ISSN No. 0976-5697
Volume 2, No. 3, May-June 2011
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Shuffle-RAT: An FPGA-based Iterative Block Cipher

Rajdeep Chakraborty*
Department of Computer Science and Engineering,
Netaji Subhas Engineering College,
Kolkata, West Bengal, India.
rajdeep_chak@yahoo.co.in

Sananda Mitra
Department of Computer Science and Engineering,
Netaji Subhas Engineering College,
Kolkata, West Bengal, India.
sananda.mitra8@gmail.com

J. K. Mandal
Department of Computer Science and Engineering,
University of Kalyani,
Kalyani, West Bengal, India.
jkm.cse@gmail.com

Abstract: Proposed FPGA based technique considers a message as a binary string on which Shuffle-RAT is applied. A block of n-bits is taken as an input stream, where n ranges from 8 to 256 – bit, then Shuffle-RAT is applied in each block to generate intermediate stream, any one intermediate stream is considered as a cipher text. The same operation is performed repeatedly on various block sizes. It is a kind of block cipher and symmetric in nature hence decoding is done in similar manner. This paper also presents an efficient hardware realization of the proposed technique using state-of-the-art Field Programmable Gate Array (FPGA). The technique is also coded in C programming language and Very High Speed Integrated Circuit Hardware Description Language (VHDL). Various results and comparisons have been performed against industrially accepted RSA and TDES. A satisfactory results and comparisons are found.

Keywords: cryptography; iterative block cipher; modulo-addition; RAT; FPGA.

I. INTRODUCTION

The explosion of communications systems brought with it a demand for methods to protect sensitive information and to provide security services. Cryptography provides us with means for securing and authenticating digital communication over insecure network [1,2,3,4,5,8]. In the modern day scenario, the cryptographic methods can be broadly classified in two branches; namely Symmetric Key and Public Key cryptography [3]. While the symmetric key ciphers consider the case of a shared key between the sender and the receiver, the public key ciphers consider one key for encryption and another different key for decryption. For data communication, symmetric key ciphers are most popular due to their low computational complexity. The proposed cipher is of symmetric key systems – Block Ciphers [5]. A block cipher is an encryption scheme that breaks up the plaintext message into blocks of a fixed length and encrypts one block at a time. Iterative block cipher [8] is a technique of Block Ciphers that encrypts a plaintext block by a process that has several rounds, and some computational transformation, called the round function, is applied to the data in each round, using a sub-key.

In recent times a number of iterated block cipher models have been proposed in the literature. One of them is a design using Rotational Addition Technique (RAT)[1]. This is an efficient iterative block cipher with proven security features, which is suitable for implementation in microprocessors. In this paper, a modified version of RAT is proposed as Shuffle-RAT, which provides much better efficiency and diffusion compared to the existing cipher. This technique has been implemented in FPGA platform using VHDL through Xilinx [2]. Section II of this paper discusses the new scheme Shuffle-RAT, including the description of the algorithm and related key generation. Section III describes the implementation of the new cipher on FPGA [6]. Section IV contains the experimental results and comparisons. Conclusion is in Section V with acknowledgments in Section VI, and details of references are given in Section VII.

II. SHUFFLE-RAT: THE PROPOSED CIPHER

The plaintext file for RAT is considered as a stream of 512 bit blocks. The basic round function is rotational addition, applied on the 512-bit plaintext over 8 rounds. The plaintext is subdivided into smaller blocks in each round of RAT, where the block sizes vary with the powers of 2 in the rounds, i.e., 2^n-bit blocks are considered for round 'n', where 'n = 1, 2, 3 … 8'. In the 'n-th' round of RAT, the rotational addition adds each block to the adjacent block modulo '$2^{n'}$', and stores the result in the second block, iteratively over the length of the plaintext. In mathematical terms, the round function of RAT is as follows.

$$B_{i+1} = (B_i + B_{i+1}) \bmod 2^n \tag{1}$$

In equation (1), the index 'i' cover all the blocks in each round. Each round of RAT is iterated for some number of times defined by 'keys', where the round-keys are of size 16 bits each. Thus, the total key-size of RAT is 8 x 16 = 128 bits. Decryption for RAT is just the opposite of encryption, where one has to use modular subtraction instead of addition and the round-keys are considered in the reverse order.

A close study of RAT reveals a few areas for improving the design even further. The degree of randomness may be increased for better homogeneity and security than the previous scheme. In terms of improving the algorithm, it was observe that RAT has a strong property of 'confusion', like all good block ciphers, but lacks good 'diffusion'. Thus, we propose the diffusion of RAT with the technique of butterfly shuffle to produce a new cipher – Shuffle-RAT. Sub-section A gives the algorithm of Shuffle-Rat, Sub-section B illustrates an example and Sub-Section C deals with the key generation issues.

A. *Algorithm of Shuffle-RAT*

The algorithm of the scheme Shuffle-RAT is based on RAT, and can be summarized as follows:

- *Step 1:* The 512 bit message is divided into a number of blocks; each containing $N = 2^n$ bits, where N is any one of 2, 4, 8, 16, 32, 64, 128, 256.
- *Step 2:* Two adjacent blocks are added and result is stored in the 2nd block where the modulus of addition is 2^n as in the case of RAT.
- *Step 3:* Each round key of 16 bits, produced similar to that in RAT, is divided into two parts each of 8 bits. Suppose that they are named as *key1* and *key2*.
- *Step 4:* Original RAT is performed on the blocks of size *N* for (*key1)* times of iterations.
- *Step 5:* The blocks are shuffled within the message to create proper diffusion. This is done by a simple butterfly shuffle, shuffling pairs of adjacent blocks, and the shuffling is done just once.
- *Step 6:* Finally, *key2* number of iterations of original RAT is performed on these shuffled blocks, each of size *N*, to complete the encryption.

Thus, Shuffle-RAT tries to incorporate diffusion in the structure of RAT using the butterfly shuffle, sandwiched between two regular rounds of RAT, which already provides sufficient amount of confusion as in the original design.

B. *Example*

This Sub-section illustrates the mechanism of Shuffle-RAT using a 16-bit plaintext, and by simplifying the iterative structure as follows: a single iteration of RAT followed by butterfly shuffle followed by another single iteration of RAT for each round. In Table I, each round has the three stages listed one by one; a complete example has been illustrated.

During decryption one can simply invert the above steps, and use modular subtraction to perform the decryption for the cipher text to produced the plain text back.

C. *Key Generation*

In the scheme of RAT, eight rounds are considered, each for 2, 4, 8, 16, 32, 64, 128 and 256-bit block size. Each round is repeated for a finite number of times and the number of iterations will form a part of the encryption-key. Although the key may be formed in many ways, for the sake of brevity it is proposed to represent the number of iterations in each round by a 16-bit binary string. The binary strings are then concatenated to form a 128-bit encryption key. An example of key generation for Shuffle-RAT is illustrated in Table II, where the 16-bit round keys will be used in halves for the encryption and decryption of Shuffle-RAT. 'SRATn' denotes the stages of Shuffle RAT, where 'n' is the block size, for example SRAT2 means Shuffle-RAT stage with 2-bit block size.

Table I. The Shuffle-RAT Encryption

Plaintext	1011110111000111

Round 1 (Block size = 2 bits)	10 01 00 01 00 00 01 00
	01 10 01 00 00 00 00 01
	01 11 00 00 00 00 00 01

Next Input	0111 0000 0000 0001

Round 2 (Block size = 4 bits)	0111 0111 0111 1000
	0111 0111 1000 0111
	0111 1110 0110 1101

Next Input	01111110 01101101

Round 3 (Block size = 8 bits)	01111110 11101011
	11101011 01111110
	11101011 01101001

Cipher text	1110101101101001

Table II. The Shuffle-RAT Key Generation

Round	*Block Size*	*Number of Iterations*	
		Decimal	*Binary*
1	2	52034	1100101101000010
2	4	11025	0010101100010001
3	8	32541	0111111100011101
4	16	31100	0111100101111100
5	32	1020	0000001111111100
6	64	42167	1010010010110111
7	128	995	0000001111100011
8	256	10056	0010011101001000

The binary strings are concatenated together to form a 128-bit binary string for the final Shuffle-RAT key:

1100101101000010 0010101100010001
0111111100011101 0111100101111100
0000001111111100 1010010010110111
0000001111100011 0010011101001000

This 128-bit binary string will be the encryption-key for this particular session. During decryption, the same key is taken but in reverse order to iterate each round of modulo-subtraction for the specified number of times define by the round keys.

III. IMPLEMENTATION OF SHUFFLE-RAT

In this section, the hardware architecture and VHDL description for the FPGA implementation [4] of Shuffle-RAT is described. Before laying out the architectural plan for our proposed cipher, let us take note of all components that we will be required to use in this context:

- *Storage:* The plaintext is stored in a 512 bit, which is a 64-byte register array denoted by 'regbox'. The key is stored in a 128 bit, which is a 16-byte register array denoted by 'keymod'. The masks for two rounds are stored in a 10-byte register array denoted by 'mskbox'.
- *Logic blocks:* This consists of the main controller module denoted by 'srat_main', the individual circuits for 8 rounds of Shuffle-RAT (SRAT2 to SRAT256), and the access logic and multiplexing circuit to read and write from the storage.

Figure 1 shows the design of Shuffle-RAT for FPGA simulation. The plaintext, keys are taken from storage registers named 'regbox' and 'keymod'. The complete Shuffle-RAT operations are done in SRAT2 to SRAT256, as described in the algorithm. Clock and reset inputs are fed to the main controller module, which instructs the SRAT modules to operate in a particular sequence, and indicates when all operations are completed successfully.

The registers are byte array (8 bit) for the storage 'regbox', whereas SRAT2 and SRAT4 require the access of 2-bit and 4-bit blocks respectively. This is why the masks are stored in 'mskbox' to access the required 2-bit or 4-bit blocks from the bytes. Another main point in terms of an efficient design is that the blocks SRAT2 to SRAT256 operate sequentially, and do not overlap in time. Thus, the access ports to the storage modules, that are 'regbox', 'keymod' and 'mskbox', can be shared among the SRAT operations. So, the port sharing logic between rounds of Shuffle-RAT is incorporated.

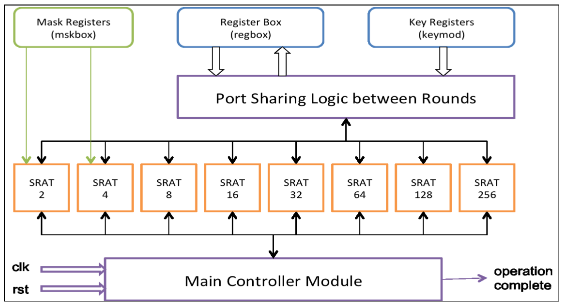

Figure 1. Top-level Hardware Architecture for Shuffle-RAT.

The main storage for the Shuffle-RAT hardware is the 'regbox' array and the 'keymod' array. The 'regbox' comprises of 8 bit registers made of edge-triggered master-slave flip-flops, with a total of 64 such registers to hold the 512-bit plaintext. To accommodate the read and write accesses to the 'regbox', we use write-access decoders and read-access decoders, which in turn control 64-to-1 multiplexer units associated to each location of the array. The 'keymod' that holds the 128-bit Shuffle-RAT key is also designed in a similar fashion, but with the exception that no intermediate write accesses are required for the registers.

IV. Results and Comparison

This section gives the various results and their comparisons with widely and industrially accepted RSA and TDES. The Sub-section A deals with the software implementation issues and the Sub-Section B deals with the hardware implementation issues.

A. *Software Implementation*

First implementation of Shuffle-RAT was using C programming language, compiled and executed in GCC on Ubuntu Linux 10.04 platform, for comparing with existing designs. Then analyzed the proposed design of Shuffle-RAT according to the testing parameters of time-efficiency, chi-square, frequency distribution and avalanche effect. The comparative studies of Shuffle-RAT against RAT, RSA and T-DES are performed and also provide some theoretical justification for the improvements of Shuffle-RAT over the original RAT scheme.

Table III shows the comparison of Shuffle-RAT with three existing ciphers in terms of avalanche test. Note that the chi-square value of Shuffle-RAT in all the above cases is always better than that of RAT, RSA and T-DES. Table IV gives the Chi-Square value. This shows that randomness and non-homogeneity of Shuffle-RAT is better than RAT. Figure 2 provides a graphical view.

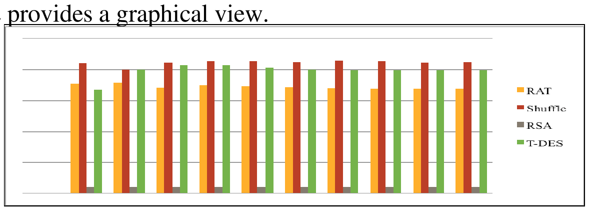

Figure 2. Comparison of Chi-Square Values.

The result of chi-square for non-homogeneity of the cipher text is supported by the flatness of the frequency distribution in case of Shuffle-RAT, shown in Figure 3.

Shuffle-RAT is much better compared to RAT in terms of diffusion. This is verified in the avalanche test. The comparative results for Shuffle-RAT and RAT are shown in table III, which clearly shows the improvement.

Block ciphers are symmetric key systems that are most often used for their better efficiency for low computational complexity over public key ciphers. Thus, one main parameter for comparison is time-efficiency of encryption and decryption. The two graphs in Figure 4 show the comparison of efficiency of Shuffle-RAT with that of RAT, T-DES and RSA, for clarity, the graphs are plotted with a logarithmic y-axis scale.

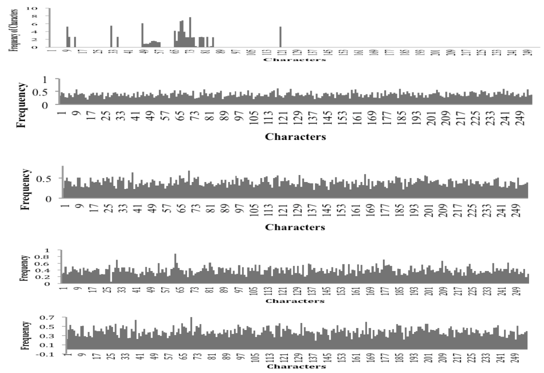

Figure 3. Frequency Distribution for Plaintext (512 KB), and corresponding Ciphertexts of Shuffle-RAT, RAT, T-DES and RSA.

Table III. Comparison of Avalanche Test

Bit changed in	*Shuffle-RAT*	*RAT*
First Byte	49.02 %	50.00 %
Middle Byte	46.68 %	26.95 %
Last Byte	46.88 %	0.58 %
Average	47.53 %	25.84 %

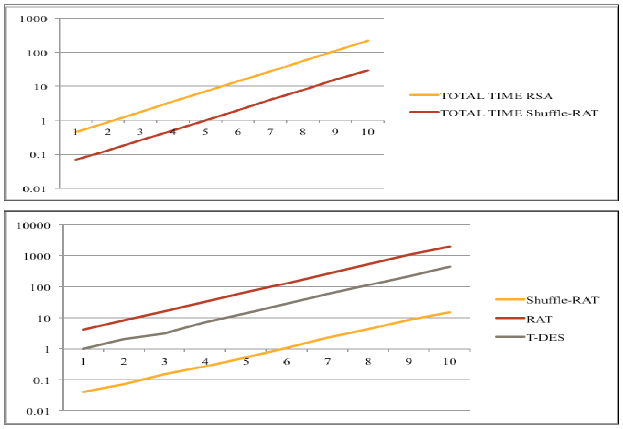

Figure 4. Efficiency Comparison of Shuffle-RAT, RAT, T-DES, RSA.

Table IV. Comparison of Chi-Square Values

Sr. No.	*Source File*	*Encrypted File*	*File Size (bytes)*	*Chi-Square value*			
				RAT	*Shuffle RAT*	*RSA*	*T-DES*
1.	OEWABLog.txt	Cipher1	833	17.762	20.994	0.998	16.715
2.	emote.dat	Cipher2	1,818	17.886	20.050	0.999	20.007
3.	lang.dat	Cipher4	4,557	17.021	21.087	0.999	20.691
4.	NOKIA6969.jpg	Cipher8	7,719	17.349	21.268	0.999	20.657
5.	EasthamBio.pdf	Cipher16	16,337	17.231	21.268	0.999	20.335
6.	BASM.doc	Cipher32	32,376	17.078	21.169	0.999	20.027
7.	peanuts.jpg	Cipher64	65,495	16.978	21.341	0.999	19.920
8.	lec13.pdf	Cipher128	131,461	16.909	21.304	0.999	19.881
9.	Espanol.txt	Cipher256	262,144	16.853	21.121	0.999	19.923
10	CP950.txt	Cipher512	522,816	16.910	21.156	0.999	19.925

Main aspects to notice while computing the runtime of Shuffle-RAT is that the key-size for the RAT iterations is 8 bits each, and there is only a single round of shuffle each round. Thus, if we consider the shuffle comparable to 1 RAT, then the runtime for Shuffle-RAT becomes equivalent to the order of

$$8 * (2^8 + 1 + 2^8) = 2^{12} \text{ RAT rounds}$$

In the discussion of RAT it was shown that the runtime for RAT (with 16 bit keys at each level) is of the order of

$$8 * 2^{16} = 2^{19} \text{ RAT rounds}$$

Hence, we get a speed-up of approximately 219/212 = 27, i.e., of the order of 100. This is also evident from the experiments.

B. Hardware Simulation

We described the hardware architecture of Shuffle-RAT using VHDL [7], and performed the simulation on FPGA using the Xilinx ISE toolkit [9]. The RTL schematic for the main controller module is shown in Figure 5, and Figure 6 shows the top-level module of the architecture.

The net-list data generated for the complete architecture of Shuffle-RAT is given in Table V. This summarizes the main list of hardware components required to synthesize the design.

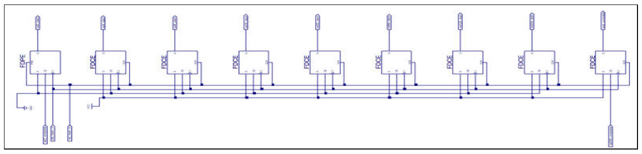

Figure 5. RTL Schematic of the Main Controller Module.

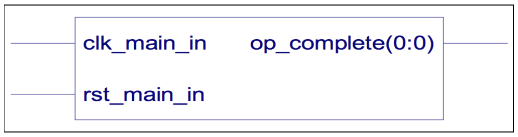

Figure 6. RTL Schematic of the Top Level Shuffle-RAT Block.

Table V. Net-List of Shuffle-RAT

Component	*Count*
ROMs	2
Adder/Subtrators	8
Registers	641
Latches	80
Multiplexes	136

V. Conclusion

In this paper, an efficient iterated block cipher Shuffle-RAT based on an existing design of rotational addition technique (RAT) with a novel inclusion of butterfly-shuffle in the process has been proposed. Detailed analysis of the new cipher based on relevant cryptographic properties, and comparison with existing well-known designs, including the original RAT has also been done. Also efficient hardware architecture for Shuffle-RAT implementation on FPGA has been designed, and has tested for the feasibility of the design using VHDL description, simulated using Xilinx ISE. The natural step for future work would be to exploit the advantages of Shuffle-RAT through its practical implementation and synthesis on FPGA or ASIC platforms.

VI. Acknowledgment

The authors express a deep sense of gratitude to the Department of Computer Science and Engineering, Netaji Subhas Engineering College, Kolkata, India and Department of Computer Science and Engineering, University of Kalyani, Kalyani, India.

VII. References

[1] R. Chakraborty and J. K. Mandal. "A Microprocessor-based Block Cipher through Rotational Addition Technique (RAT)". In Proceedings of ICIT, 2006.

[2] J. Fry and M. Langhammer, "RSA and Public Key Cryptography in FPGAs". Technical Report. Altera Corporation, USA.

[3] C. Chitu, D. Chien, C. Chien, I. Verbauwhede, and F. Chang. "A Hardware Implementation in FPGA of the Rijndael Algorithm". *Technical Report*, University of California, Los Angeles, 2002.

[4] T. Wollinger, J. Guajardo, and C. Paar. "Security on FPGAs: State-of-the-art Implementations and Attacks". ACM Sp. Issue Security and Embedded Systems, 2003.

[5] W. Stallings. *"Cryptography and Network Security"*. Prentice Hall, 2005.

[6] W. H. Wolf. *"FPGA Based System Design"*. Prentice Hall, 2009.

[7] J. Bhasker. *"A VHDL Primer"*. Prentice Hall, 1994.

[8] *Wikipedia: The Free Encyclopedia*. Wikimedia Foundation Inc. [http://en.wikipedia.org/]

[9] *Xilinx ISE Toolkit and Tutorial.* The Xilinx Foundation.