Implementation of Simplified AES algorithm for Wireless Sensor Nodes on FPGA

Kavan A N & Premananda B S

Abstract— A Wireless Sensor Networks (WSN) is an ad-hoc wireless network made of sensor nodes that are physically small, communicate wirelessly among each other. When sensor nodes are used in security domains, it requires security architectures to protect its resources. Rijndael or Advanced Encryption Standard (AES) algorithm is a well-known standard algorithm for encryption. The encryption algorithm running on such sensor nodes have two main constraints: memory and processing speed. This paper proposes S-AES to overcome the above two constraints and make it suitable for sensor nodes. The Zigbee CC2500 transceiver module is used as the sensor nodes. The proposed Simplified AES encryption and decryption is coded in Verilog HDL, simulated using Xilinx ISESIM tool and synthesized using Xilinx XST. The implementation and validation are done on Spartan 3A FPGA.

Index Terms— FPGA, S-AES, WSN, UART, Zigbee CC2500.

1 INTRODUCTION

AES plays a significant role of a secure algorithm for data exchanged. Two prime processes are involved: Encryption (cipher) and Decryption (decipher). An encryption algorithm provides Confidentiality, Authentication, Integrity and Non-repudiation. Encryption algorithms are broadly classified as Symmetric or Asymmetric algorithms based on the kind of keys used.

Sensor nodes are typically battery powered, making sensor networks highly energy constrained. Therefore, a key challenge in a wireless sensor networks is the reduction of energy consumption [2]. Sensor networks are becoming a cost-effective solution to a range of applications in critical domains. When sensor networks are used in security domains, security becomes a strong and important requirement. These applications should not only timely detect a potential risk, but should also be protected from malicious attacks such as fake messages or corrupted data [3].

AES is a cryptographic algorithm, selected for protecting the resources on sensor nodes. When encryption algorithm is applied on sensor nodes, processing speed is degraded and occupies large memory resources that make the AES implementation not suitable for WSN [1]. For this reason, the main focus is on the implementation of Simplified AES (S-AES) [7] for sensor nodes to overcome the above two constraints. S-AES algorithm is discussed in section 2.

- *Kavan A N, pursuing M.Tech. in Digital Communication Engineering, Department of Telecommunication Engineering, R. V. College of Engineering, Bangalore, India. E-mail: kavangowda31@gmail.com*
- *Mr. Premananda B S, Assistant Professor, Department of Telecommunication Engineering, R. V. College of Engineering, Bangalore, India E-mail: premanandabs@gmail.com*

The Zigbee CC2500 transceiver module is used as the sensor nodes. Two transceiver modules are used for the implementation. The CC2500 is a low-cost 2.4 GHz transceiver designed for very low-power wireless applications. It can be used to transmit and receive data at 9600 baud rates from any standard CMOS/TTL source. This module is a direct replacement for serial communication; it requires no extra hardware and no extra coding in Half Duplex mode. The circuit is intended for the 2400-2483.5 MHz ISM and Short Range Device (SRD) frequency band.

This paper is organized as follows: Proposed S-AES Algorithm, proposed block diagram, Zigbee CC2500 interfacing, Results and conclusion.

2 PROPOSED S-AES ALGORITHM

The structure of S-AES is exactly same as AES. It consists of encryption and decryption process. The differences are in the key size (16 bits), the block size (16 bits) and number of rounds is 2 in each stages. S-AES targets optimization in S-Box to 4x4 instead of 16x16 as in AES [7] to reduce memory consumption and number of rounds is reduced to 2 in each stage, to increase the processing speed. The encryption and decryption process of S-AES shown in fig. 1. is discussed below.

2.1 Encryption process

Encryption process converts Plaintext into Ciphertext. It consists of Initial round, Rounds of transformations, Final round and KeyExpansion process. Initial round consists of AddRound key. Round transformations consists of Subnibbles, ShiftRow, MixColumn and AddRound key. Final round consists of Subnibbles, ShiftRow and AddRoundKey. The KeyExpansion process is same for both encryption and decryption.

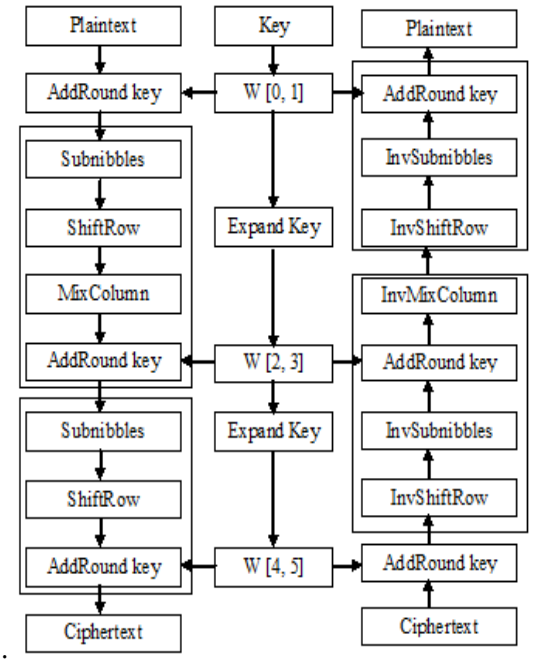

Fig. 1. S-AES Encryption and decryption process

The 16-bit Plaintext and 16-bit key are taken as the inputs for S-AES, involves following steps which are explained briefly.

***AddRoundKey*:** Instead of dividing the block into a four by four array of bytes, S-AES divides it into two by two array of "nibbles", which are four bits long. This is called the state array shown in the fig. 2. Each nibble of the state is bitwise XORed with the key which is arranged in the same manner.

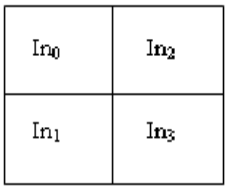

Fig. 2. State array.

Subnibbles: In this, an S-Box shown in Table 1 is used to translate each nibble of state array into a new nibble.

Table 1
S-AES S-Box

Nibbles	00	01	10	11
00	9	4	A	B
01	D	1	8	5
10	6	2	0	3
11	C	E	F	7

ShiftRow: This step is same as that of standard AES. The first Row is unchanged and the second row is cyclically shifted once shown in fig. 3.

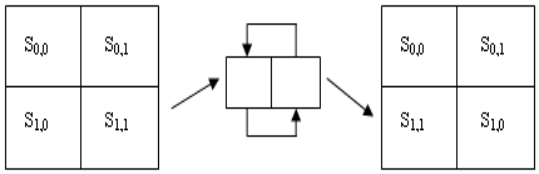

Fig. 3. Shift row transformation

MixColumn: After shifting the rows, mix the columns. Each column is multiplied by the matrix given below. The Mixcolumn transformation is shown in fig. 4.

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix}$$

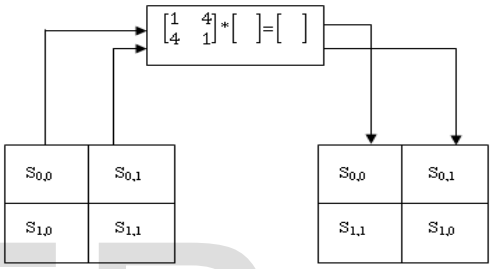

Fig. 4. MixColumn transformation

KeyExpansion: KeyExpansion is done very similarly to AES. The four nibbles in the key are grouped into two 8-bit "words", which will be expanded into 6 words.

The Generator function (g) is very similar to AES, first rotating the nibbles and then putting them through the S-Boxes. The main difference is that the round constant is produced using x^{j+2}, where j is the number of the round of expansion. For first time expanding the key, use a round constant of $x^3 = 1000$ for the first nibble and 0000 for the second nibble. For second time, use $x^4 = 0011$ for the first nibble and 0000 for the second nibble.

2.2 Decryption Process

Decryption process in fig. 1 converts Ciphertext into Plaintext. It consists of Initial round, Rounds of transformations and Final round. Initial round consists of AddRound key. Rounds of transformation consist of InvShiftRow, InvSubnibbles, AddRound key and InvMixColumn. In final round, InvMixColumn step is omitted.

AddRoundKey: Ciphertext which is obtained during the encryption process is taken as the input for decryption process. This Ciphertext is bitwise XORed with the Subkey which is obtained from KeyExpansion during encryption.

InvShiftRow: It is the inverse of the ShiftRow transformation. The first Row is unchanged and the second row is cyclically right shifted once.

InvSubnibbles: It is the inverse of the Subnibbles transformation, in which the inverse S- box is applied to each nibble of the State. The inverse S-box is presented in Table 2.

Table 2
S-AES Inv S-Box

Nibbles	00	01	10	11
00	A	5	9	B
01	1	7	8	F
10	6	0	2	3
11	C	4	D	E

InvMixColumn: In this step each column is multiplied by the matrix given as

$$\begin{bmatrix} 9 & 2 \\ 2 & 9 \end{bmatrix}$$

Table 3 shows the major differences between Existing AES and Simplified AES

Table 3
Differences between AES and S-AES

	AES	S-AES
Key length	128, 192, 256 bits	Variable length (max 128 bits)
Number of rounds/stages	10, 12, 14	2 rounds
Round key	Different for each round	Same for all rounds
Key expansion	Complex	Simplified
S-Box size	16x16	4x4
S-Box memory (Bytes)	256	8
Inv S-Box memory (Bytes)	256	8
Number of clock cycles/stages	10 (10 rounds in each stage)	2 (2 rounds in each stage)

3 PROPOSED BLOCK DIAGRAM

The conceptual block diagram is shown in the fig. 5. The Zigbee CC2500 transceiver device is used as the sensor node. The sensor monitoring system is used to monitor the sensor nodes. The personal computer (PC) is used as the monitoring system. The 16-bit plaintext and 16-bit key are passes through Zigbee devices connected to the PC. The data are received by other Zigbee device connected to the FPGA which is shown in the fig. 5. The software called TMFT v2.6 is used in the monitoring system to transmit and receive the data.

The data received undergoes encryption process in S-AES encryption/decryption core. The encrypted data is passed to the UART Tx/Zigbee through which it is passed to the monitoring system. In UART protocol, UART Tx/Rx do not share common clock signal. For data reception/transmission, both Tx and Rx should be synchronize to each other. So, the Baud rate generator is used to generate the baud rate which is commonly shared by UART Tx/Rx.

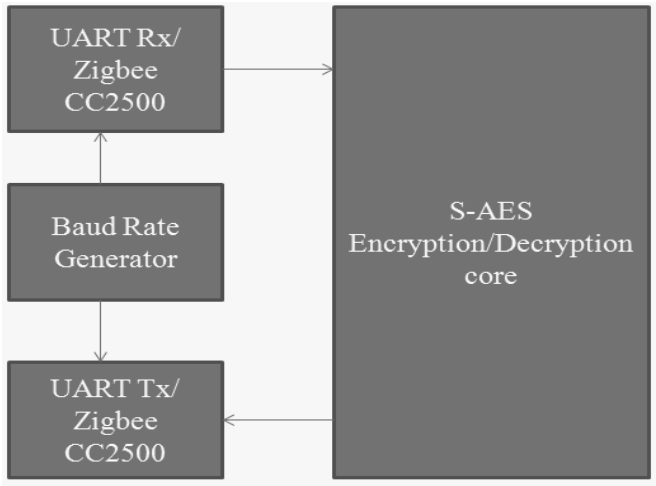

Fig. 5. Conceptual Block diagram

4 ZIGBEE CC2500 INTERFACING

The Zigbee CC2500 interfacing is based on the UART interfacing. The UART interfacing is necessary for serial communication. For serial communication, the transmitter and receiver of UART must be synchronous to each other, can be achieved by Baud-rate generator. UART interfacing is mainly depends on functional description of UART receiver, transmitter and Baud rate generator. The Zigbee CC2500 device consists of serial port (DB9 connector) which can be directly connected to any serial port (DB 9). One Zigbee device is connected to the serial port of FPGA and other Zigbee is connected to the PC. Since PC doesn't consist of serial port, USB to serial connector is used for the connection. The connections are shown in the fig. 6.

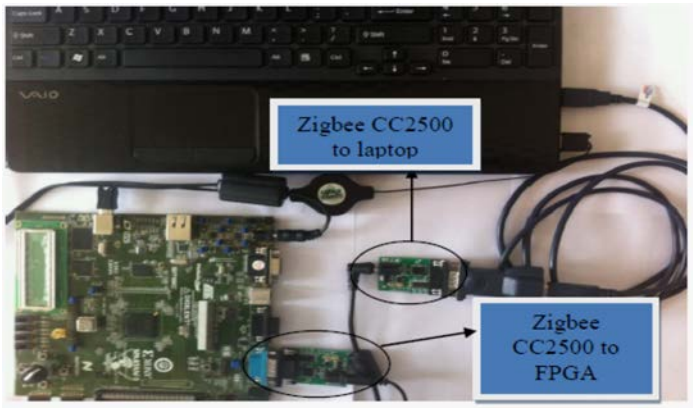

Fig. 6. Hardware setup

5 RESULTS AND DISCUSSION

5.1 Simulation results for Encryption process

The Simulation results for encryption process is shown in fig. 7 where the 16-bit Plaintext is represented as state[15:0]=0110111101101011 and 16-bit key is represented as key[15:0] =1010011100111011. The output of round 1 is out1[15:0]=1111000010000101 and final round output is denoted as out[15:0]= 0000011100111000 which is the Ciphertext.

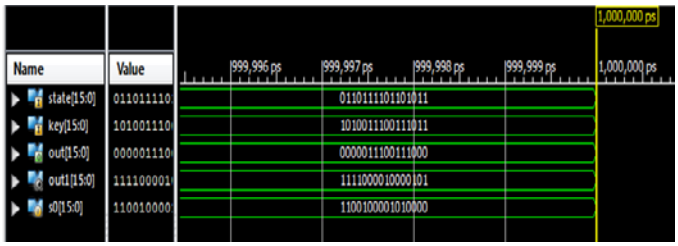

Fig. 7. Simulation results for Encryption process

5.2 Simulation results for decryption process

The simulation results for decryption process is shown in the fig. 8 where state [15:0]=0000011100111000 is the Ciphertext obtained from the encryption process and Key1[15:0]=0111011001010001 derived from KeyExpansion process are represented as inputs. The inputs undergoes decryption process to give back original Plaintext as out1[15:0]=0110111101101011.

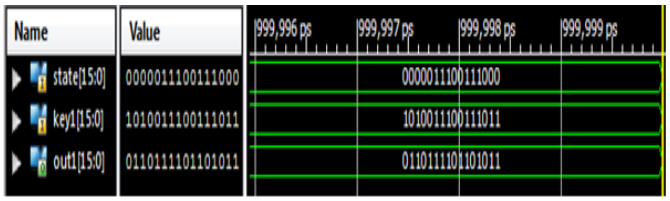

Fig. 8. Simulation result for Decryption process

5.3 Synthesis result of S-AES

A summary of synthesis results is shown in Table 4 where the area requirements (in slices) and the maximum work frequency are provided. All results are extracted after place and route with the Xilinx ISE Design Suite v13.2 on xc3s700-5fg484 Spartan-3A platform with speed grade −5. The reason for using Spartan 3A FPGA is Zigbee Interfacing will not supported by lower version FPGA devices like Spartan 3 and 3E. From Table 4, it is noted that the S-AES is utilizing 3% of available slices, 1% of flip flops, 3% of LUTs, 3% of bonded IOBs and 8% of GCLKs on target FPGA platform.

Table 4
Device utilization summary of S-AES.

Logic utilization	Used	Available	Utilization
Number of Slices	219	5888	3%
Number of Slice Flip Flops	149	11776	1%
Number of 4 input LUTs	390	11776	3%
Number of IOs	14	-	-
Number of bonded IOBs	14	372	3%
Number of GCLKs	2	24	8%

5.4 Comparison of S-AES with other Cryptographic algorithms

Table 5 shows the performance comparison of S-AES implementation with other Cryptographic algorithms AES in different platforms. The experimental results shows S-AES achieve better performance with smaller area requirement when compared to AES and humming bird cryptographic algorithms.

Table 5
Performance comparison of Cryptographic algorithms

Cipher	FPGA device	Total occupied slices	Maximum frequency (MHz)
S-AES	**Spartan 3A xc3s700-5**	**219**	**87.80**
Hummingbird	Spartan-3 xc3s400-5	273	85.5
AES	Spartan-2 xc2v40-6	1,214	123
AES	Spartan-2 xc2s15-6 Spartan-3 xc3s2000-5	264 17,425	67 196.1
AES	Spartan-3	1800	150

5.5 Hardware Implementation Results

The implementation results are discussed in the following steps:

Step 1: After initializing process shown in fig. 6, the encrypted and decrypted code is configured into the FPGA. For data transmission/reception, 16-bit input Plaintext "6F6B" or "0110111101101011'and 16-bit key "A73B" or

"1010011100111011" is passed through hyper terminal shown in fig. 9. The LSB bits are transmitted or received first followed by the MSB bits.

Step 2: The Plaintext and key are transmitted through Zigbee device (PC side) and is received through other Zigbee device (FPGA side). The encryption process will be performed by Spartan 3A FPGA. Then the encrypted data will be transmitted and received back through Zigbee which is displayed on terminal window shown in fig. 10.

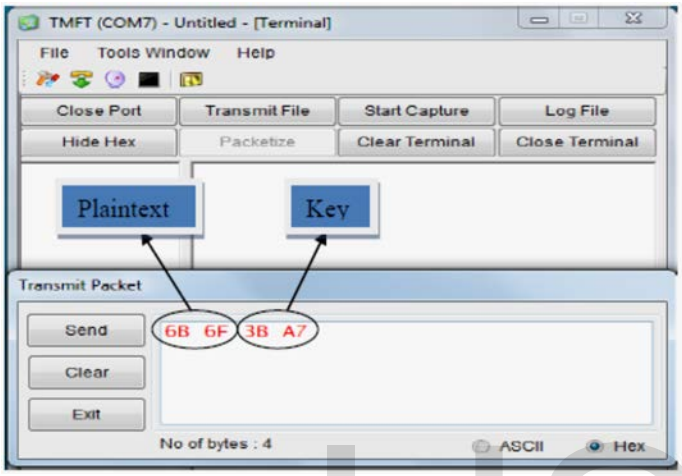

Fig. 9. Transmitting Plaintext and key in terminal window

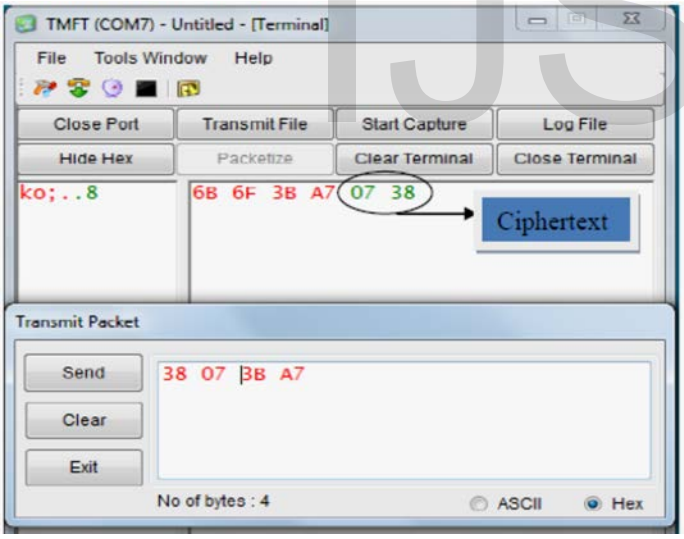

Fig. 10. Receiving Ciphertext "0738" in terminal window

Step 3: The key along with encrypted data "3807" is transmitted for decryption process through hyper terminal from one Zigbee device (PC side) and received by another Zigbee (FPGA side). Then, decryption process will performed by Spartan 3A. The decrypted data is received back through Zigbee transceiver devices. The recovered Plaintext is displayed on terminal window shown in fig. 11.

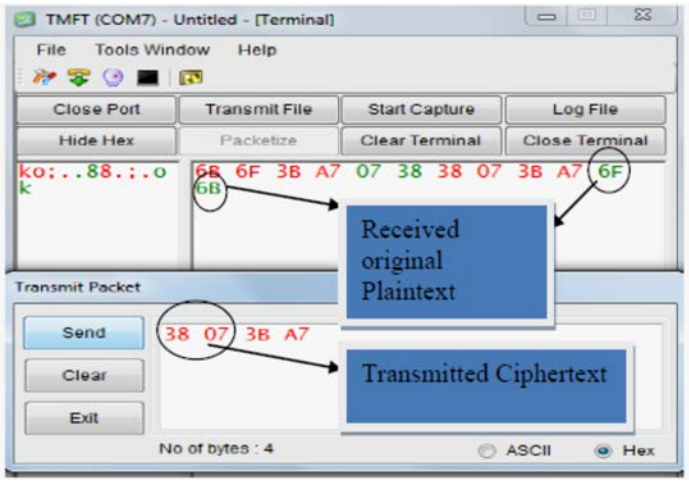

Fig. 11. Transmitting Ciphertext "0738" and receiving original Plaintext "6B6F" in terminal window

6 CONCLUSION

The S-AES encryption/decryption implementation is done on Spartan-3A FPGA where ciphertext is transferred from one Zigbee (FPGA side) and receiving from other Zigbee (PC side). The original plaintext is also retrieved back through same Zigbee modules. The S-AES process is utilizing 219 out of 5888 Slices (3%), 149 out of 11776 (1%) Flip flops and 390 out of 11776 LUTs (3%) for encrypting/decrypting 16-bit Plaintext and 16-bit Key. The proposed S-AES is utilizing 2 clock cycles. Compared to other FPGA implementations of cryptographic algorithm such as AES and Hummingbird, S-AES has better processing speed with smaller area requirement. Consequently, S-AES can be considered as an ideal Cryptographic algorithm for resource-constrained environment such as WSN, smart cards, ID cards, mobile phones and routers etc. When encryption algorithm is needed for high memory embedded processors, then S-AES has to undergo Cryptanalysis.

7 ACKNOWLEDGMENT

I deeply express my sincere gratitude to my guide Mr. Premananda B S, Assistant professor, Department of Telecommunication Engineering, RVCE, Bengaluru for his valuable guidance, continuous encouragement and assistance for this project.

REFERENCES

[1] Andrea Vitaletti and La Sapienza, "Rijndael for sensor networks: is speed the main issue" *Electronic Notes in Theoretical Computer Science (ENTCS)*, vol. 171, April 2007, DOI:10.1016/j.entcs.2006.11.010, pp. 71-81.

[2] Yee Wei Law, Jeroen Doumen and Pieter Hartel "Benchmarking Block Ciphers for Wireless Sensor Networks" *in the proceedings of International conference on Mobile Ad-hoc and Sensor Systems*, 25-27 Oct. 2004, DOI: 10.1109/ MAHSS.2004.1392185, pp. 447-456.

[3] Jing Deng, Richard Han and Shivakant Mishra "A Performance Evaluation of Intrusion Tolerant Routing in Wireless Sensor Networks" *in the proceedings of international conference on dependable systems and networks*, DOI: 10.1109/ ICSENS.2009.5398313, pp. 585-590.

[4] Luanlan, "The AES Encryption and Decryption realization based on FPGA", *in the proceedings of Seventh International Conference on Computational Intelligence and Security*, 3-4 Dec. 2012, DOI: 10.1109/cis.2012.138, pp. 603-607.

[5] Ashwini M. Deshpande, Mangesh S. Deshpande and Devendra N. Kayatanavar, "FPGA Implementation of AES Encryption and Decryption" *in the proceedings of International conference on control, automation, communication and energy conservation*, 4-6 Jun. 2009, pp. 1-6.

[6] Suhas Manangi, Parul Chaurasia and Mahendra Pratap Singh, "Simplified AES for Low Memory Embedded Processors" *in the proceedings of global journal of computer science and technology*, vol. 10, Issue 14, Nov. 2010, pp. 7-11.

[7] Ai-Wen Luo, Qing-Ming Yi and Min Shi, "Design and Implementation of Area optimized AES Based on FPGA" *in the proceedings of International conference* on *Business management and Electronic Information (BMEI*), May. 2011, vol. 1, DOI: 10.1109ICBMEI.2011.5917092, pp. 743-746.

[8] Shuenn-Shyang Wang and Wan-Sheng Ni, "An Efficient FPGA Implementation of Advanced Encryption Standard Algorithm" *in the proceedings of 2004 international symposium on circuits and systems*, 23-24 May. 2004, vol. 2, DOI: 10.1109ISCAS.2004.1329342, pp. 597-600.

[9] Yang Jun, Ding Jun, Li Na and Guo Yixiong, "FPGA-based design and implementation of reduced AES algorithm" *in the proceedings of International conference on challenge in environmental science and computer engineering*, 6-7 Mar. 2012, vol. 2, DOI:10.1109 /CESEC.2012, pp. 67-70.

[10] Xinxin Fan, Guang Gong, Ken Lauffenburger and Troy Hicks, "FPGA Implementation of the Hummingbird Cryptographic Algorithm" *in the proceedings of IEEE International Symposium on Hardware-Oriented Security and Trust*, ISBN: 978-4244-7812-5/10, pp. 48-51.

[11] William Stallings, "Cryptography and network security" 5th edition, Springer publications.