Flowchart Approach to Scalable Encryption Algorithm Design and Implementation in FPGA

Dilja.K
PG scholar/Applied Electronics
Bannari Amman Institute of Technology
Sathyamangalam-638 401, Tamilnadu

Dr. S. Natarajan
Asst. professor, Department of ECE
Bannari Amman Institute of Technology
Sathyamangalam-638 401, Tamilnadu

ABSTRACT
The implementation of encryption/decryption algorithm is the most essential part of the secure communication. In currently existing encryption algorithms there is a tradeoff between implementation cost and resulting performances. Scalable encryption algorithm is targeted for small-embedded application with limited resources (such as memory size, processor capacity). SEA $_{n, b}$ is parametric in the text, key and processor word size and uses a limited instruction set (i.e. NOT, AND, OR, XOR gates, word rotation and modular addition). And it has a provable security against linear and differential cryptanalysis. This paper includes the conversion of loop architecture of SEA into flowchart, in such a way that encryption and decryption process are separated, loop is split into two parts and controlling inputs are removed. By this method it is easy to design in VHDL language, for implementation in FPGA.

Keywords: Scalable Encryption Algorithm, VHDL, FPGA.

1. INTRODUCTION
Scalable encryption algorithm (SEA) is a parametric block cipher for resource-constrained systems (e.g., sensor networks, RFIDs) that has been introduced in [4]. It was initially designed as a low-cost encryption/ authentication routine (i.e., with small code size and memory) targeted for processors with a limited instruction set (i.e., AND, OR, XOR gates, word rotation, and modular addition). The algorithm takes the plaintext, key, and the bus sizes as parameters and, therefore, can be straightforwardly adapted to various implementation contexts and/or security requirements. SEA benefits from a stronger security analysis, derived from recent advances in block cipher design/cryptanalysis. In practice, SEA has been proven to be an efficient solution for embedded software applications using micro controllers.

2. ALGORITHM DESCRIPTION

2.1 Parameters And Definitions
SEAn,b operates on various text, key, and word sizes. It is based on a Feistel structure with a variable number of rounds, and is defined with respect to the following parameters:

• n plaintext size, key size;

• b processor (or word) size;

• nb = n/2b number of words per Feistel branch;

• nr number of block cipher rounds.

As an only constraint, it is required that n is a multiple of 6b (Because both the plain text are separated into 2 parts, and all the operation are done in 3 words). Example- using 8-bit processor, we can derive a 48-bit block ciphers, denoted as SEA48, 8.

Let x be a n/2-bit vector. We consider the following two representations.

• Bit representation: $x_b = x((n/2)-1)\ldots\ldots\ldots x(2)\ x(1)\ x(0)$.

• Word representation: $x_w = x_{nb-1}\ x_{nb-2}\ldots\ldots\ldots x_2\ x_1\ x_0$.

2.2 Basic Operations
Due to its simplicity constraints, SEAn,b is based on a limited number of elementary operations (selected for their availability in any processing device) denoted as follows:

1) Bit wise XOR

2) Mod 2^b addition

3) A 3-bit substitution box S: = [0, 5, 6, 7, 4, 3, 1, 2] that can be applied bit wise to any set of 3-bit words for efficiency purposes. In addition, we use the following rotation operations:

4) Word rotation R, defined on nb-word vectors

R: $x \rightarrow y = R(x) \leftrightarrow$

$y_{i+1} = x_i \quad 0<=I<=nb-2$

$y_0 = x_{nb}\text{-}1$

5) Bit rotation r, defined on nb-word vectors

R: $x \rightarrow y = r(x) \leftrightarrow$

$y = x_{3i} >>>1$

$y_{3i+1} = x_{3i+1}$

$y_{3i+2} = x_{3i+2} <<<1$

Where $0<= i <=(nb/3) -1$ and >>> and <<<, respectively, represent the cyclic right and left shifts inside a word.

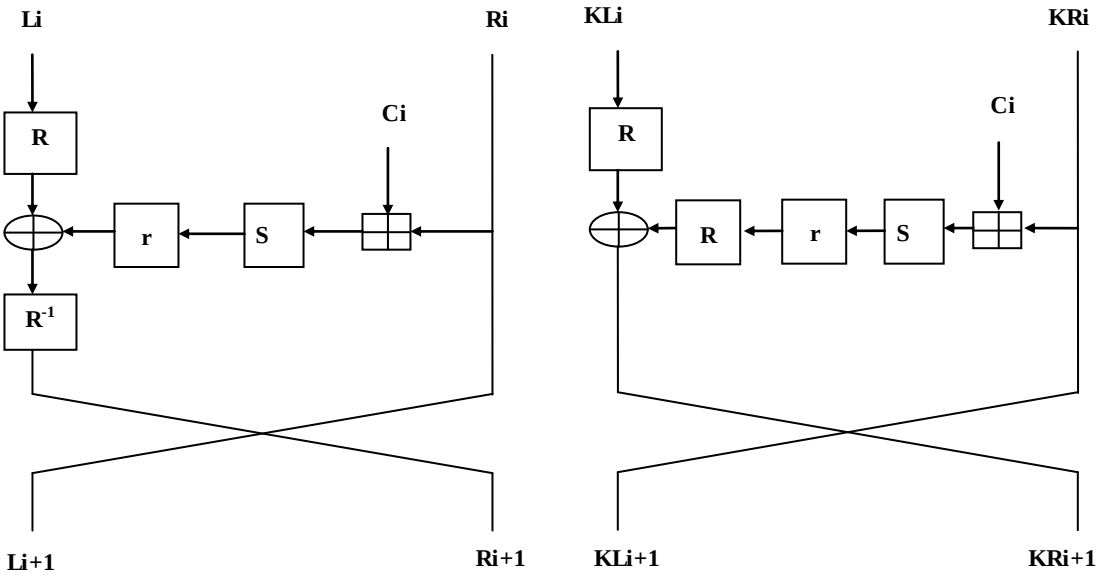

Fig 1. Encrypt/decrypt Round and key round

2.3 Round and Key Round

Based on the previous definitions, the encrypt round FE, decrypt round FD, and key round FK are pictured in Fig.1 and defined as

$[L_{i+1}, R_{i+1}] = F_E(L_i, R_i, K_i) \leftrightarrow$

$R_{i+1} = R(L_i) \oplus r(S(R_i \boxplus K_i))$

$L_{i+1} = R_i$

$[L_{i+1}, R_{i+1}] = F_D(L_i, R_i, K_i) \leftrightarrow$

$R_{i+1} = R^{-1}(L_i \oplus r(S(R_i \boxplus K_i)))$

$L_{i+1} = R_i$

$[K L_{i+1}, KR_{i+1}] = F_K(KL_i, KR_i, C_i) \leftrightarrow$

$KR_{i+1} = KL_i \oplus R(r(S(R_i \boxplus K_i)))$

$KL_{i+1} = KR_i$

2.4 Complete cipher

The cipher iterates an odd number nr of rounds. The following pseudo-C code encrypts a plaintext P under a key K and produces a cipher text C. P, C, and K has a parametric bit size n. The operations within the cipher are performed considering parametric b-bit words.

Pseudo-C code

C=SEAn,b (P,K)

%Initialization

L0&R0=P;

KL0&KR0=K;

%Key scheduling

for i in 1 to [nr/2]

KLi,KRi =FK(KLi-1,KRi-1,C(i));

Switch KL[nr/2], KR[nr/2];

for i in [nr/2] +1 to nr/2-1

KLi,KRi =FK(KLi-1,KRi-1,C(r-i));

% Encryption

for i in 1 to [nr/2]

KLi,KRi =FE(Li-1, Ri-1, KRi-1);

for i in [nr/2] +1 to nr/2

KLi,KRi =FK(Li-1, Ri-1, KLi-1));

%final

C=Rnr&Lnr;

Where & is the concatenation operator, $KR_{[n/2]}$ is taken before the switch and C(i) is a nb-word vector of which all the words have value 0 excepted the LSW that equals i. Decryption is exactly the same, using the decrypt round FD.

3. LOOP ARCHITECTURE OF SEA

The structure of our loop architecture for SEA is depicted in Fig.2, with the round function on the left part and the key schedule on the right part. Resource-consuming blocks are the S boxes and the mod2^b adder; the Word Rotate and Bit Rotate blocks are implemented by swapping wires.

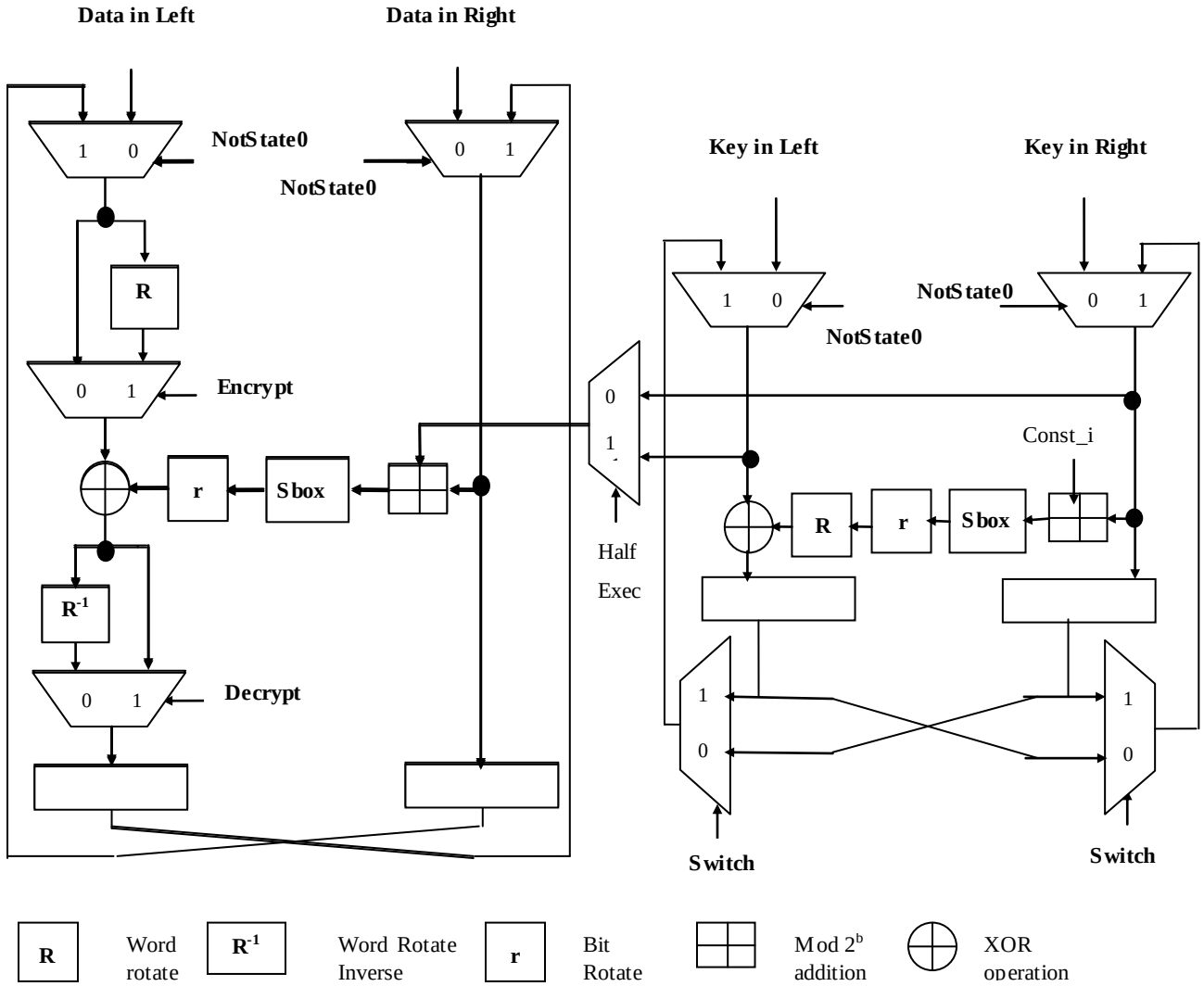

Fig .2. Loop architecture for SEA

According to the specifications, the key schedule contains two multiplexors allowing to switch the right and left part of the round key at half the execution of the algorithm using the appropriate command signal Switch. The multiplexor controlled by Half Exec provides the round function with the right part of the round key for the first half of the execution and transmits its left part instead after the switch. To support both encryption and decryption, finally added two multiplexors controlled by the Encrypt signal. Supplementary area consumption will be caused by the two routing paths. In the round function, the mod 2 adders are realized by using nb, b-bits adders working in parallel without carry propagation between them. In the key schedule, the signal Const_i (provided by the control part) can only take a value between 0 and nr/2.

4. ENCRYPTION AND DECRYPTION FLOWCHART

Figure.3 shows the encryption flow chart used in design of the program. The data and key are the inputs. In the next step both inputs are divided into two parts and applied to the processing blocks. The encryption is completed in two loop operations. In first loop i will take a value of 1 to nr/2. That is the half execution part, the right part of the key is selected during this operations. Both key and data swap in end of each, iteration. After finishing the half execution switch operation is performed. It is done by swap left and right part of key and the remaining rounds the key part will not swap in the next loop. The same operation is performed in next loop except that the left part key is selected in the round operation. Finally the encrypt output is taken by concatenating right and left part out put of encrypt round.

Figure.4 shows decryption flow chart, the same process is done during this flowchart except that inverse word rotation operation is performed after bit rotation, instead in encrypt round the word rotation is performed before bitwise XOR.

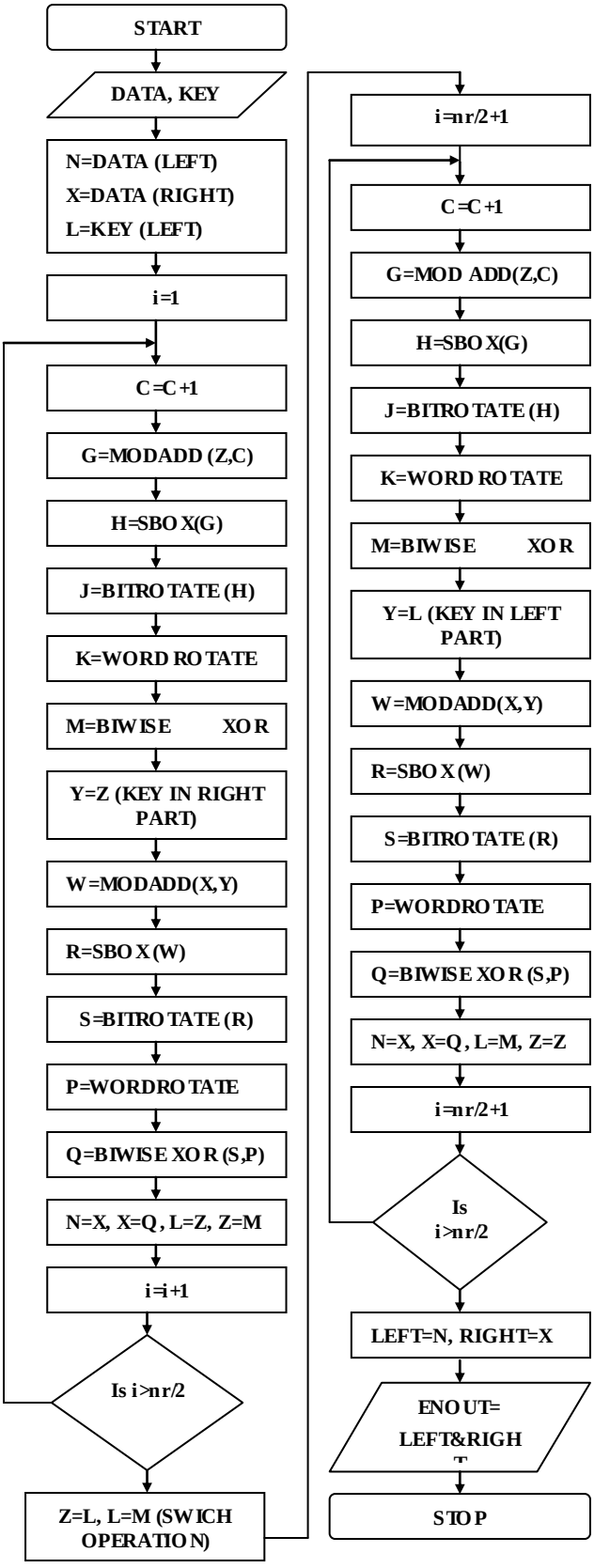

Fig 3. Encryption flowchart

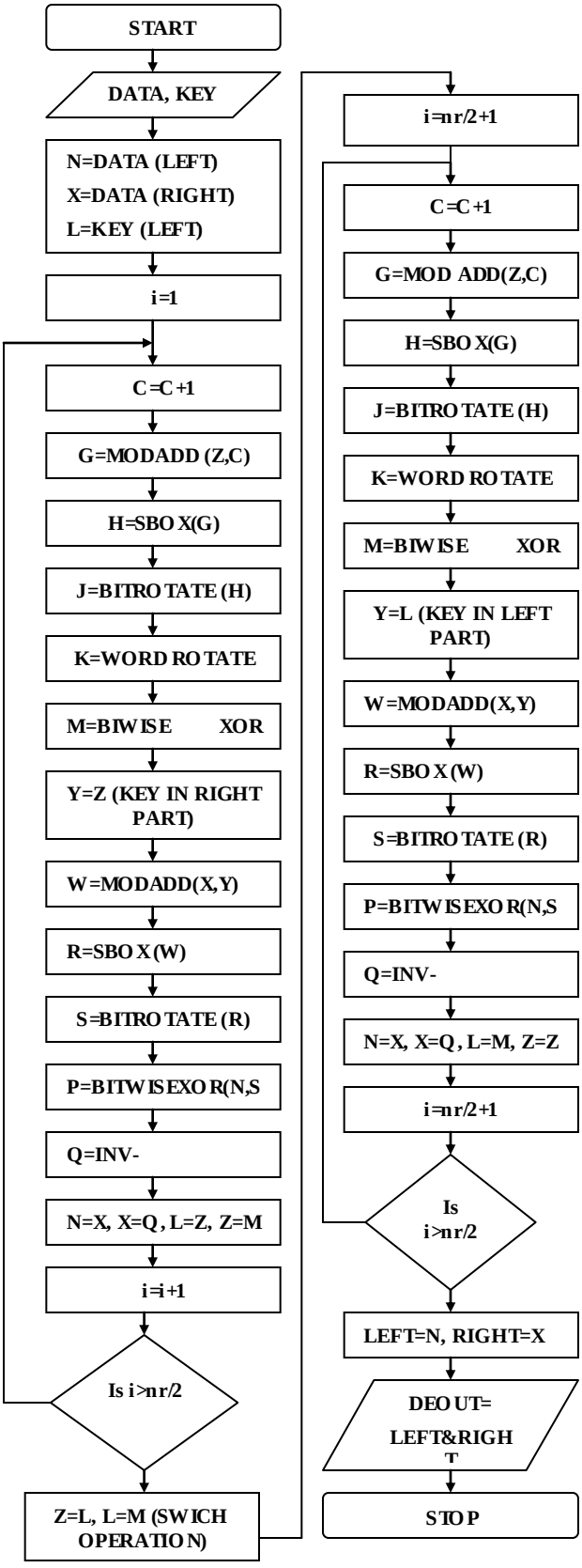

Fig 4. Decryption flowchart

5. EXPERIMENTAL RESULTS

The Scalable Encryption Algorithm has is written in VHDL coding and synthesized using ISE 9.1i tool from Xilinx on a vertex4 platform with speed grade of –12. The device utilization summary and timing summary is given below. From the device utilization summary we can see that 1071 slices are used out of 6144, that is only 17% of the total slices, and look up table used is 1878 out of 12288, that is only 15% of total LUTs. And from timing summary we can see that maximum combinational path delay is 140.603ns. The synthesis report is given below.

The Scalable Encryption Algorithm has is written in VHDL coding and synthesized using ISE 9.1i tool from Xilinx on a vertex4 platform with speed grade of –12. The device utilization summary and timing summary is given below. From the device utilization summary we can see that 1071 slices are used out of 6144, that is only 17% of the total slices, and look up table used is 1878 out of 12288, that is only 15% of total LUTs. And from timing summary we can see that maximum combinational path delay is 140.603ns. The synthesis report is given below.

Device utilization summary:

Selected Device : 4vlx15sf363-12

Number of Slices:	1071	out of	6144	17%
Number of 4 input LUTs:	1878	out of	12288	15%
Number of IOs:	144			
Number of bonded IOBs:	96	out of	240	40%

Timing Summary:

Speed Grade: -12

Minimum period: No path found

Minimum input arrival time before clock: No path found

Maximum output required time after clock: No path found

Maximum combinational path delay: 140.603ns

The Scalable Encryption Algorithm is written in VHDL coding and compiled and simulated in ModelSim SE 5.7g and forced with 2 input values. The waveform of fig.5 shows that we have obtained an output, which is entirely different from the plain text value. That is we have got an encrypted output.

6. CONCLUSION

Scalable encryption algorithm constitutes a suitable solution for a low cost embedded system application like RFID, where area and power is minimum. The on-the-fly key derivation done for iterations, hence look up table is reduced compared to other encryption methods.

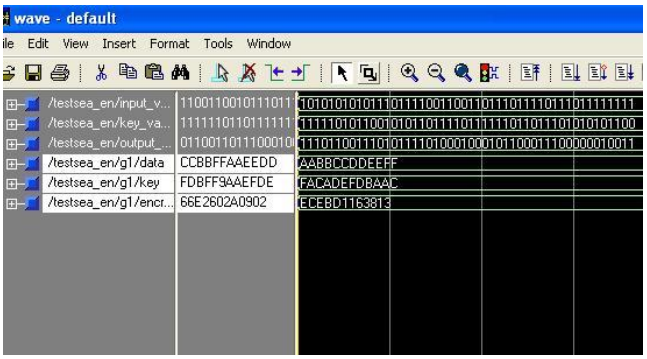

Fig.5-Simulation output

7. REFERENCES

[1] A. J. Elbirt, W. Yip, B. Chetwynd, and C. Paar, "An FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists," in Proc. AES Candidate Conf., 2000, pp. 13–12 .Oct 2005.

[2] K. Jarvinen, M. Tommiska, and J. Skytta, "Comparative survey of high-performance cryptographic algorithm implementations on FPGAs," IEE Proc. Inf. Security, vol. 152, pp. 3–12, Oct. 2005.

[3] F. Macé, F.-X. Standaert, and J.-J. Quisquater, "FPGA Implementation(s) of a Scalable Encryption Algorithm"IEEE Transactions on very large scale integration (VLSI) system .vol. 16, no. 2, FEB 2008.

[4] F.-X. Standaert, G. Piret, N. Gershenfeld, and J.-J. Quisquater, "Sea: A scalable encryption algorithm for small embedded applications," in Proc. CARDIS, 2006, pp. 222–236.

[5] F.-X. Standaert, G. Piret, G. Rouvroy, and J.-J. Quisquater, "FPGA implementations of the ICEBERG block cipher," in Proc. ITCC, 2005, pp. 556–561.

[6] K.Wong, M.Wark and E.Dawson" A single-chip FPGA implementation of the data encryption standard (des) algorithm" Global Telecommunications Conference, 1998. GLOBECOM98. The Bridge to Global Integration. IEEE, 10.1109/ GLOCOM.1998.776849

[7] Advanced Encryption Standard, FIPS PUB 197, Nov. 2001.

[8] Data Encryption Standard, FIPS PUB 46-3, Oct. 1999