A Novel Approach for Block-Based Data Encryption

Sainik Kumar Mahata[#], Monalisa Dey[*], Subhranil Som[#]

[#] *JIS College of Engineering, West Bengal. India.*
[*] *JIS College of Engineering, West Bengal. India.*

[1]sainik.mahata@gmail.com
[3]subhranil.som@gmail.com
[2]monalisa.dey.21@gmail.com

***Abstract*—With the growth of data communication over the internet, the need for security has obtained utmost importance. Data needs to be kept hidden from all the internet users apart from the authorized ones. Data should be encrypted before sending it through the internet. Here, an algorithm based on Block-Based data encryption is introduced, where the encryption and decryption process is done on binary data, so it will be applicable to all data in the field of computer science.**

***Keywords*— Bit, Cipher, Decryption, Encryption, Key.**

I. INTRODUCTION

With the rapid growth of data communication, the need to securely transfer data from one computer to another has gained utmost significance. Here I try and introduce an encryption and decryption algorithm based on Block-based data encryption technique. The process of encryption is initiated by generating a key. Key length and data length can be random. We then generate secondary key A from the key. Form secondary key A, we generate secondary key B, C and D. The plain text is then broken down into four parts of equal lengths. We try and segmentize the pain text into lengths that is equal to that of the secondary keys, while the remaining bits remaining unchanged. Each sub blocks are then XNORed with the secondary keys to produce the cipher text. The decryption process is just the opposite of the encryption process. We divide the cipher text into four segments. Each segment id then XNORed with the secondary keys to get the plain text.

II. ALGORITHM

A. Primary and Secondary Key Generation Algorithm

1. We take a random number as the primary key K.
2. This random primary key should be one bit less in length than the segment size of the plain text.
3. We take primary key K as the first secondary key SK1.
4. To generate second secondary key SK2, we retain the i^{th} bit of SK1 and bit wise EXOR the i^{th} and the $(i+1)^{th}$ bit of SK1.
5. To generate second secondary key SK3, the i^{th} bit of SK1 and SK2 are ORed.
6. To generate second secondary key SK4, the i^{th} bit of SK2 and SK3 are EXORed.

B. Encryption Algorithm

1. We break the plain text into four segments.
2. Let us name the segments as PT1, PT2, PT3 and PT4 respectively.
3. We then reduce the PTn size by one by reducing the last bit.
4. We rename these as PPTn.
5. We then XNOR the segments with the secondary keys. i.e,
   PTT1 XNOR SK1, PTT2 XNOR SK2
   PTT3 XNOR SK3, PTT4 XNOR SK4
6. We join the unused bits with the results to get the cipher text.

C. Decryption Algorithm

1. We divide the cipher text into four equal blocks.
2. Let us rename the blocks as CTn.
3. We then reduce the CTn size by one by reducing the last bit.
4. We rename these as CTTn.
5. We then EXOR the segments with the secondary keys. i.e,
   CTT1 XNOR SK1, CTT2 XNOR SK2
   CTT3 XNOR SK3, CTT4 XNOR SK4
6. We join the unused bits with the results to get the plain text.

III. IMPLEMENTATION

Suppose the plain text is 011101010110111101110011.

A. Primary and Secondary key generation

We take a random primary key K as 10011. So, SK1 = 10011.

In Fig. 1, we generate the secondary key 2, SK2 by taking the MSB as it is and XOR ing the i^{th} term with the $(i+1)^{th}$ term.

SK2 = 11010.

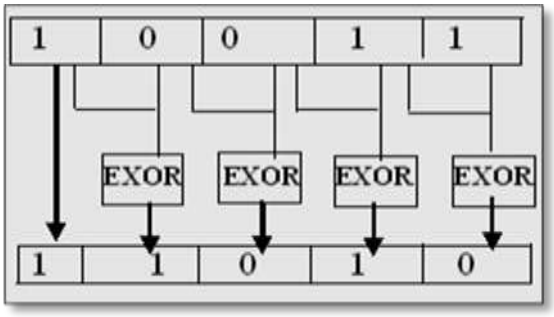

Fig. 1: Generation of the secondary key SK2.

In Fig. 2, we generate the secondary key 3, SK3 by OR in the i^{th} term of SK1 and the i^{th} term of SK2.

SK3 = 11011

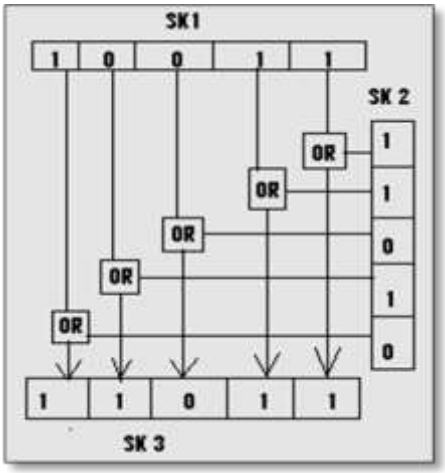

Fig. 2: Generation of the secondary key SK3.

In Fig. 3, we generate the secondary key 4, SK4 by XOR ing the i^{th} term of SK2 and SK3.
SK4 = 00001

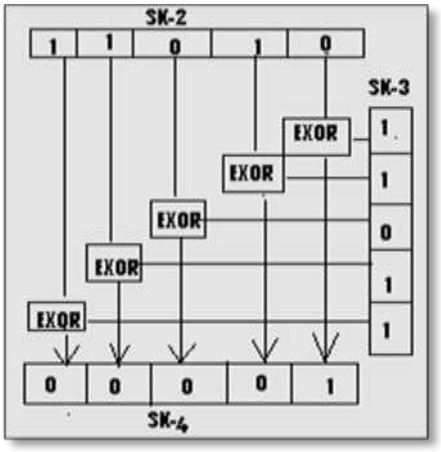

Fig. 3: Generation of the secondary key SK4.

B. Encryption Process

In Fig. 4, we show the division of the plain text into four parts. We name these parts as PTn.

So, PT1 = 011101, PT2 = 010110,
PT3 = 111101, PT4 = 110011

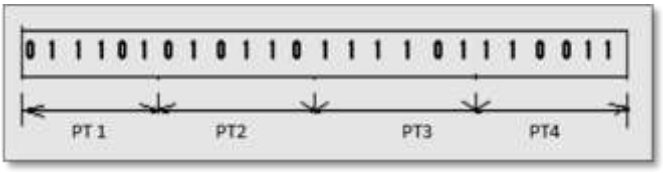

Fig. 4: Division of the plain text into four parts.

In Fig. 5, we show the reduction of the last bits from the four parts that are derived from the plain text. We name these parts as PPTn.

So, PPT1 = 01110, PPT2 = 01011,
PPT3 = 11110, PPT4 = 11001

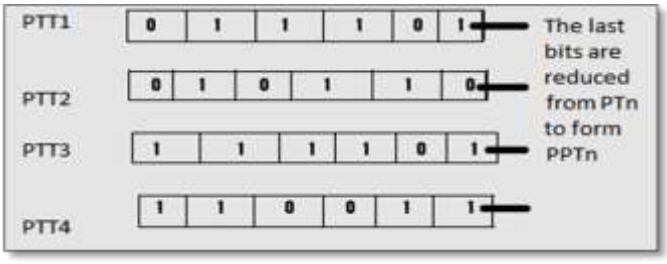

Fig. 5: Reduction of the last bits from PTn to form PPTn.

Fig. 6, shows the XNOR function between PPT1, the part derived from Fig. 5, and secondary key SK1.

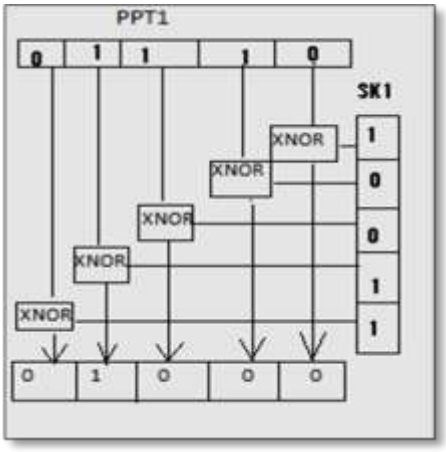

Fig. 6: Implementation of PPT1 XNOR SK1

Fig. 7, shows the XNOR function between PPT2, the part derived from Fig. 5, and secondary key SK2.

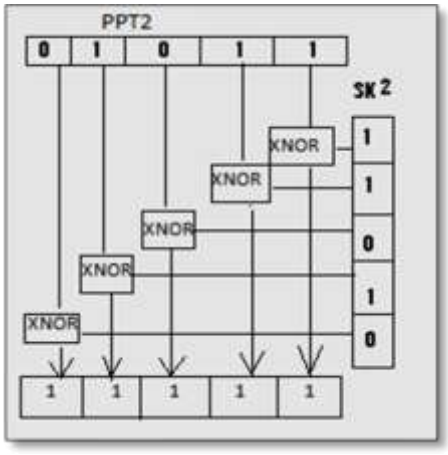

Fig. 7: Implementation of PPT2 XNOR SK2.

Fig. 8, shows the XNOR function between PPT3, the part derived from Fig. 5, and secondary key SK3.

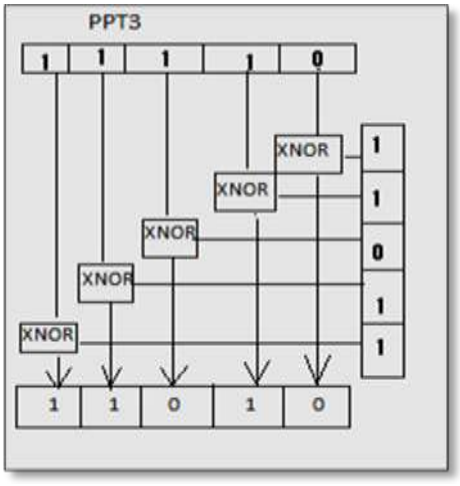

Fig. 8: Implementation of PPT3 XNOR SK3.

Fig. 9, shows the XNOR function between PPT4, the part derived from Fig. 5, and secondary key SK4.

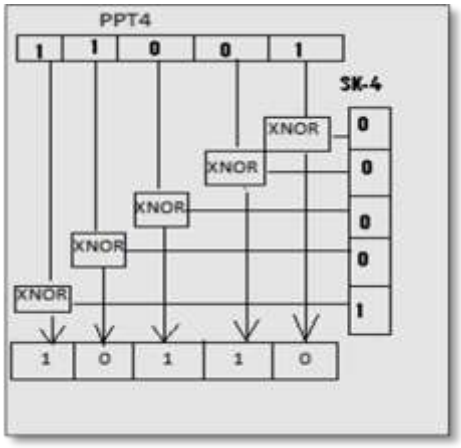

Fig. 9: Implementation of PPT4 XNOR SK4.

Fig. 10 shows the joining of the results found out in Fig. 6, 7, 8 and 9 and the left out bits in Fig. 5 to generate the cipher text.

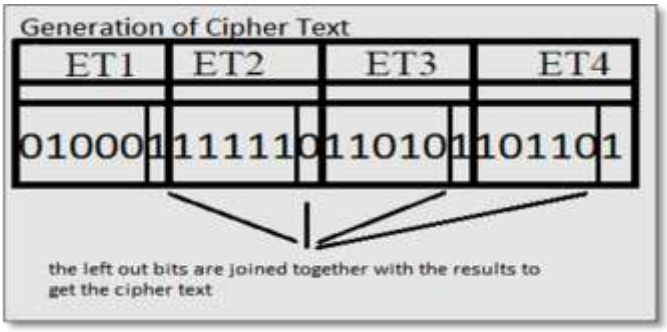

Fig. 10: Generation of the cipher text, joining the results and the left out bits.

C. Decryption Process

For the purpose of decryption, we take the same secondary keys.

Fig. 11 shows the division of the cipher text into 4 parts. We name these parts as CTn.

So,CT1 = 010001, CT2 = 111110,
CT3 = 110101, CT4 = 101101

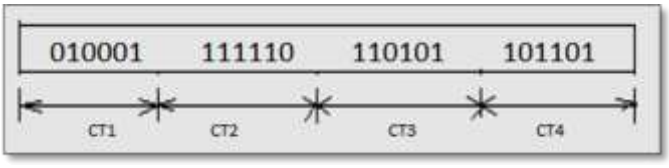

Fig. 11: Division of the cipher text into four parts.

In Fig. 12, we show the reduction of the last bits from the four parts that are derived from the cipher text. We name these parts as CTTn.

So, CTT1 = 01000, CTT2 = 11111,
CTT3 = 11010, CTT4 =10110

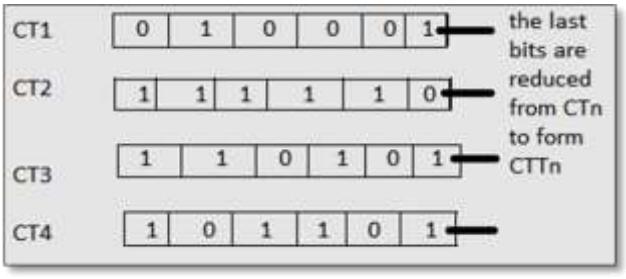

Fig. 12: Reduction of the last bits from CTn to form CTTn.

Fig. 13, shows the XNOR function between CTT1, the part derived from Fig. 12, and secondary key SK1.

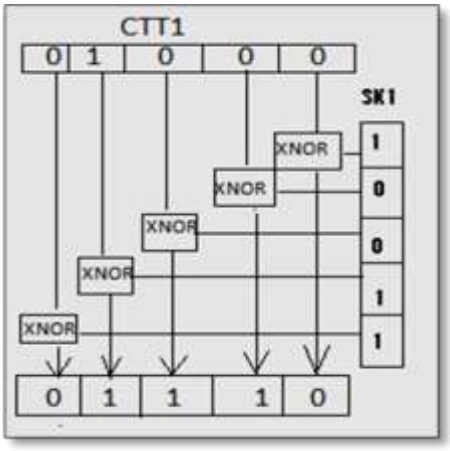

Fig. 13: Implementation of CTT1 XNOR SK1.

Fig. 14, shows the XNOR function between CTT2, the part derived from Fig. 12, and secondary key SK2.

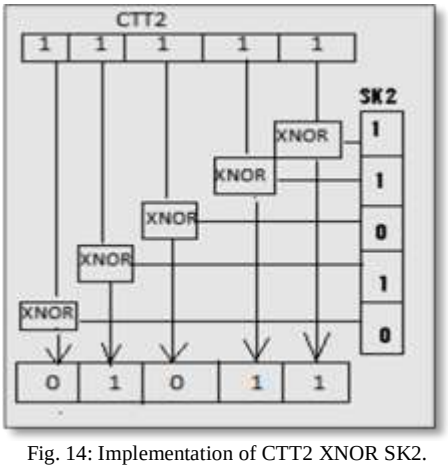

Fig. 14: Implementation of CTT2 XNOR SK2.

Fig. 15, shows the XNOR function between CTT3, the part derived from Fig. 12, and secondary key SK3.

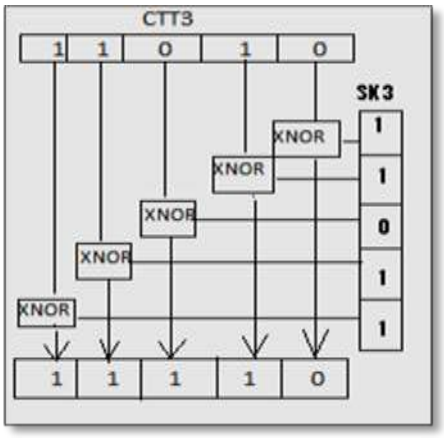

Fig. 15: Implementation of CTT3 XNOR SK3.

Fig. 16, shows the XNOR function between CTT4, the part derived from Fig. 12, and secondary key SK4.

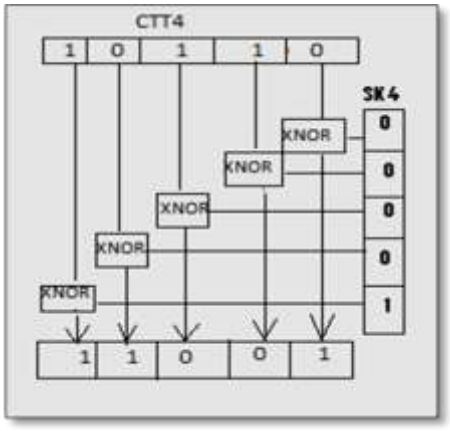

Fig. 16: Implementation of CTT4 XNOR SK4.

Fig. 17 shows the joining of the results found out in Fig. 13, 14, 15 and 16 and the left out bits in Fig. 12 to generate the plain text.

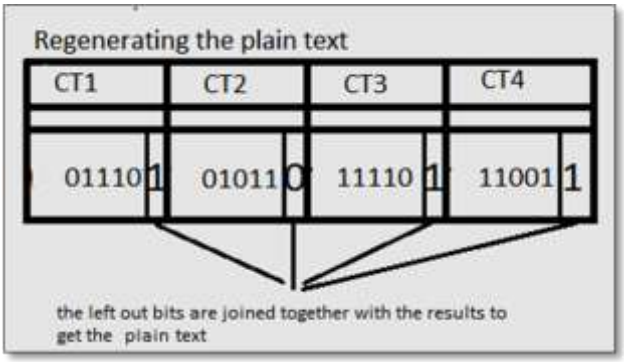

Fig. 17: Generation of the plain text, joining the results and the left out bits.

IV. CONCLUSION

The algorithm has been implemented and designed on binary data. Any data, be it image, sound or text, that is recognized by the computer can be converted to binary data. So this algorithm applies to all types of data. As mentioned earlier, that the key length and the data length is random. So we can take large files and encrypt them. Last but not the least, we have left out the LSB component of the plain text to implement the algorithm. MSB or any other bit can also be left out, to make the algorithm and the encryption process more complex.

REFERENCES

[1] J. K. Mandal, S. Dutta, "*A 256-bit recursive pair parity encoder for encryption*", Advances D -2004, Vol. 9 nº1, Association for the Advancement of Modelling and Simulation Techniques in Enterprises (AMSE, France), www.AMSE-Modeling.org, pp. 1-14

[2] Dutta S., Mal S., "*A Multiplexing Triangular Encryption Technique – A move towards enhancing security in ECommerce*", Proceedings of IT Conference (organized by Computer Association of Nepal), 26 and 27 January, 2002, BICC, Kathmandu.

[3] William Stallings, *Cryptography and network security*, 2005, 4th Edition, Prentice Hall.

[4] Atul Kahate, *Cryptography and network security*, 2005, 4th reprint, Tata McGraw-Hill.

[5] Block Cipher [Online]. Available: http://en.wikipedia.org/wiki/Block_cipher