Advanced Encryption Standard Implementation Using Truly Random Number Generation

POORNIMA PRASAD M G
Dept of ECE, Mtech (VLSI Design & ES)
PES College of Engineering
Mandya, Karnataka.
prasadpoornima21@gmail.com

Dr K N MURLIDHARA
Professor, Dept of ECE
PES College of Engineering
Mandya, Karnataka.
knm08@rediffmail.com

ABSTRACT

This paper describes a novel Sub bytes approach for implementation of the advanced encryption standard (AES) algorithm, which provides a significantly improved strength against first-order differential electromagnetic and power analysis with a minimal additional overhead. Our method is based on randomization in composite field arithmetic, which entails a low implementation cost while does not alter the algorithm, does not reduce the working frequency, and keeps perfect compatibility with the published standard. In the TRNG design architecture the Dual Modulus Prescaler is being used. It basically comprises of AND logic which controls the pulses produced at the output i.e. 1 or 2 pulses that need to be generated at the output, from 256/257 input pulses respectively. The synchronous divide-by-4/5 divider uses symmetric fashion D flip-flops and the inverters to achieve more than 10 GHz maximum operating frequency. Techniques to improve the statistical quality of the 4/5 dual modulus-prescaler based TRNGs' bit sequences have been presented and verified by simulation and measurement. Post digital processor is added to further enhance the randomness of the output bits. This Proposed TRNG System Implemented using Verilog HDL and Simulated by Modelsim 6.4 c and Synthesized by Xilinx tool. The proposed system implemented in FPGA Spartan 3 XC3S 200 TQ-144. The proposed TRNG has been made into an IP and successfully applied in an SD card for encryption application. The proposed TRNG has passed the NIST tests and Diehard tests.

Keywords - Smart cards, AES, FPGA, Rijndael, Encryption, Modulus Prescaler, Truly random number generator (TRNG).

I. INTRODUCTION

With the extremely rapid development of integrated circuits, smart cards have been widely used in electronic financial transactions, identification, and wireless communication systems. The extending use of smart cards such as SD cards has raised demanding security issues to fulfill the requirements for secrecy of information. The security of the smart cards relies on the generation of unpredictable and irreproducible digital key streams using a nondeterministic random number generator [1] [2]. Therefore, a high-quality truly random number generator (TRNG) plays a fundamental role for the encryption of the smart cards.

Power and area are limited resources in most smart cards. However, most TRNGs mentioned earlier have either high power consumption or low statistical property. The proposed TRNG has been used in an SD card successfully. A post digital processor is added to further enhance the randomness of the output bits. Typically, a TRNG consists of Dual Modulus Prescaler, a random seed generator and a post digital processor, which produces the final output [3]. An important objective of the post digital processor is to provide robustness of the statistical properties of the TRNG output sequence. The scheme of our post digital processor is shown in Fig. The proposed post digital processor is realized by a 64-bit linear feedback shift register and four nonlinear combined functions, which can improve the unpredictability and decorrelation of output random sequence [4].

NIST selected the Rijndael as the AES algorithm which has broad applications, including smart cards, cellular phones, WWW servers, automated teller machines (ATMs), and digital video recorders. Compared to software implementations, hardware implementations of the AES algorithm provide more physical security as

well as higher speed [9]. On 2nd January 1997, the National Institute of Standards and Technology (NIST) invited proposals for new algorithms for the new Advanced Encryption Standard (AES).

The remainder of this brief is organized as follows: Section II introduces the Architecture of the proposed TRNG. Prescaler -based TRNGs are given and verified by simulation and a post digital processor use to further improve randomicity of the TRNG is presented. In section III the proposed algorithm [AES] is introduced and implemented. Measurement results are given in Section IV. Then, the conclusion is drawn in Section V.

II. PROPOSED BLOCK DIAGRAM

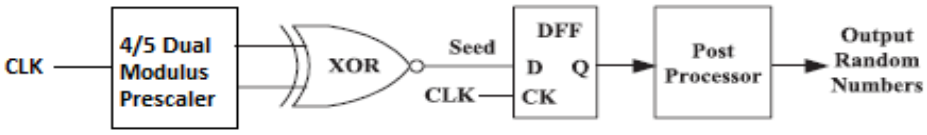

Fig.1. Architecture of the proposed TRNG.

The architecture of the proposed TRNG, (Fig.1)which consists of 4/5 dual modulus prescaler, which is basically comprises of AND logic which controls the pulses produced at the output i.e. 1 or 2 pulses that need to be generated at the output. The synchronous divide-by-4/5 divider uses D flip-flops and the inverters to achieve more than 10 GHz maximum operating frequency [5]. The outputs from the prescaler are combined by the XOR operation to yield Seed, a random sequence of higher statistical quality.

Divide by 4/5 Dual Modulus prescaler also called divide by 4/5 counter (Fig.2). It consists of D flip flops and NOT gates. The input frequency Fin is given to all the flip flops. The selection of the counter depends on the Mode control. If MC=0, one output clock cycle is equal to 4 input clock cycles else if MC=1, one output clock cycle is equal to 5 input clock cycles [6].

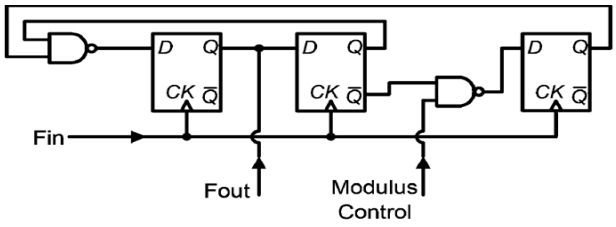

Fig.2. Divide by 4/5 dual modulus prescaler

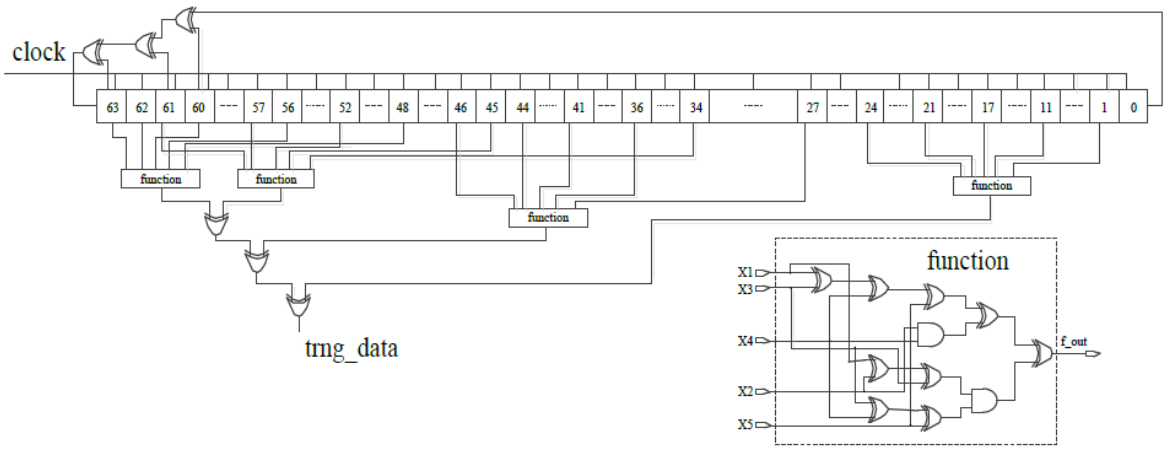

Fig.3. Postdigital process scheme of the TRNG.

An important objective of the postdigital processor is to provide robustness of the statistical properties of the TRNG output sequence. The scheme of our postdigital processor is shown in Fig.3. The proposed postdigital processor is realized by a 64-bit linear feedback shift register and four nonlinear combined functions, which can improve the unpredictability and decorrelation of output random sequence.
The Main objective of this paper describes a novel Sub bytes approach for implementation of the advanced encryption standard (AES) algorithm and another contribution of this paper is that it designs Encryption Design using Shit rows, Mixed Column, Add Round Key and Decryption Part will also be designed. The design uses an iterative looping approach with block and key size of 128 bits, look up table implementation of s-box, mix column operation and inverse mix column operations are designed as LUTs. Finally with the help of Random Number Generator Block the TRNG output 128 bit data will be given as input key to AES Encryption. The results of this paper can be served for protecting the Data with the High Security.

III PROPOSED ALGORITHM

A. ADVANCED ENCRYPTION STANDARD (AES)

AES algorithm is a symmetric block cipher that can encrypt and decrypt the information. Encryption can convert the original data into unintelligible data and is called as cipher text. Decryption can convert the cipher text form into original data, which is called plain text [8]. The AES algorithm has a fixed block size of 128 bits and a key length of 128, 192 or256 bits. It generates its key from an input key using the Key Expansion function. The AES operates on a 4x4 array of bytes which is called a state. AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys [7]. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

AES algorithm has two processes:
1. Encryption / Decryption Process
2. Key generating and Scheduling Process

The algorithm begins with an Add round key stage followed by 9 rounds of four stages and a tenth round of three stages. This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of its counterpart in the encryption algorithm. The transformations in Rijndael consider the data block as a four-column rectangular array of 4-byte vectors. The key is also considered to be a rectangular array of 4-byte vectors—the number of columns is dependent on key length.

Rijndael encryption consists of four operations
1. Substitution Byte
2. Shift Row
3. Mixcolumn
4. Addround Key

The Rijndael decryption consists of four inverse operations of encryption which are compliment functions of encryption.
1. Inverse Substitution Byte
2. Inverse Shift Row
3. Inverse Mixcolumn
4. Addround Key

A. Sub Byte and Inverse Sub Byte transformation

The SubBytes transformation is a nonlinear byte substitution that operates independently on each byte of the substitution table (S-box). In the Sub Bytes step, each byte in the state matrix is replaced with a Sub Byte using an 8-bit data from the Rijndael S-Box. In the Inverse Sub Byte step, each byte in the cipher matrix is replaced with corresponding inverse Sub Byte. Sub Byte operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over Galois Field (2^8) [10].

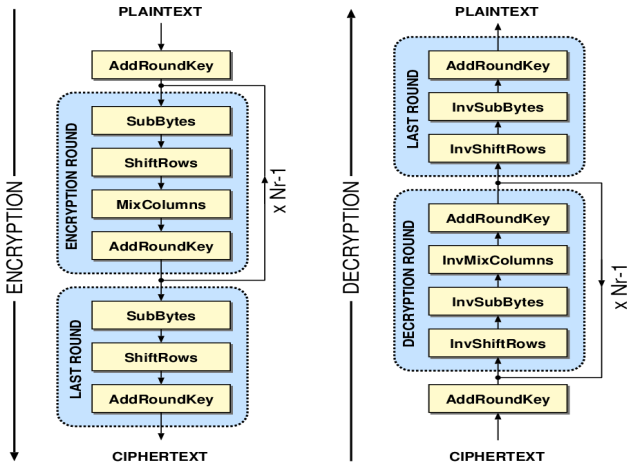

Fig 3. AES Encryption & Decryption process in each round

B. Shift Row Transformation

In the Shift Row transformation, the bytes in the last three rows of the state are cyclically shifted over different numbers of bytes (offsets). The first row, r=0, is not shifted. Inverse Shift Row transformation does the same shift operation towards right. Fig shows the Shift Row operation.

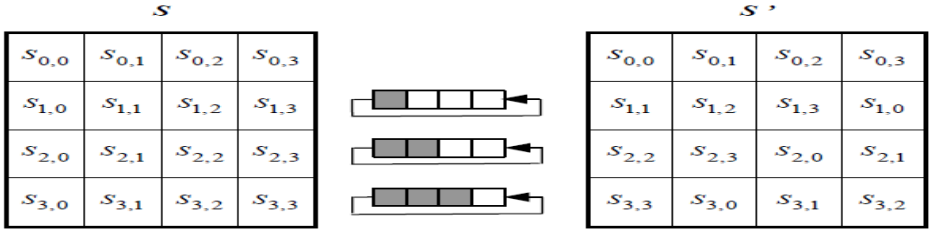

Fig.4. AES Shift Row operation

C. Mix Column and Inverse Mix Column operation

In the mix Column step, the four bytes of each column of the state are combined using an invertible linear transformation. All entries in the state matrix are considered to be a polynomial and multiplied by a fixed polynomial. The Mix Column and Inverse Mix Column are represented in a matrix form as form as equation A, B.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

(A) (B)

D. Add Round Key operation

In this operation, bitwise exclusive-or (XOR) operation is performed between outputs from the Mix Column and Round key. For AES-128, 128 bit operation is performed.

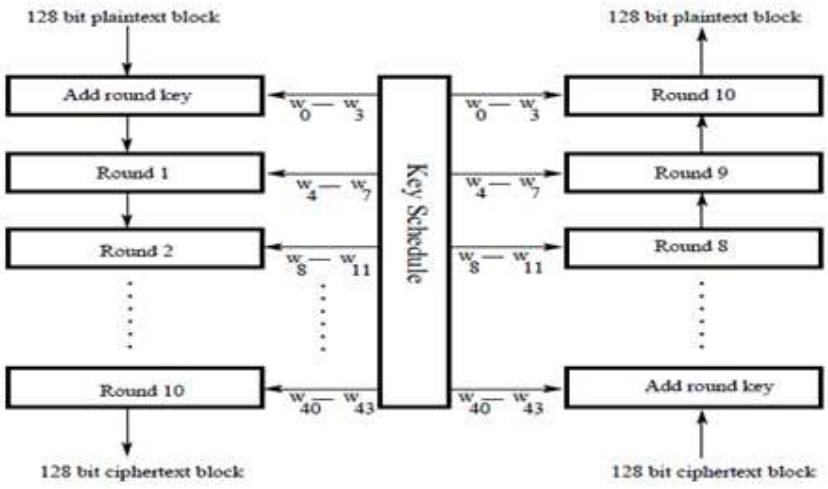

Fig 5. The overall structure of the AES for the case of 128-bit Encryption key.

B. AES Key Schedule

We will start the implementation of AES with the Cipher Key expansion. We intend to enlarge our input cipher key, whose size varies between 128 and 256 bits into a larger key, from which different RoundKeys can be derived.The Key Schedule is responsible for expanding a short key into a larger key, whose parts are used during the different iterations. Each key size is expanded to a different size:
An 128 bit key is expanded to an 176 byte key.
An 192 bit key is expanded to an 208 byte key.
An 256 bit key is expanded to an 240 byte key.
There is a relation between the cipher key size, the number of rounds and the Expanded Key size. For an 128-bit key, there is one initial AddRoundKey operation plus there are 10 rounds and each round needs a new 16 byte key, therefore we require 10+1 Round Keys of 16 byte, which equals 176 byte. The same logic can be applied to the two other cipher key sizes. The general formula is that:
ExpandedKeySize = (nbrRounds+1) * BlockSize

- **Rotate:**

The 4-byte word is cyclically shifted 1 byte to the left:

- **Rcon:**

Just note that the Rcon values can be pre-calculated, which results in a simple substitution (a table lookup) in a fixed Rcon table.

- **S-Box:**

The Key Schedule uses the same S-Box substitution as the main algorithm body. The S-Box values can either be calculated on-the-fly to save memory or the precalculated values can be stored in an array. We will store the values in an array.

IV. SIMULATION RESULT

The 128bit TRNG data is generated from the post-processor scheme. From the TRNG design key expansion process is designed in AES algorithm. It reduces the Area of AES design. The TRNG design and the AES algorithm design is simulated and synthesized using Xilinx 9.1i ISE tool and the targeted FPGA is spartan 6 XC6SLX45 which belongs to virtex-5 family. The design uses only LUTs, ROMs for all the operations of AES encryption and decryption. This approach reduces device utilization and significantly improves the speed compared to other implementation. The utilization summary for the device is presented in Table-1.

Table 1: Slice Logic Utilization

Number of Flip flops	128 out of 9840
Number of Slices	75 out of 1920
Number of I/Os	130 out of 487
Number of Input LUTs	10 out of 9840
Number used as Logic	1106 out of 69120

In this proposed design, the encryption unit takes 10 clock cycles to complete the operation. The maximum path delay of the design is 5.02ns resulting in a maximum frequency of operation as 292.402MHz. The throughput of the proposed encryption module is 3.47Gbps.

V. CONCLUSION

A novel AES implementation with a simple and integrated counter measure against Data Hacking is presented. The counter measure is based on mathematical properties of Rijndael algorithm and retains perfect compatibility with the published Standard. Relevant AES Implementation with TRNG for Key Generation Method is conducted with efficient area and speed characteristics.. Nearly all the algorithms embedded in smart cards have been designed to resist at high level to linear, differential and high order differential attacks, whereas nothing has been done to make them inherently resistant attacks. However, this work will be implemented in the FPGA. The proposed system implemented in FPGA Spartan 3 XC3S 200 TQ-144. Another contribution of this paper is that it designs Encryption Design using Shit rows, Mixed Column, Add Round Key and We Will Design a Decryption Part also. Finally with the help of Genetic Algorithm based Encoding for Key Generation is used for Encryption and Decryption Process.

REFERENCES

[1]. K.H. Tsoi, K. H. and K.H. Leung ET. al., “Compact FPGA-based True and Pseudo Random Number Generators”, Proceedings of the 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM’03), 2003, pp. 51- 61.
[2]. Kohlbrenner, Paul and Kris Gaj, “An Embedded True Random Number Generator for FPGAs”, Proceedings Of The 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays (FPGA ’04), pp. 71-78.
[3]. Knuth, Donald E., “Random Number Generator”, US Patent 3,548,174, August 10 2009.
[4]. Oerlemans, Robert Vincent Michael, “Digital True Random Number Generating Circuit”, US Patent 6,807,553 B2, October 19, 2004.
[5]. Ching-Yuan Yang, Guang-Kaai Dehng and Shen-Iuan Liu, “High-speed divide-by-4/5 counter for a dual Modulus prescaler,” vol. 33, no. 20, Sep 1997.
[6]. Xuan Jiahui, Wang Zhigong, Tang Lu and Xu Jian, “A 3-GHz Dual-Modulus Prescaler Based on Improved Master-Slave DFF,” Institute of RF- & OE-ICs, Southeast University, China.
[7]. M. Goswami and S. Kannojiya, "High performance FPGA Implementation of AES Algorithm with 128-bit Keys," Proc. IEE Int. Conf. Advances Computing Comm., vol. 1, Himarpur, India, 2011, pp.281-286
[8]. W. Wei, C. Jie and X. Fie, "An Implementation of AES Algorithm on FPGA," IEEE Int. Conf. on Fuzzy Systems and Knowledge discover 2012, pp. 1615-1617.
[9]. W. Stallings, "Cryptography and Network security principles and practice," Pearson edition 2009, pp. 135 160.
[10]. P.V.S. Shastry, A. Agnihotri, D. Kachhwaha, J. Singh and M.S. Sutaone," A combinational logic Implementation of S-box of AES” IEEE 54th Int. Midwest symp. On Circuits and Systems (MWSCAS), Aug. 2011, pp. 1-4