# Secure Transmission of Data using Rabbit Algorithm

**Shweta S Tadkal[1], Mahalinga V Mandi[2]**

[1] *M. Tech Student , Department of Electronics & Communication Engineering, Dr. Ambedkar institute of Technology,Bengaluru-560056*

[2] *Associate Professor, Department of Electronics & Communication Engineering, Dr. Ambedkar institute of Technology,Bengaluru-560056*

---------------------------------------------------------------------***---------------------------------------------------------------------

**Abstract—** This paper presents the design and simulation of secure transmission of data using rabbit algorithm. The rabbit algorithm is a stream cipher algorithm. Stream ciphers are an important class of symmetric encryption lgorithm, which uses the same secret key to encrypt and decrypt the data and has been designed for high performance in software implementation. The data or the plain text in our proposed model is the binary data which is encrypted using the keys generated by the rabbit algorithm. The rabbit algorithm is implemented and the language used to write the code is Verilog and then is simulated using Modelsim6.4a. The software tool used is Xilinx ISE Design Suit 14.7.

**Keywords—Cryptography, Stream ciphers, Rabbit algorithm**

## 1. INTRODUCTION

In today's world most of the communication done using electronic media. Data securit plays a vital role in such communication. Hence there is a need to predict data from malicious attacks. This is achieved by cryptography. Cryptography is the science of secret codes, enabling the confidentiality of communication through an in secure channel. It protects against unauthorized parties by preventing unauthorized alteration of use. Several encrypting algorithms have been built to deal with data security attacks. Encryption algorithms are concerned of transforming readable texts(plain text) to unreadable text(cipher text). In stream ciphers, the encryption algorithm generates a stream of bits that are ex-or'ed with a stream of plain text bits to generate a stream of cipher texts. Traditionally, stream ciphers use secret key to initiate the key generation method. For security functions, these key ought to be long enough (at least 128 bit) to satisfy the minimum security needs.

The rest of the paper is organized as follows; Section 2 gives a brief literature survey of the related work. Section 3 presents brief explanation of rabbit algorithm. Section 4 presents proposed work and section 5 concludes the paper.

## 2. RELATED WORK

Few applications have been implemented using rabbit algorithm. Muhammad Anwari Leksono et.al, [1] proposed a rabbit algorithm in e-mail application for android smart phone to secure e-mail content. Rabbit algorithm was used to encrypt and decrypt the e-mail's content. It can be used to send, retrieve, edit, create etc., the author has chosen java language to implement the application, as java has several free libraries that change and support the appliance. In eclipse 3.6 with android plug-in, android SDK, JDK 1.6 and JRE 16, the application is developed. Using Sony Xperia Ray with android version 4.0.4 operating system, the application is tested. Using SMTP protocol e-mails are sent. Google mail is the only e-mail service that is working for this application. To retrieve e-mails, IMAP protocol is used.

Ruhma Tahir et.al, [2] Proposed a mechanism used in wireless sensor network to provide confidentiality, referred as LSRA i.e. light weight encryption mechanism based on rabbit stream cipher. In wireless sensor networks, the LSR provides the data confidentiality needs for all security applications. The LSR was implemented to meet the goals such as performance, security and ease of use. Two schemes were proposed, one is Symmetric Key Cryptography (SKC) based scheme to encrypt bulk data and another is Public Key Cryptography (PKC) to encrypt the secret key used for communication. The simulator of tinyOS i.e. TOSSIM was used to test LSRA. The time taken to encrypt and decrypt 128-bits of plain text is 39us.

Khaled Suwais et.al, [3] presented a paper on parallel model for rabbit stream cipher for multi core processor. Improving the performance by accelerating the keystream generation and encryption process was his main goal. Parallel processing can be described as the usage of multi processors for solving the computational problem, where a problem is divided into segments and solved concurrently using multiple processors. The experiment is carried on three different platforms i.e.

Platform 1: Intel Pentium IV of CPU speed 1.93 GHz (single core).

Platform 2: Intel Dual-core of CPU speed 2.93 GHz (two cores).

Platform 3: Intel core 2 quad of CPU speed 2.40 GHz (four cores).

Khalida Shaaban Rijab et.al., [4] presented a paper for designing two special Huffman tree (SHT) and implementing them for encoding with an MPEG video file instead of standard Huffman tree algorithm. Each SHT was built with 89 and 100 entries respectively. One part of key stream generated by rabbit algorithm is used for encrypting SHTs and the other part is used in insertion operation. To evaluate the performance of this algorithm, many types of tests and measurements are performed such as efficiency, compression, speed and security measurements. The primary goal was to get a key which is large enough against well known attacks, save the computation time by taking the advantage of combining MPEG compression and data encryption, and avoid affecting the video compression ratio.

Fikaril Akhyar et.al, [5] proposed a rabbit algorithm implementation for video on demand based on digital rights management. The purpose was to improve the security of video data and analyze the performance to calculate the encrypt and decrypt the processing time, avalanche effect and video quality. The video is split into frames and encryption is done on the frames. The time taken to encrypt the video depends on the amount of frames processed.

## 3. RABBIT ALGORITHM

Rabbit is a synchronous stream cipher which was presented in 2003 at Fast Software Encryption (FSE) workshop by Martin Boesgaard, Mette Vesterager, Thomas Christensen and Erik Zenner [6]-[8]. Before introducing this algorithm there was no IV set up function, which provides additional security. The goal of this algorithm is to provide higher security and speed. Rabbit was designed to be faster than commonly used ciphers. As of now, there are no cryptographical weaknesses.

It takes 128-bit secret key and 64-bit IV as input and generates for each iteration an output block of 128-bit pseudo random bits from a combination of internal 513 bits. The internal bits are further divided into eight 32-bit state variables, eight 32-bit counter variables and one carry bit. For an attacker who does not know the key, it is not possible to distinguish upto $2^{64}$ blocks of cipher output from the output of a truly random generator, needs to calculate $2^{128}$ possible combination of keys.

### A. Key Set up Scheme

The first step in the algorithm is to set up a key. The 128-bit key is divided into eight sub keys each of 16-bit. Then the state variables and counter variables are calculated from the sub keys. The system is then iterated four times, as per the next state function to diminish correlation between bits in the key and internal state variables.

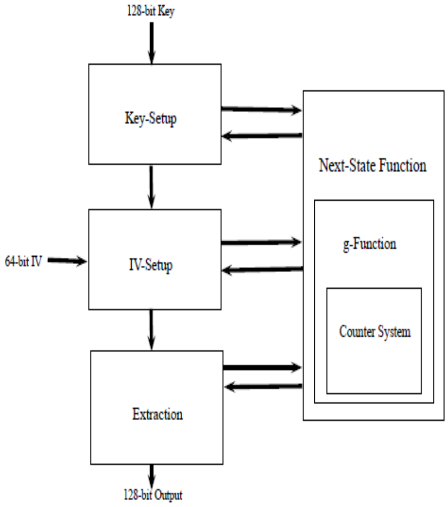


Fig 1. Block diagram of rabbit algorithm

### *B. IV Set up Scheme*

Second step is the IV set up scheme. The IV set up scheme works by modifying the counter state as function of IV, which is done by XOR'ing the 64-bit IV on all the 256- bits of the counter variables.

### *C. Next State Function*

Next step is the next state function; this is the core of the rabbit algorithm. It ensures the right mixture of bits of IV using the values of counters and state registers.

### *D. Counter System*

Next step is the counter system; in this the counter registers are updated by combining the current state of all counter registers with a constant and carry bit value.

### *E. Extraction Scheme*

Final step is the Extraction scheme; in this the XOR operation is applied on different state registers to create eight 16-bit key stream registers. These key bits are then used to XOR with the plain text bit stream.

## 4. PROPOSED SYSTEM

The proposed system can be explained with transmitter and receiver as shown in the Fig.1 and Fig.2 respectively.

At transmitter side as shown in Fig.2, the plain text is encrypted to generate a cipher text. The plain text in our proposed model is the binary data which is XOR'ed with the keys generated by rabbit algorithm. The input to the rabbit algorit

hm is random 64-bit initialization vector (IV) and random 128-bit key which generates random key as output. The output of the XOR will be the encrypted data.

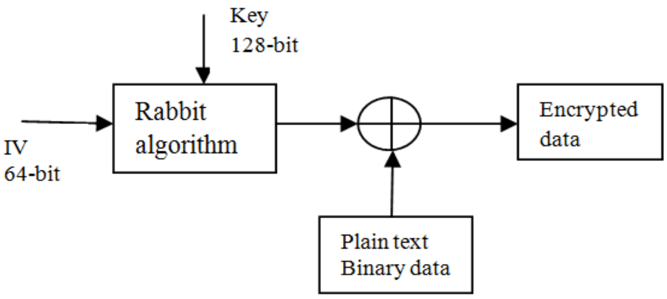


Fig.2 Transmitter

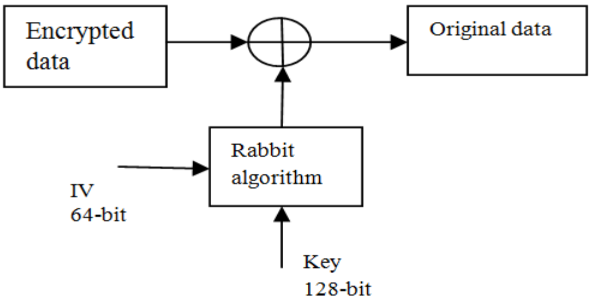


Fig.3 Receiver

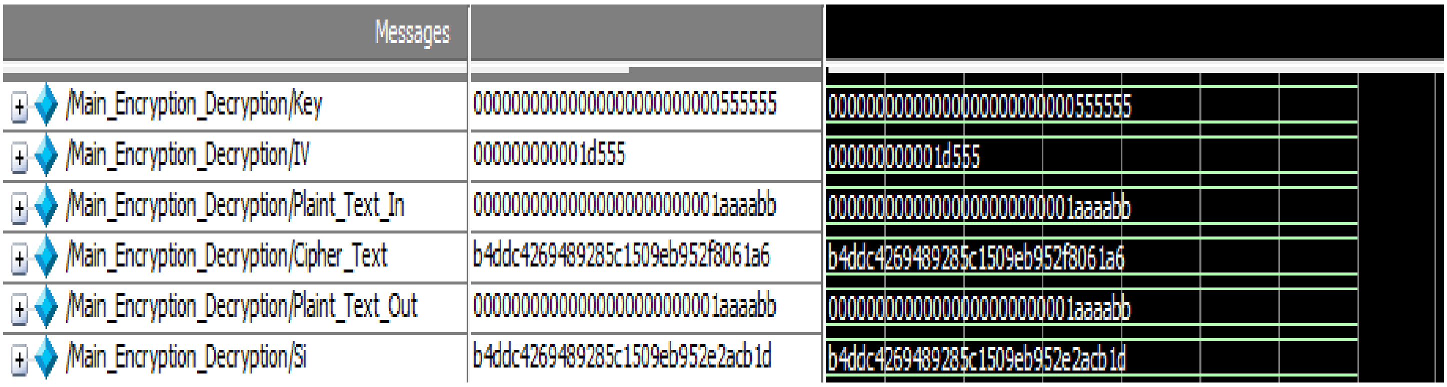
At the receiver side as shown in Fig.3, the encrypted data will be the input which is XOR'ed with the keys generated by the rabbit algorithm. The output will be the original data.

## 5. RESULTS AND DISCUSSION

The input to the rabbit algorithm is the 128-bit key and 64-bit IV, which is combined to get the 128-bit pseudo random key. Using those pseudo random generated keys the plain text (in our case it is 128-bit binary data), is encrypted to get the cipher text. Afted that, the encrypted data is xor'ed with rabbit generated keys to get the original data.

The Fig.3 shows the simulation result for encryption and decryption the data using rabbit algorithm. The simulation is done using modelsim6.4a simulator.

The 128-bit data considered is [00000000000000000000000001aaaabb]H
The 128-bit key generated using the rabbit algorithm is [b4ddc4269489285c1509eb952e2acb1d]H
The cipher text generated is [b4ddc4269489285c1509eb952f8061a6]H
The decrypted original data is [00000000000000000000000001aaaabb]H

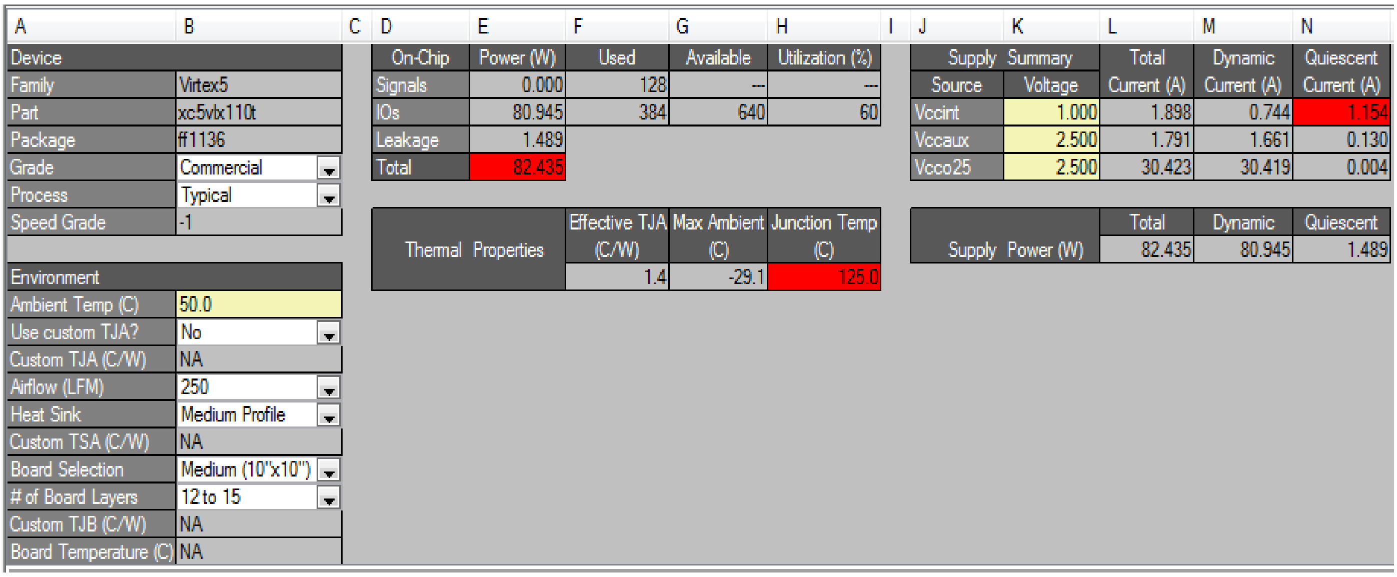


| Messages | | |
|---|---|---|
| /Main_Encryption_Decryption/Key | 00000000000000000000000000555555 | 00000000000000000000000000555555 |
| /Main_Encryption_Decryption/IV | 000000000001d555 | 000000000001d555 |
| /Main_Encryption_Decryption/Plaint_Text_In | 00000000000000000000000001aaaabb | 00000000000000000000000001aaaabb |
| /Main_Encryption_Decryption/Cipher_Text | b4ddc4269489285c1509eb952f8061a6 | b4ddc4269489285c1509eb952f8061a6 |
| /Main_Encryption_Decryption/Plaint_Text_Out | 00000000000000000000000001aaaabb | 00000000000000000000000001aaaabb |
| /Main_Encryption_Decryption/Si | b4ddc4269489285c1509eb952e2acb1d | b4ddc4269489285c1509eb952e2acb1d |

Fig 3. Simulation result for encryption and decryption

The delay and power calculation is given below

Delay=5.939ns



| A | B |
|---|---|
| Device | |
| Family | Virtex5 |
| Part | xc5vlx110t |
| Package | ff1136 |
| Grade | Commercial |
| Process | Typical |
| Speed Grade | -1 |
| | |
| Environment | |
| Ambient Temp (C) | 50.0 |
| Use custom TJA? | No |
| Custom TJA (C/W) | NA |
| Airflow (LFM) | 250 |
| Heat Sink | Medium Profile |
| Custom TSA (C/W) | NA |
| Board Selection | Medium (10"x10") |
| # of Board Layers | 12 to 15 |
| Custom TJB (C/W) | NA |
| Board Temperature (C) | NA |

| On-Chip | Power (W) | Used | Available | Utilization (%) |
|---|---|---|---|---|
| Signals | 0.000 | 128 | --- | --- |
| IOs | 80.945 | 384 | 640 | 60 |
| Leakage | 1.489 | | | |
| Total | 82.435 | | | |

| Thermal Properties | Effective TJA (C/W) | Max Ambient (C) | Junction Temp (C) |
|---|---|---|---|
| | 1.4 | -29.1 | 125.0 |

| Supply Source | Summary Voltage | Total Current (A) | Dynamic Current (A) | Quiescent Current (A) |
|---|---|---|---|---|
| Vccint | 1.000 | 1.898 | 0.744 | 1.154 |
| Vccaux | 2.500 | 1.791 | 1.661 | 0.130 |
| Vcco25 | 2.500 | 30.423 | 30.419 | 0.004 |

| | Total | Dynamic | Quiescent |
|---|---|---|---|
| Supply Power (W) | 82.435 | 80.945 | 1.489 |

Fig 4. Power consumption

## 6. CONCLUSION

In this work a method is proposed to implement rabbit algorithm and can be used for encrypting the binary data. The rabbit algorithm is implemented using Verilog and is simulated using Modelsim6.4a. The main advantage of using rabbit algorithm as a stream cipher is fast, efficient and more secure algorithm more secure algorithm.

## 7. REFERENCES

[1] Muhammad Anwari Leksono, Rinaldi Munir, 2012 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA) "Email Client Application with Rabbit Algorithm for Android Smart Phone".

[2] Ruhma Tahir, Muhammad Younas Javed, Muhammad Tahir, Firdous Imam 2008 IEEE DOI 10.1109/IITA.2008.523 "LRSA: Lightweight Rabbit based Security Architecture for Wireless Sensor Networks".

[3] Khaled Suwais, "Parallel Model for Rabbit Stream Cipher over Multi-core Processors" Volume 11, 2014 WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS.

[4] Sufyan T. Faraj Al-Janabi, Khalida Shaaban Rijab, Ali Makki Sagheer : "Video Encryption Based on Special Huffman Coding and Rabbit Stream Cipher" 2011 Developments in E-systems Engineering.

[5] Fikaril Akhyar, Surya Michrandi Nasution & Tito Waluyo Purboyo "Rabbit Algorithm for Video on Demand" 2015 IEEE Asia Pacific Conference on Wireless and Mobile.

[6] Martin Boesgaard, Mette Vesterager, Thomas Christensen, Erik Zenner "The Stream Cipher Rabbit" ECRYPT Stream Cipher Project Report 2005/006.

[7] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius, "Rabbit: A New High-Performance Stream Cipher", proceedings of Fast Software Encryption (FSE'03), LNCS: 2887, pp. 307-329, Springer-Verlag, 2003.

[8] Parameshachari B D,K M Sunjiv Soyjaudah,Sumitrha Devi K A, "Secure Transmission of an Image using Partial Encryption based Algorithm" International Journal of Computer Applications (0975 – 8887) Volume 63– No.16, February 2013.