International Journal of Emerging Engineering Research and Technology
Volume 3, Issue 12, December 2015, PP 18-25
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Efficient Cryptography using Cellular Automata Rules

[1]MD Sadiq, [2]Bhupalam Harish Kumar

[1]Assistant professor, Department of Electronics and Communication, Aurora's Scientific and Technological Institute, Ghatkesar, JNTUH, Telangana, India.

[2]Mtech Scholar, Department of Electronics and Communication, Aurora's Scientific and Technological Institute, Ghatkesar, JNTUH, Telangana, India.

ABSTRACT

A novel cryptographic algorithm mainly depends on encryption and decryption using cellular automata. The algorithm has plaintext and key with 128bits and 12 rounds of operation. Here we are extending size of plaintext and key to 256 bits with 16 rounds of operation. This increases the security. We have increased complexity by implementing different modules such as bit-permutation, inverse bit-permutation NRCA, RCA.

Keywords: RCA. NRCA, bit permutation, RBP.

INTRODUCTION

Cryptography is the art and science of keeping messages secure. The basic objective of network security and cryptographic algorithms is to communicate securely over an insecure medium. The security to the data is provided by the cryptographic algorithms. The word cryptography has come from a Greek word, which means secret writing. The message to be sent through an unreliable medium is known as plaintext, which is encrypted before sending over the medium. The encrypted message is known as cipher text, which is received at the other end of the medium and decrypted to get back the original plaintext message. Cryptographic algorithm is a mathematical function used for encryption and decryption. Cryptographic algorithms are broadly classified into three types: - i.e. symmetric algorithm, asymmetric algorithm and authentication. As per the symmetric algorithm, same key is used for encryption and decryption. In asymmetric algorithm, different keys are used for encryption and decryption. Authentication algorithms mean that the receiver should be sure about sender’s identity. The cellular automata (CA) have been used since the forties of last century. It was used in many physical applications. The applications of Cellular Automata extended to fields such as biological models, image processing, language recognition, simulation, Computer architecture, cryptography etc. The Cellular Automata is also one of the modern methods used to generate binary pseudo–random a Sequences using registers. The concept of CA was initiated by J. Von Neumann and Stan Ulam in the early 1940's. He devised a CA in which each cell has a state space of 29 states, and showed that the devised CA can execute any computable operation. He studied the 1 dimensional rules of Cellular Automata. However, due to its complexity, in the 1970, the mathematician John Conway proposed his now famous game of life which received widespread interest among researchers. His research was based on 2D Cellular Automata rules. Stephen Wolfram studied in much detail and showed that a family of simple one-dimensional cellular automata rules (now

Address for correspondence:

snathick786@gmail.com

famous Wolfram rules) and are capable of emulating complex behavior .The main concern of this paper is secret key systems. In such systems the encryption key and the decryption key are same (symmetric key). The encryption process is based on generation of pseudorandom bit sequences, and CAs can be effectively used for this purpose Cellular Automata (CA) is an organized lattice of cells and each cell have finite number of states, such as "TRUE" (T) or "FALSE" (F). The lattice dimensions can be of any finite value. Each cell within a collection of cells is called as hood. It is particular cell. To start with at time t=0, a state is assigned to the cells. The new states of the cell depend on its own previous state and states of its neighborhood. The new states are assigned based on some predefined rule using mathematical calculations.

Cellular Automata has following inherent Properties

- Parallelism means that the individual cell updates are performed independently of each other. That is, we think of all of the updates being done at once
- Homogeneity means that each cell is updated according to the same rules.

In this paper a new algorithm for encryption and decryption is introduced. Here, we have mostly concentrated on securing the key, by using the concept of cellular automata. In cellular automata, we have different rules based on each rule we generate the random key for each round. By using these keys, we are performing encryption and decryption operation. In encryption, we have xormodule, bit permutation, non reversible cellular automata. In decryption, we have xormodule, reverse bit permutation, non reversible cellular automata.

The structure of the paper is as follows: Section 1 describes introduction, Section 2 describes key generation, Section 3 describes encryption and Section 4 describes decryption, Section 5 describes simulation waveforms, Section 6 describes conclusion, Section 7 describes references.

KEY GENERATION

The input key is 256 bit. It is divided into 2 blocks of 128 bit. Each block is given to RCA module. First block is further divided into sixteen 8 bit blocks then given to rule 30 then we xor k1 with k16 to get the k16 of next block. Likewise k15 is xored with old k16 to get new k15 and so on. After completion of this operation we generate the random key for second round. As for first round the input key itself serves as the round key. In a similar manner, we apply rule 45,rule 86,rule 90,rule 105,rule 150,rule 165,rule 218,rule 30,rule 45, rule 86,rule 90,rule 105,rule 150,rule 165 for the further rounds. The same procedure follows for the second block as well.

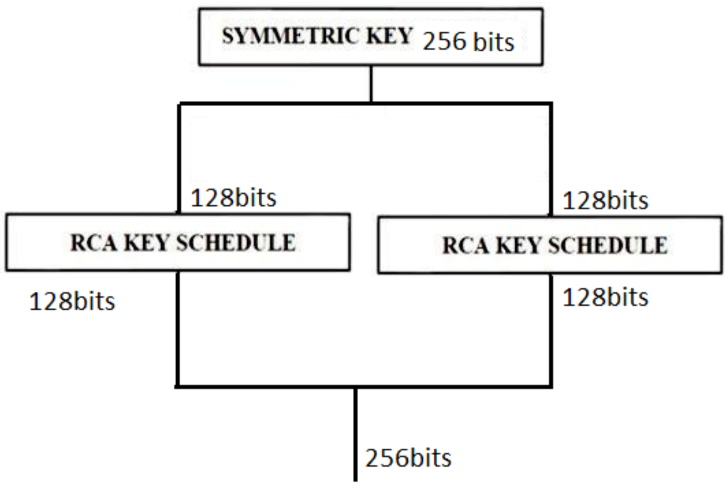


Fig1. *256 bit key schedule*

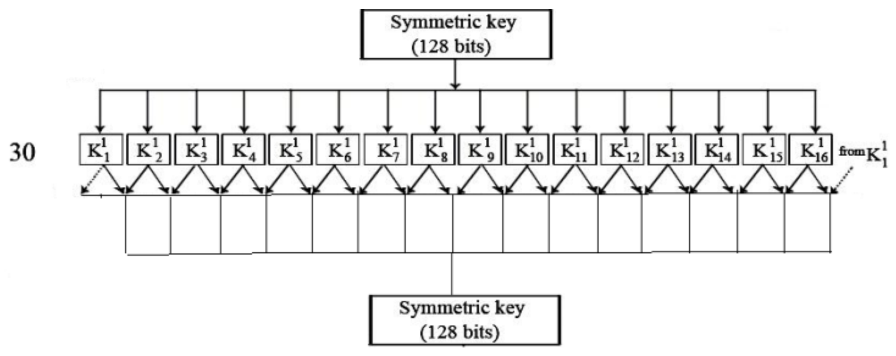


Fig2. *Key generation round*

ENCRYPTION

In encryption, we have three blocks i.e. xorblock, bit permutation, NRCA. The input to encryption is 256 bit plain text. This is divided into two 128 bit blocks. For the first block, we are passing 128 bit plaintext and 128 bit key, then the plaintext and key is further divided into sixteen 8 bit blocks. All these blocks of plaintext and key are given as input to xor block. Then we get sixteen 8 bit blocks as output from xor block. This output is clubbed such that we get 4 blocks of 32 bit. This serves as input to bit permutation block. In bit permutation the bits are arranged in different order based on the formula (9*I mod 31) +1. The permuted output from bit permutation block is divided into sixteen 8 bit blocks and given to NRCA block along with input key. One of the inputs to the NRCA is the 8 bit input key. Each bit of this 8 bit input key is applied with one particular rule. Then the above output along with bit permutation output is xored. The same operation follows for 2nd block also. These operations are performed for 15 more rounds.

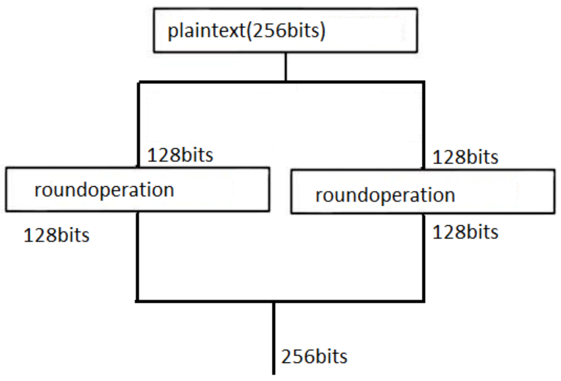


Fig3. *256 bit encryption*

XOR Module

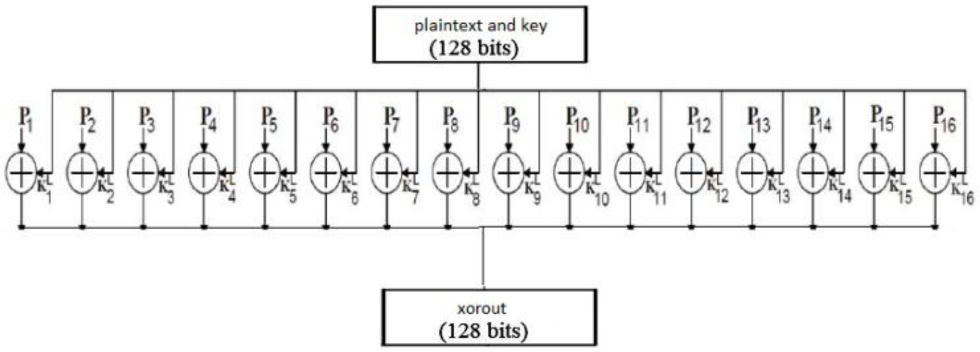

plaintext and key
(128 bits)
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
K1 K2 K3 K4 K5 K6 K7 K8 K9 K10 K11 K12 K13 K14 K15 K16
xorout
(128 bits)

Fig4. *XOR module*

Bit Permutation

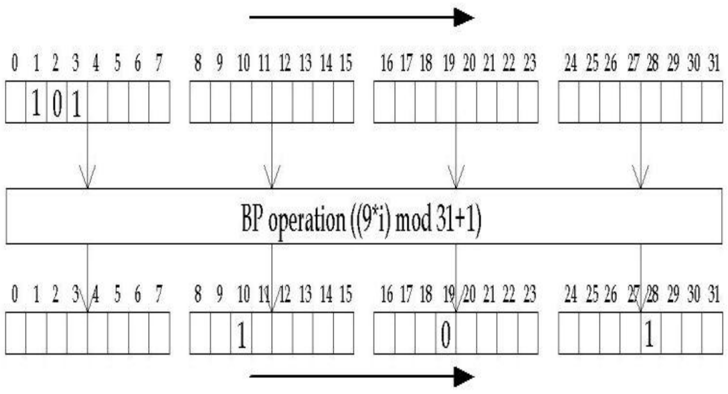


Fig5. *32bit BIT PERMUTATION*

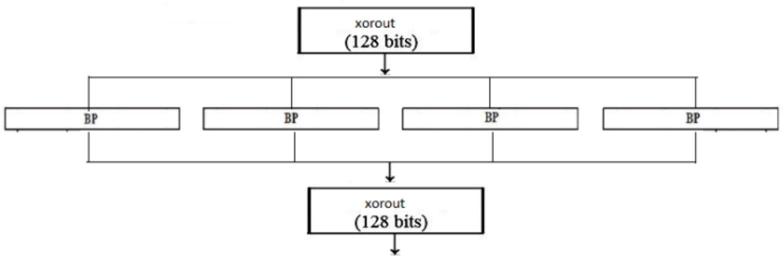


Fig6. *128bit BIT PERMUTATION*

NRCA

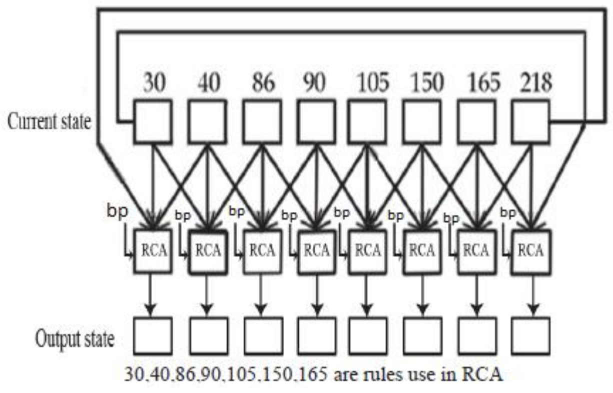
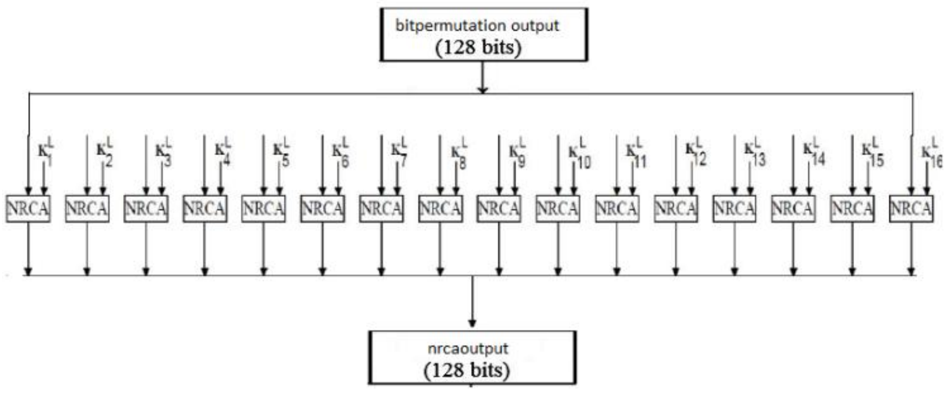


Fig7. *8 bit NRCA*



bitpermutation output
(128 bits)
K^L_1 K^L_2 K^L_3 K^L_4 K^L_5 K^L_6 K^L_7 K^L_8 K^L_9 K^L_{10} K^L_{11} K^L_{12} K^L_{13} K^L_{14} K^L_{15} K^L_{16}
NRCA
nrcaoutput
(128 bits)

Fig8. *128bit NRCA*

DECRYPTION

In decryption, we have three blocks i.e. xor block, Reverse bit permutation, NRCA. The input to decryption is 256 bit cipher text. It is divided into two 128 bit blocks. For the first block, we are passing 128 bit cipher text and 128 bit key, then the cipher text and key is further divided into sixteen 8 bit blocks. All these blocks of cipher text and key are given as input to NRCA block. Then we get sixteen 8 bit blocks as output from xor block. This output is clubbed such that we get 4 blocks of 32 bit. This serves as input to reverse bit permutation block. In reverse bit permutation the bits are re organized into the original position. The output from Reverse bit permutation block is divided into sixteen 8 bit blocks and given to xor block along with input key. The same operation follows for 2nd block also. These operations are performed for 15 more rounds.

NRCA

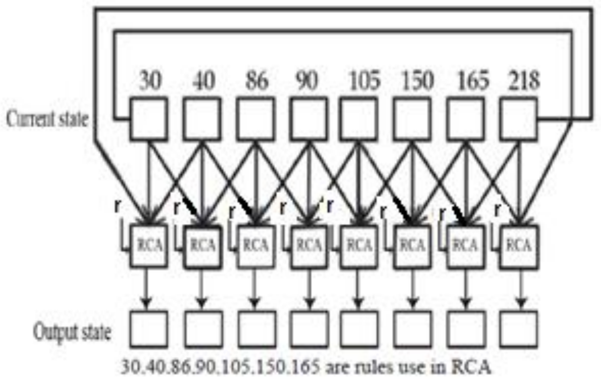


Fig9. *8bit NRCA*

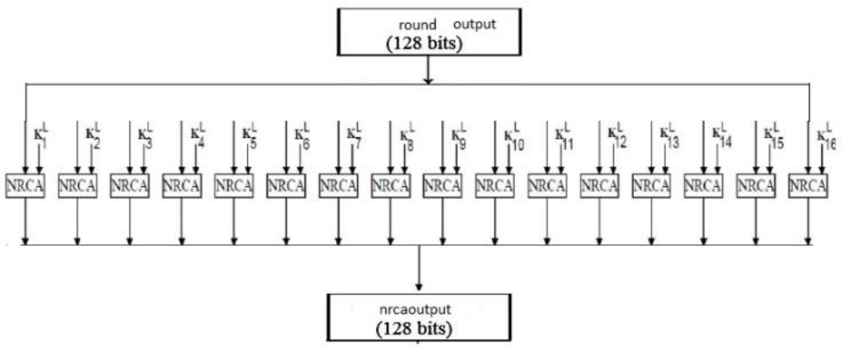


Fig10. *128bit NRCA*

Reverse Bit Permutation

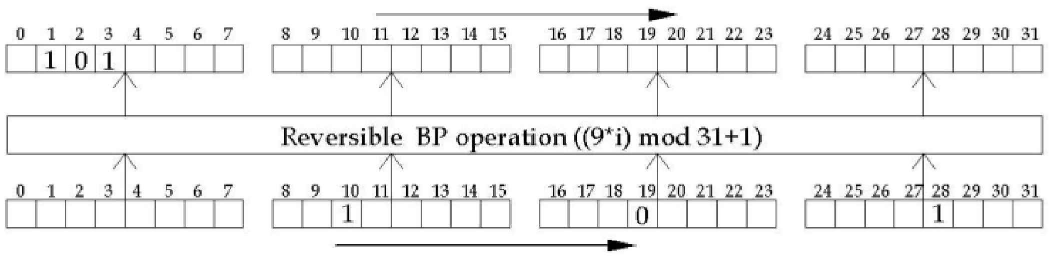


Fig11. *32bit REVERSE BIT PERMUTATION*

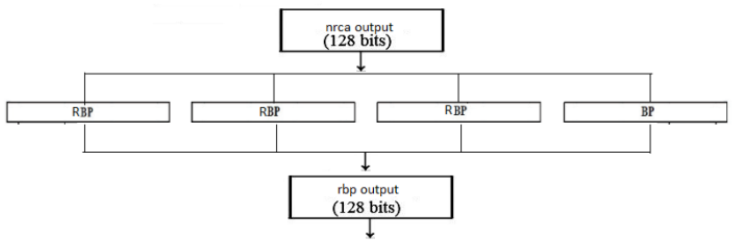


Fig12. *128bit REVERSE BIT PERMUTATION*

XOR Module

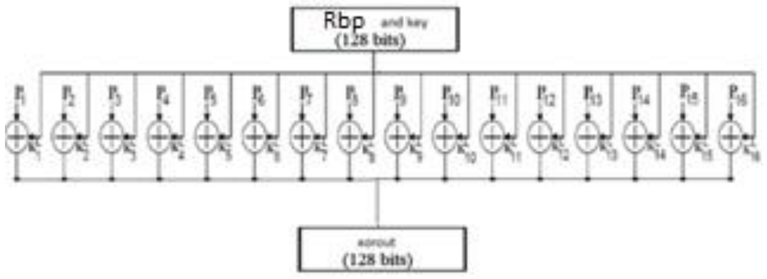
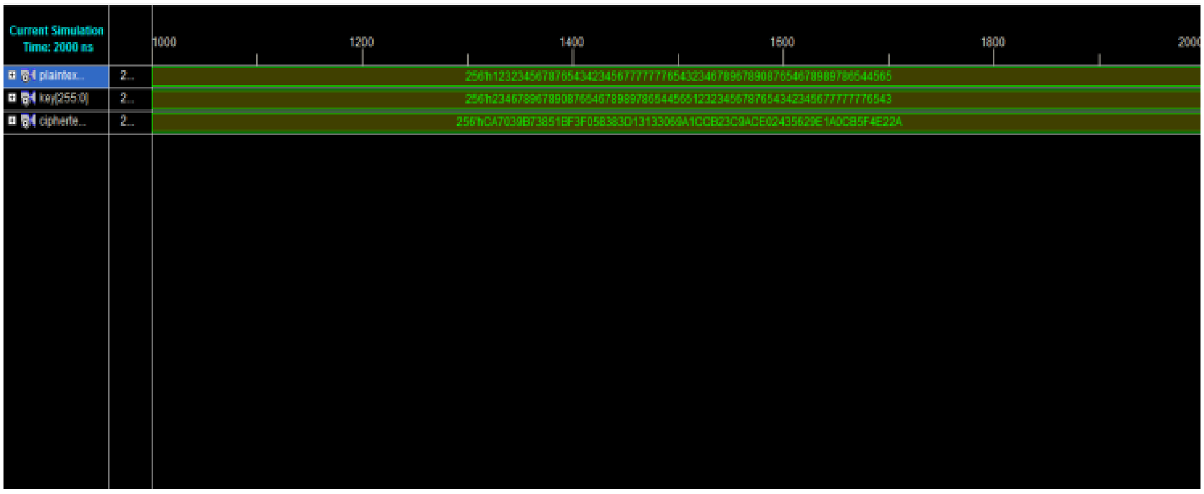

Rbp and key
(128 bits)
P_1 P_2 P_3 P_4 P_5 P_6 P_7 P_8 P_9 P_{10} P_{11} P_{12} P_{13} P_{14} P_{15} P_{16}
K_1 K_2 K_3 K_4 K_5 K_6 K_7 K_8 K_9 K_{10} K_{11} K_{12} K_{13} K_{14} K_{15} K_{16}
xorout
(128 bits)

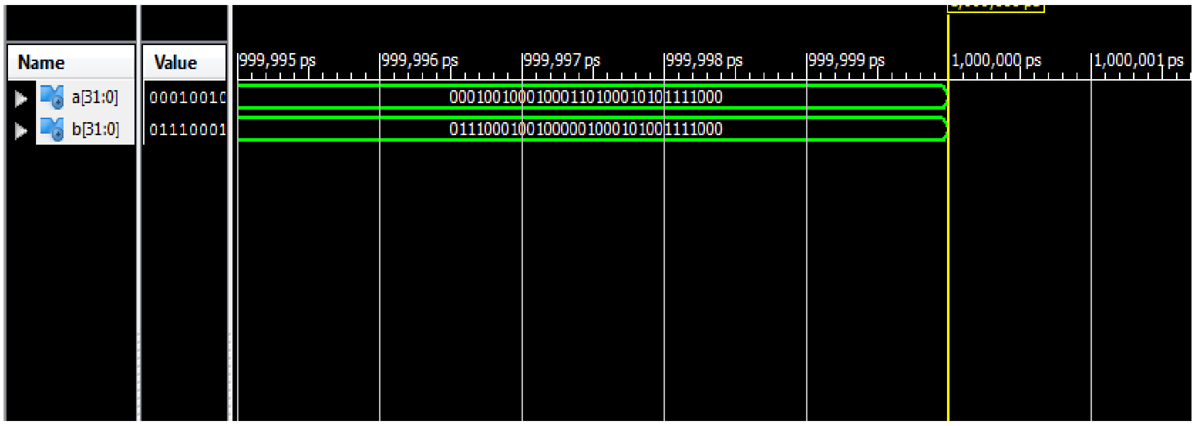
Fig13. *XOR Module*

SIMULATION WAVEFORMS



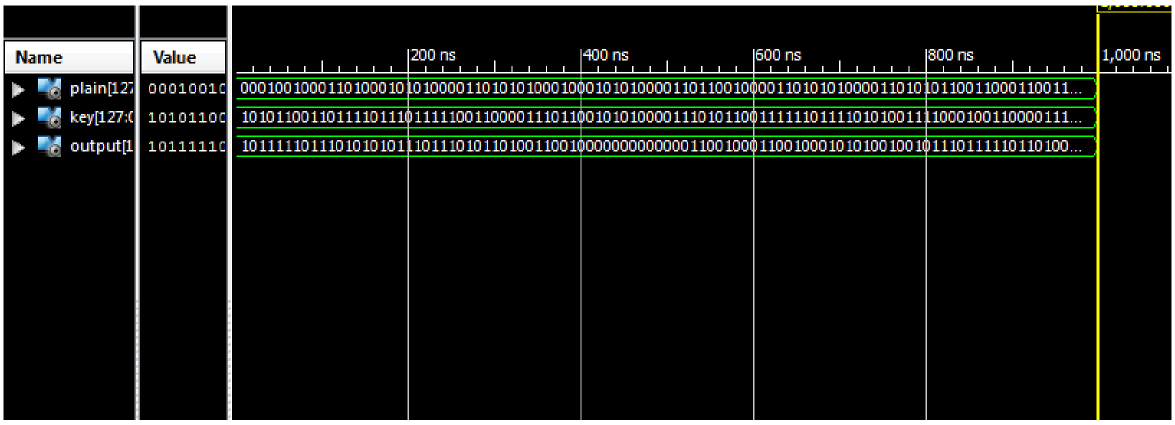
```
plaintex...   2...   256'h12323456787654342345677777777654323467896789087654678989786544565
key[255:0]    2...   256'h2346789678908765467898978654456512323456787654342345677777776543
cipherte...   2...   256'hCA7039B73851BF3F058383D13133069A1CCB23C9ACE02435629E1A0CB5F4E22A
```

256 bit Encryption



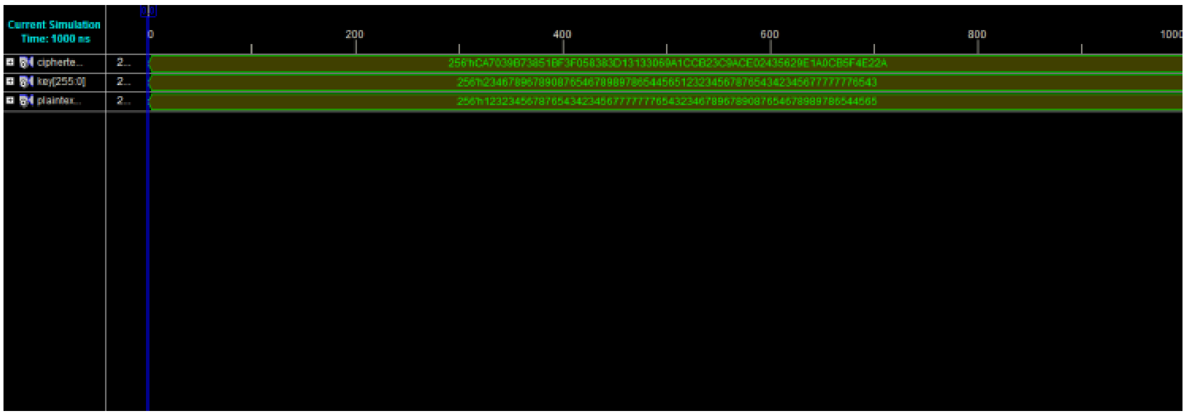
```
a[31:0]   00010010   00010010001000110100010101111000
b[31:0]   01110001   01110001001000001000101001111000
```

32 Bit permutation



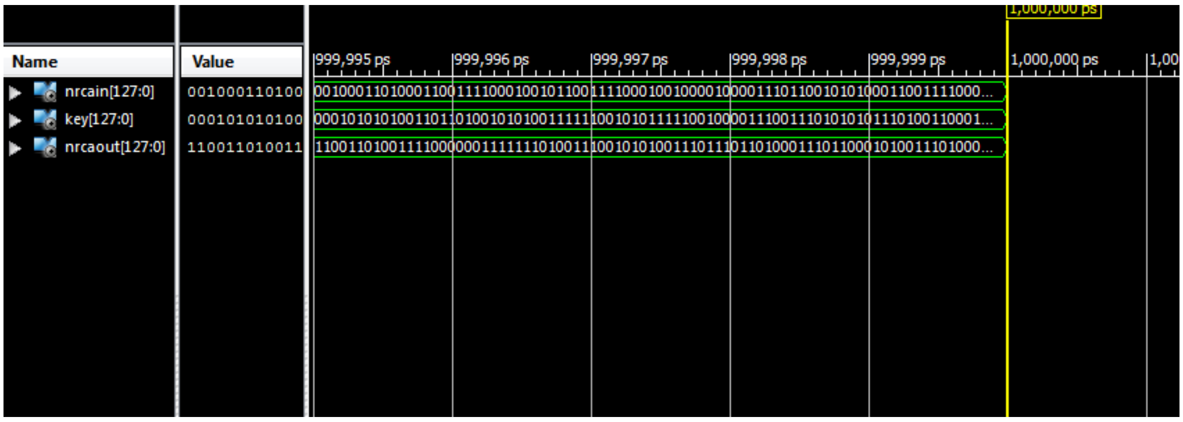
```
Name          Value
plain[127     00010010   000100100011010001010100001101010100010001010100001101100100001101010100001101010110011000110011...
key[127:(     10101100   10101100110111101110111110011000011101100101010000111010110011111101111010100111000100110000111...
output[1      10111110   101111101110101010111011101011010011001000000000000001100100011001000101010010010111011111011010 0...
```

128 bit Xormodule



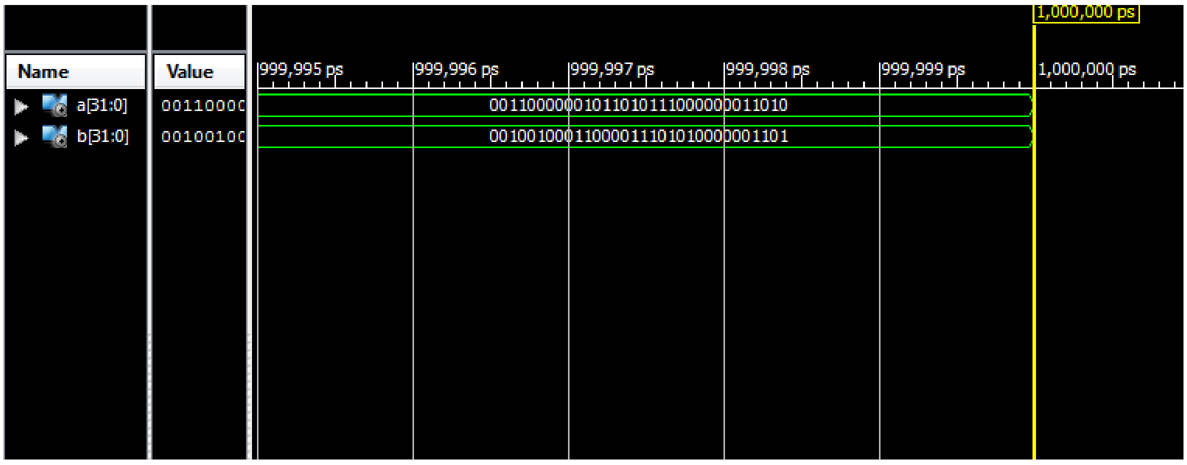
```
Current Simulation
Time: 1000 ns
cipherte...   2...   256'hCA7039B73851BF3F058383D13133069A1CCB23C9ACE02435629E1A0CB5F4E22A
key[255:0]    2...   256'h2346789678908765467898978654456512323456787654342345677777776543
plaintex...   2...   256'h1232345678765434234567777777654323467896789087654678989786544565
```

256 bit Decryption



```
Name            Value
nrcain[127:0]   001000110100   0010001101000110011110001001011001111000100100001000011101100101010001100111100 0...
key[127:0]      000101010100   00010101010011011010010101001111110010101111100100001110011101010101110100110001...
nrcaout[127:0]  110011010011   11001101001111000000111111101001110010101001110111011010001110110001010011101000...
```

128 bit NRCA



```
Name       Value
a[31:0]    00110000   00110000001011010111000000011010
b[31:0]    00100100   00100100011000011101010000001101
```

32 bit Reverse bit permutation

CONCLUSION

In this paper the 256 bit cellular automata encryption and decryption is designed and synthesized using vhdl in Xilinx ise 13.2i

REFERENCE

[1] B.Schneier, "Applied Cryptography," Wiley, New York, 1996.

[2] S Tripathy and S Nandi, CASE: "Lightweight Cellular Lightweight Cellular Automata-based Symmetric-key encryption.

[3] Palash Sarkar, "A Brief History of Cellular automata," Journal of ACM Computing Surveys (CSUR), Volume 32 Issue 1, March 2000.

[4] S. Wolfram, "Cryptography with Cellular Automata," Crypto '85, LNCS 218, pp. 429-432, Springer-Verlag, 1986.

[5] S. Wolfram, "Random sequence generation by cellular automata," Advances in Applied Maths, vol. 7, No. 2, pp. 123-169, 1986.

[6] Franciszek Seredynski, Pascal Bouvry, and Albert Y. Zomaya. "Cellular automata Computations and secret key cryptography," Parallel Computing Journal, 30(5-6):753–766, 2004. F. Standaert, G. Piret, G. Rouvroy, J. Quisquater, and J. Legat, "ICEBERG: An involutional cipher efficient for block encryption in reconfigurable hard- ware," FSE '04, LNCS 3017, pp. 279-299, Springer- Verlag, 2004.

[7] N. Sklavos, N. A. Moldovyan, and O. Koufopavlou, "High speed networking: Design and implementation of two new DDP-based ciphers, Mobile Networks and Applications-MONET," Vol. 25, No. 1-2, pp. 219-231, Springer-Verlag, 2005.

[8] N. A. Moldovyan, P. A. Moldovyan, and D.H. Sum- merville, "On software implementation of fast DDP- based ciphers," International Journal of Network Security, Vol. 4, No. 1, pp. 81-89, 2007.

[9] T. Toffoli and N. Margolus, "Invertible cellular automata: A review," Physica D, vol. 45, pp. 229-253, (reprinted with correction as of Oct. 2001).