Res., 16(1): 22-29.