FPGA Based Cipher Design & Implementation of Recursive Oriented Block Arithmetic and Substitution Technique (ROBAST)

Rajdeep Chakraborty
Dept. of Computer Science & Engineering,
Netaji Subhash Engineering College, Garia, Kolkata-700152,
West Bengal, India.

JK Mandal, Professor,
Dept. of Computer Science & Engineering,
University of Kalyani, Nodia, West Bengal, India.

Abstract - **Proposed FPGA based technique considers a message as a binary string on which ROBAST is applied. A block of n-bits is taken as an input stream, where n ranges from 8 to 256 – bit, then ROBAST 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 – VHDL; FPGA; RTL; Block Cipher; Session key and Private Key; Cryptography; Symmetric/Private key cryptosystem.

I. INTRODUCTION

Transmission of sensitive electronic information [7] from and all around the globe has emphasizes the need of fast & secure network [2,3,5]. For achieving this secrecy, integrity and confidentiality, cryptographic techniques [1,2,3,14] are the tools. To achieve high performance it is highly recommended to implement the cryptographic techniques in hardware. A promising solution that combines high flexibility with the speed and physical security of Application Specific Integrated Circuits (ASIC) [7,8,9,10] is the implementation of cryptographic technique on state-of-the-art re-configurable devices such as Field Programmable Gate Array (FPGA) [7,8,9,10]. Sub-Section A, discussed the framework of the scheme along with private/symmetric key cryptosystem [4,7].

Private Key / Symmetric Cryptosystem

The aim is to develop an efficient crypto hardware. The figure 1 illustrates the conventional encryption model [2,7]. The main objective is to convert the intelligent plain text [2,3,4,5,7], X, to a non-sense cipher text [2,3,4,5,7], Y, using a single key [2,3,4,5,7], K. This process is the encoding/encryption [1,2,14], E, and the decoding/decryption [1,2,14], D, is performed similarly in case of symmetric or the opposite in other private key algorithms. The Session Key [1,2,14], K must be sent through a secured channel and cipher text, Y, may be sent through unsecured channel. Cryptanalyst [1,2,7,14] are the entity who attempts to discover plain text, X, and or key, K.

The Section II illustrates the principle of ROBAST, Section III gives the key generation process, result and simulations are given in Section IV, A brief analysis is given in Section V, Section VI draws the conclusion and finally the list of the references are given.

II. PRINCIPLE OF ROBAST

The message can be considered as blocks of bits with different block size [1,7,14] like 8, 16, 32, 64, 128 & 256 bits. The rules to be followed for generating a cycle are as follows:

1. Consider any source stream [1,2,7,14] containing finite number of bits (where $N=2^n$, n =3 to 8) and divide it into two equal parts.
2. Make the source stream into paired form so that a pair can be used for the operation.
3. Perform the modulo-4 addition [6,13] between the first and second pair, second and third pair, and so on of the source stream, to obtain the first intermediate block.
4. The same process is repeated recursively [7,11,12] between second and first, third and second, fourth and third and so on of the source stream, to generate the next intermediate block.

This process is repeated until the source stream is generated. After a finite number of iterations source stream is regenerated. So, decryption is basically the iteration of the same process. In this proposed technique the modulo addition with substitution and permutation is given but to enhance the security further other arithmetic operations has also been implemented in this technique. Sub-Section A illustrates the scheme numerically and that of Sub-Section B outlines the implementation issues.

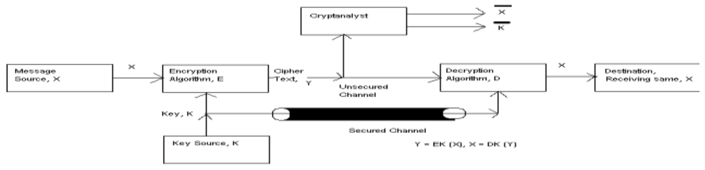

Figure 1: Model of Conventional Cryptosystem

A. *Example*

Consider the block S = 10010011 of size 8 bits. The Flow diagram to show how positions of the bits of S and the different intermediate blocks can be reoriented with the key values to complete the cycle is shown in figure 2. In this diagram, each arrow indicates positional orientation of a bit during iteration. Therefore the final cipher text is S'=00011001.

B. *Implementation of ROBAST*

The technique executes modulo addition between two blocks, the first iteration performs in forward basis and then backward operation is performed. Next, final permutation is done to get the final cipher text. This technique has been implemented in C [11,12,13] and then feasibility study has been performed. Finally, FPGA [8,9,10] based implementation has been done in VHDL [8,9,10]. In both implementation, the technique takes input from file as a source stream and encryption is performed. The cipher text generated is finally written in another file [7,10,11,12]. The data blocks (8, 16, 32, 64, 128 and 256-bits) from the input file have been stored in array. Then encryption is performed and also stored in array. The reading and writing of data from and in file is based on 8-bit ASCII codes [7,11,12]. XilinX [8,9,10] software has been used for writing codes in VHDL. The encryption/decryption entity input bit vector (16-bit), output bit vector (16-bit), key bit vector (8-bit) and EN_DN signal. If EN_DN = 1 then encryption is performed else decryption is performed. Figure 3 gives the main ROBAST entity coded in VHDL.

The above operations discussed are substitute technique [6,7,13] followed by permutation technique [6,7,13] has been performed by orientation of bits based on the session key. Therefore, these resultant blocks of stream can be considered as cipher text.

III. The Key Generation Process

The key generation process depends on block size, iteration of each block and final permutation performed. Thus, in the proposed scheme, eight rounds have been considered, each for 2, 4, 8, 16, 32, 64, 128, and 256-block size. As mentioned in Section II, 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 key for a particular key. Example in Sub-Section A illustrates the key generation process. Sub-Section B describes the modulo addition, which is an important operation in the technique and should be taken into account while forming the session key.

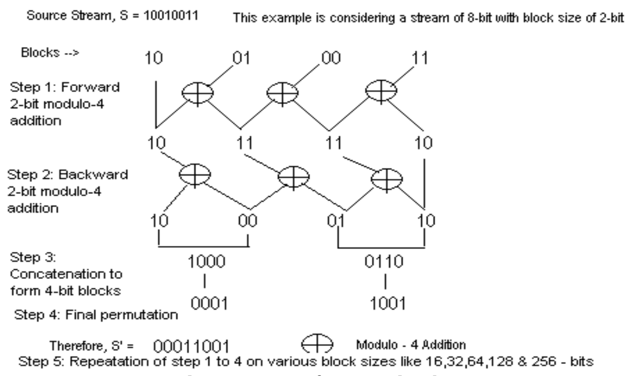
Source Stream, S = 10010011
This example is considering a stream of 8-bit with block size of 2-bit
Blocks -->
10
01
00
11
Step 1: Forward 2-bit modulo-4 addition
10
11
11
10
Step 2: Backward 2-bit modulo-4 addition
10
00
01
10
Step 3: Concatenation to form 4-bit blocks
1000
0110
Step 4: Final permutation
0001
1001
Therefore, S' = 00011001
Modulo - 4 Addition
Step 5: Repeatation of step 1 to 4 on various block sizes like 16,32,64,128 & 256 - bits

Figure 2: Flow Diagram of proposed technique, ROBAST

```
library std;
library ieee;
use ieee.std_logic_arith.all;
use work.pack.all;
use std.textio.all;
use ieee.std_logic_TEXTIO.all;
entity ROBAST_VHDL is

Port ( input_bits : in  BIT_VECTOR (16 downto 1);
output_bits : out  BIT_VECTOR (16 downto 1);
key_bits : in  BIT_VECTOR (8 downto 1);
EN_DN : in  BIT);
end ROBAST_VHDL;

architecture Behavioral of ROBAST_VHDL is

begin

process(EN_DN)

variable varin_bits,varout_bits: bit_vector(16 downto 1);

begin

if (EN_DN='1')then
varin_bits:=input_bits;
AA: ROBAST_Encryption(varin_bits,key_bits,varout_bits);
output_bits<=varout_bits;
else
BB: ROBAST_Decryption(varin_bits,key_bits,varout_bits);
output_bits<=varout_bits;
end if;

end process;

end Behavioral;
```

Figure 3: ROBAST Entity and its function

A. *Example*

Consider a particular session where the source file is encrypted using iterations for block sizes 2, 4, 8, 16, 32, 64, 128, and 256 bits, respectively. Table I shows the corresponding binary value [7,8,10,13] for the number of iterations in each round. The binary strings are concatenated together to form the 128-bit binary string:

11000011011001011100001011001110101111100110110
10101101100110111011010010101010000100001110000100
000010101100100000000001001000.

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

B. Modulo Addition

An alternative method for modulo addition is proposed here to make the calculations simple. The need for computation of decimal equivalents of the blocks is avoided here since it may generate large decimal integer values for large binary blocks. The method proposed here is just to discard the carry out of the MSB after the addition to get the result. For example, if we add 1101 and 1001 we get 10110. In terms of decimal values, 13+9=22. Since the modulus of addition is 16 (24) in this case, the result of addition should be 6 (22-16=6). Discarding the carry from 10110 is equivalent to subtracting 10000 (i.e. 16 in decimal). So the result will be 0110, which is equivalent to 6 in decimal. The same is applicable to any block size.

IV. Results and Simulation

Any cryptographic technique is to be accepted, a satisfactory results are very much required. Proposed technique has been tested for feasibility both in terms of algorithmic parameters and cryptographic parameters. Sub-Section A gives the time complexity results [7,11,12,13], Sub-Section B tests for non-homogeneity using Chi-Square values [1,6,14,15] and degree of freedom [1,6,14,15], Sub-Section C illustrates the frequency distribution [1,6,14,15] of ASCII characters [7,11,12], Sub-Section D test for cryptanalysis using avalanche ratio [2,3,4,5] and finally Sub-Section E gives the FPGA-based simulation results [8,9,10]. All these results are against well known and industrially accepted RSA and TDES [1,2,3,4,13,14]. For the shake of brevity 20 (twenty) sample files of different types has been taken for these results. The Section V briefly analyses all these results.

A. Time Complexity

Time complexity is based on encryption time and decryption time [1,2,14]. Encryption time is the time required to encrypt a source file and decryption time is the time to decrypt the cipher text file to get the original file. Table II gives the time complexities and Figure 4 illustrates the same. This test is in terms of efficient algorithmic parameter.

B. Tests for Non-Homogeneity

Test for non-homogeneity has also been done using Chi-Square value and degree of freedom; this is one of the important cryptographic parameters. Chi square value is the statistical value between source file and encrypted files, which gives the difference. Degree of freedom in the character distribution of the above said files. Table III gives the Chi-Square value and Figure 5 illustrates the same.

C. Frequency Distribution

The frequency distribution is the distribution of the all-256 ASCII characters in the respective files. This is also a cryptographic parameter, which measures the degree of cryptanalysis. Figure 6 illustrates the various frequency distribution results found after implementation of respective algorithms/techniques.

D. Avalanche Ratio

The avalanche ratio is the ratio between the modified results to the original result. The avalanche ratio is obtained by modifying 2-3 bits/bytes in the encryption key as well as in source files. It's a strong cryptographic parameter and this may be conceptualize with the avalanche occurs in hill area. Table IV gives the avalanche ratio values of ROBAST.

TABLE I. Representation of Number of Iterations in Each Round by Bits

Round	Block Size	Number of Iterations	
		Decimal	Binary
8.	256	50021	1100001101100101
7.	128	49870	1100001011001110
6.	64	48950	1011111100110110
5.	32	44443	1010110110011011
4.	16	46250	1011010010101010
3.	8	4321	0001000011100001
2.	4	690	0000001010110010
1.	2	72	0000000001001000

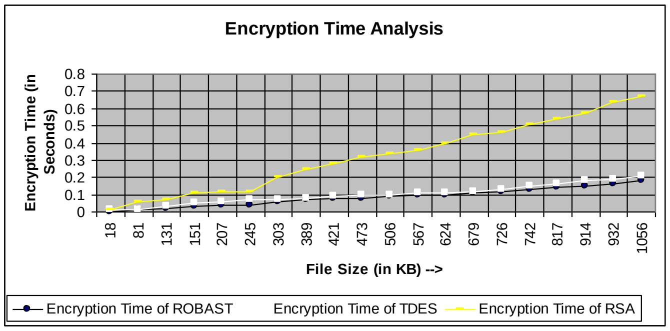

Figure 4 (A): Encryption Time

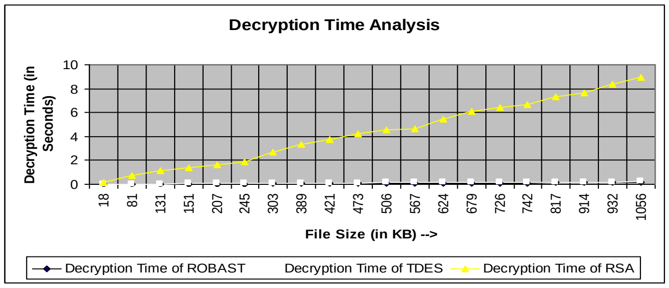

Figure 4 (B): Decryption Time

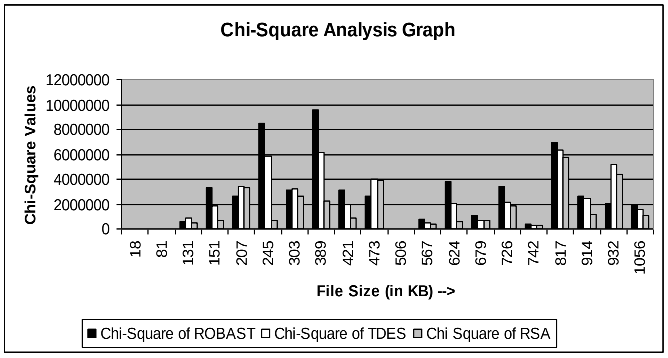

Figure 5: Chi-Square Analysis Graph

TABLE II. TIME COMPLEXITY ANALYSIS

Serial no.	File Name	File Size (in Kilo Bytes)	Encrypted time (in second)			Decryption time (in second)		
			ROBAST	TDES	RSA	ROBAST	TDES	RSA
01	Poppy.jpg	18	0.00	0.01	0.01	0.01	0.02	0.15
02	07.jpg	81	0.01	0.01	0.06	0.02	0.03	0.71
03	Sqmapi.dll	131	0.02	0.03	0.07	0.03	0.03	1.15
04	Jview.exe	151	0.03	0.05	0.11	0.03	0.06	1.36
05	Gender.txt	207	0.04	0.06	0.12	0.04	0.07	1.61
06	Pod.exe	245	0.04	0.07	0.12	0.04	0.08	1.86
07	Devices.txt	303	0.06	0.07	0.20	0.05	0.09	2.71
08	Dtliteui.dll	389	0.07	0.08	0.25	0.06	0.10	3.34
09	Vssapi.dll	421	0.08	0.09	0.28	0.07	0.11	3.73
10	Names.txt	473	0.08	0.10	0.32	0.08	0.11	4.25
11	Photo000.jpg	506	0.09	0.10	0.34	0.08	0.13	4.54
12	Uninst.exe	567	0.10	0.11	0.36	0.10	0.14	4.67
13	Iexplore.exe	624	0.10	0.11	0.40	0.11	0.14	5.43
14	Cordic.pdf	679	0.11	0.12	0.45	0.11	0.16	6.10
15	Iedvtool.dll	726	0.12	0.13	0.46	0.12	0.17	6.40
16	Guide.pdf	742	0.13	0.15	0.51	0.12	0.18	6.67
17	Setuplog.txt	817	0.14	0.16	0.54	0.14	0.18	7.34
18	Adobearm.exe	914	0.15	0.18	0.57	0.15	0.19	7.68
19	De10.txt	932	0.16	0.19	0.64	0.16	0.20	8.39
20	Omat.doc	1056	0.18	0.21	0.67	0.19	0.23	8.92

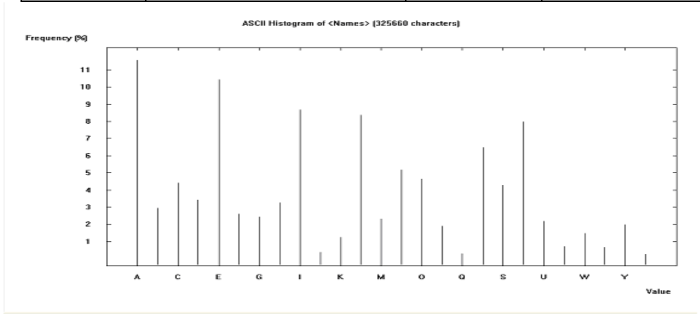

Figure 6 (A): Frequency distribution of Source File

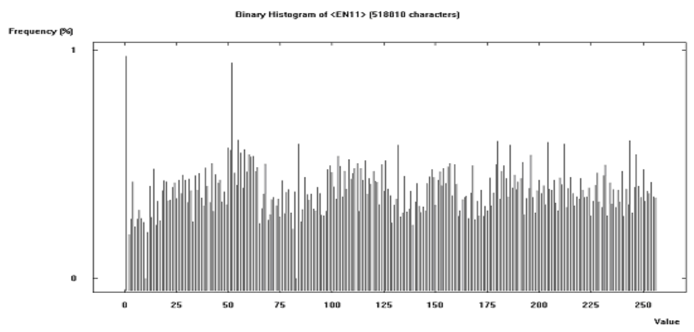

Figure 6 (B): Frequency distribution of ROBAST encrypted file

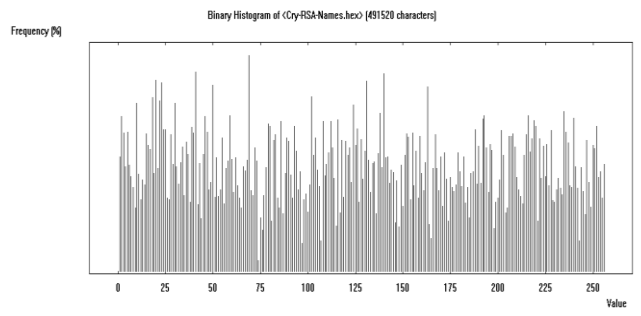

Figure 6 (C): Frequency Distribution of RSA encrypted file

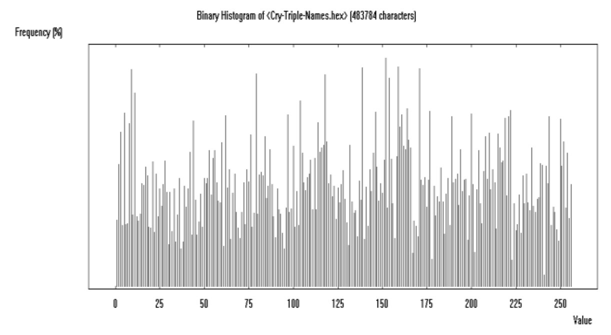

Figure 6 (D): Frequency distribution of Triple-DES encrypted file

E. *FPGA-Based Simulation Result*

This Section gives some of the results found after implementing the proposed technique in VHDL. This code has been simulated and synthesized in XilinX. The main objective is to find an efficient FPGA-based cryptographic technique for implementation in embedded systems. The Figure 7 gives the RTL schematic [8,9,10] of the proposed technique and the Figure 8 gives the chip diagram for Spartan 3E [8,9,10].

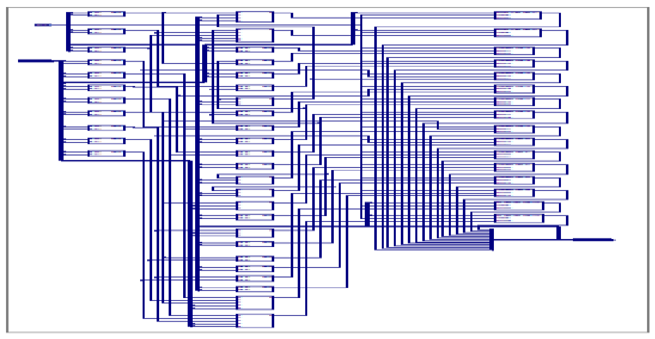

Figure 7: RTL Schematic

TABLE III. CHI-SQUARE AND DEGREE OF FREEDOM VALUES

Serial no.	File Name	File Size (in Kilo Bytes)	Chi-Square Value			Degree of Freedom		
			ROBAST	TDES	RSA	ROBAST	TDES	RSA
01	Poppy.jpg	18	6472	4572	5668	253	255	255
02	07.jpg	81	4407	2943	2654	253	255	255
03	Sqmapi.dll	131	560357	890752	447984	253	255	255
04	Jview.exe	151	3307374	1847893	685963	253	255	255
05	Gender.txt	207	2679799	3426290	3318506	253	93	84
06	Pod.exe	245	8495675	5810912	694410	253	255	255
07	Devices.txt	303	3131296	3220112	2667664	254	66	88
08	Dtliteui.dll	389	9559993	6190253	2216429	253	255	255
09	Vssapi.dll	421	3102369	1980059	906300	253	255	255
10	Names.txt	473	2590855	4044603	3896171	253	88	90
11	Photo000.jpg	506	38465	31719	30353	253	255	255
12	Uninst.exe	567	776122	512668	342450	253	255	255
13	Iexplore.exe	624	3799155	2043250	588049	253	255	255
14	Cordic.pdf	679	1065255	684198	686392	253	255	255
15	Iedvtool.dll	726	3422000	2192824	1845040	253	255	255
16	Guide.pdf	742	420469	320825	311524	253	255	255
17	Setuplog.txt	817	6904009	6340148	5737525	253	255	255
18	Adobearm.exe	914	2625926	2458497	1196585	253	255	255
19	De10.txt	932	2043522	5194261	4407281	251	86	11
20	Omat.doc	1056	1968558	1516848	1082800	253	255	255

TABLE IV. AVALANCHE RATIO OF ROBAST ENCRYPTED FILES

Serial no.	File Name	File Size (in Kilo Bytes)	Avalanche Ratio of ROBAST encrypted files (in %)
01	Poppy.jpg	18	96.25
02	07.jpg	81	99.69
03	Sqmapi.dll	131	99.93
04	Jview.exe	151	99.96
05	Gender.txt	207	97.72
06	Pod.exe	245	99.84
07	Devices.txt	303	98.22
08	Dtliteui.dll	389	99.97
09	Vssapi.dll	421	99.98
10	Names.txt	473	99.95
11	Photo000.jpg	506	99.91
12	Uninst.exe	567	99.98
13	Iexplore.exe	624	99.99
14	Cordic.pdf	679	99.70
15	Iedvtool.dll	726	99.97
16	Guide.pdf	742	99.56
17	Setuplog.txt	817	99.56
18	Adobearm.exe	914	99.98
19	De10.txt	932	06.83
20	Omat.doc	1056	99.73
Average Avalanche Ratio			94.84
The avalanche ratio is obtained by modifying 2-3 bits/bytes in the encryption key as well as source files			

V. ANALYSIS OF THE RESULTS

Analyzing all the results presented in the result Section(s), following are the points obtained on the proposed technique:

1. The encryption time and decryption time varies linearly with the file sizes. Also the time complexity of ROBAST is quite less than RSA, but it's slight less than TDES.
2. Considering the Chi-Square values, the proposed technique, ROBAST, is most non-homogeneous than that of RSA and TDES. But, there is no substantial result found in terms of degree of freedom because all three (ROBAST, RSA, TDES) have almost same value.
3. Result for the frequency distribution illustrates the ASCII characters are well distributed in ROBAST. The well distribution was also found for RSA and TDES. So, the frequency distribution result is at par with that found in Chi-Square and degree of freedom values.
4. A very good result has been obtained in the avalanche ratio of the proposed technique. The average avalanche ratio is 94.84. So, cryptanalysis is quite difficult.
5. The RTL diagram signifies that the proposed technique has been successfully implemented in VHDL and the same is illustrated for Spartan 3E FPGA. If we closely look at, there are 29 Look-Up-Tables (LUT s) [8,9] used for this technique.

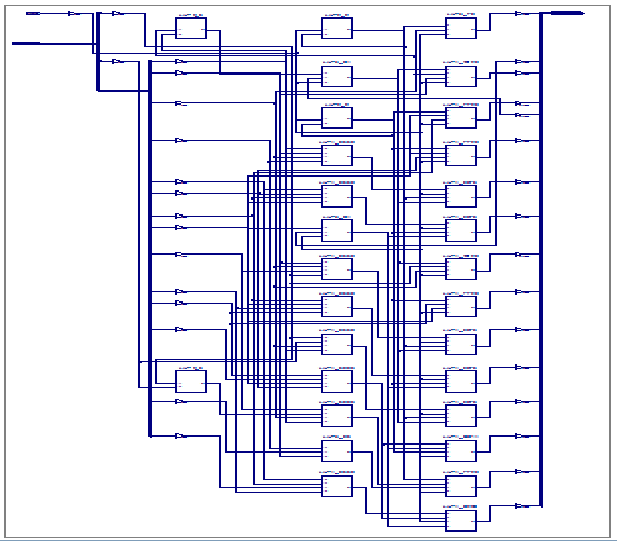

Figure 8: Spartan 3E Schematic

Sub-Section A. gives the application of this proposed technique, ROBAST and along with future scope of work.

A. Application of ROBAST

The following are some of the avenues of application for the proposed technique, ROBAST: -

1. Since the proposed system is simple, fast and low power consumption based crypto solution, it can be used in various embedded systems.
2. This technique may be used to develop electronic codebooks.
3. It's can also be used to develop a private network and master-key-based applications.
4. This proposed FPGA-based system may be used in hardware applications such as switch, gateways and routers.

The FPGA implementation of Vertex series with increased block length and also with low computational complexity is the future scope of the work.

VI. Conclusion

The proposed technique given here is easily implemented in high-level language and in VHDL. This technique is very easy and it's implemented in FPGA-based systems, the goal of fast execution and strong cryptanalysis requirements are also obtained here. This technique can be fabricated in chip to be used in embedded systems. The main goal of the author(s) is to develop an efficient FPGA-based crypto hardware and this paper is the first step towards this.

Acknowledgment

The authors express their deep sense of gratitude to The Department of Computer Science and Engineering, Netaji Subhash Engineering College, Garia, Kolkata, West Bengal, India and The Department of Computer Science and Engineering, University of Kalyani, Kalyani, Nodia, West Bengal, India.

References

[1] Rajdeep Chakraborty, Dr. J.K.Mandal, "A Microprocessor-based Block Cipher through Rotational Addition Technique (RAT)", ICIT – 2006 18-21 December, 2006, Bhubaneswar, INDIA.

[2] W. Stallings, Cryptography and Network Security: Principles and Practices, Prentice Hall, Upper Saddle River, New Jersey, USA, Third Edition, 2003.

[3] B. Schneier. Applied Cryptography. John Wiley & Sons Inc., New York, New York, USA, 2nd edition,1996.

[4] U.S. Department of Commerce/National Institute of Standard and Technology. FIPS PUB 197, Specification for the Advanced Encryption Standard (AES), November 2001. Available at http://csrc.nist.gov/encryption/aes.

[5] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. Handbook of Applied Cryptography. CRC Press, Boca Raton, Florida, USA, 1997.

[6] A.M. Goon, M.K. Gupta, B. Dasgupta, Fundamentals of Statistics, Vol. 1, The World press Ltd.

[7] Dictionary of Computers and Information Technology Terms, 1st edition, low point, Kolkata, India.

[8] FPGA- Based System Design by W. Wolf, Pearson Education.

[9] Embedded Core Design with FPGA's by Z. Navavi, TMGH.

[10] AVHDL: Premier by J. Bhasker, Pearson Education

[11] Programming in C by Balaguruswamy, India.

[12] Pointers in C by Y Kanitkar, India

[13] The software cryptographic tools for educational purpose available at http://www.cryptool.com/

[14] S. Mal, J.K. Mandal, S. Dutta, "A Microprocessor Based Encoder for Secured Transmission", Proceedings of the National Conference on Intelligent Computing on VLSI, Kalyani Govt. Engg. College, 16-17 February, 2001, pp 164-169.

[15] Number theory home page for secured key generations http://www.numbertheory.org/ntw/web.html

[16] Pasha, A., & Gafoor, A. (2011). Transparent Data Encryption- Solution for Security of Database Contents. *International Journal of Advanced Computer Science and Applications - IJACSA*, *2*(3), 25-28.

[17] Meshram, C. (2010). Modified ID-Based Public key Cryptosystem using Double Discrete Logarithm Problem. *International Journal of Advanced Computer Science and Applications - IJACSA*, *1*(6).

[18] Nath, J. (2011). Advanced Steganography Algorithm using Encrypted secret message. *International Journal of Advanced Computer Science and Applications - IJACSA, 2*(3).

AUTHORS PROFILE

RAJDEEP CHAKRABORTY, born on 23rd August ' 1978, did his Bachelor of Engineering (B.E.) in Computer Science and Engineering (CSE) from Utkal University, Bhubaneswar, India at 2002, then he did his Master of Technology (M. Tech) in Information Technology (IT) from Sikkim Manipal University of Health Medical and Technological Sciences (SMUHMTS), Gangtok, India at 2004, he is presently persuing PhD in Computer Science and Engineeing (CSE) from University of Kalyani, Kalyani, India, in the field of Cryptography. Presently he is assistant professor in the department of Computer Science and Engineering (CSE), Netaji Subhash Engineering College, Kolkata, India. He has almost 6 years of teaching and research expirence and the total number of his publication is eight, all in internatinal conferences and journals. The average makrs throughout his carrer is 71.0%.

JK MANDAL did his M. Tech in Computer Science & Engineering at University of Calcutta, then did his PhD in CSE, at Jadavpur University in the field of Data Compression and Error Correction techniques. Presently he is professor in Computer Science and Engineering department, University of Kalyani, India. He is also a life member of Computer Society of India (CSI) since 1992 and life member of Cryptology Research Society India. Moreover, he is also serving as a dean, faculty of Engineering, Technology and Managemat, at University of Kalyani. The fields of his work are Network Security, Steganography, Remote Sensing, GIS application and Image Processing. He has 23 years of Teaching and Research Experience and seven scholars are awarded PhD and now eight scholars are pursuing PhD under his guidance. The total number of his publication is 140.