International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 3 (2026)

ijerst.editor@gmail.com

editor@ijerst.com

ISSN 2319-5991 www.ijerst.com

Peer Reviewed, Referred & Indexed Journal

Research Paper

DESIGN AND FUNCTIONAL ANALYSIS OF THE ADVANCED ENCRYPTION STANDARD (AES) ALGORITHM ON FPGA

Aira Fatima[1], Tazeen[2]
[1]PG Scholar, Department of ECE, Shadan Women's College of Engineering and Technology, Hyderabad, fatima.aira@gmail.com
[2]Asst Professor, Department of ECE, Shadan Women's College of Engineering and Technology Tazeen@gmail.com

ABSTRACT
Internet of things (IoT), internetworking of smart devices, embedded with sensors, software, electronics and network connectivity that enables to communicate with each other to exchange and collect data through an uncertain wireless medium. Recently IoT devices are dominating the world by providing it's versatile functionality and real-time data communication. Apart from versatile functionality of IoT devices, they are very low battery powered, small and sophisticated, and experience lots of challenges due to unsafe communication medium. Despite the fact of many challenges, the energy issue is now becoming the prime concern. Optimization of algorithms in terms of energy consumption has not been explored specifically, rather most of the algorithms focus on hardware area to minimize it extensively and to maximize it on security issue as possible. But due to recent emerge of IoT devices, the main concern is shifting to moderate security and less energy consumption rate. We present AES Algorithm is working with one key, to improving the security of the design here we are adding one module for supplying key randomly, for applying random key we are taking random number generator module in this project.

1. INTRODUCTION
The selective application of technological and related procedural safeguards is an important responsibility of every Federal organization in providing adequate security to its electronic data systems. This publication specifies a cryptographic algorithm, the Advanced Encryption Standard (AES) which may be used by Federal organizations to protect sensitive data. Protection of data during transmission or while in storage may be necessary to maintain the confidentiality and integrity of the information represented by the data.

The algorithms uniquely define the mathematical steps required to transform data into a cryptographic cipher and also to transform the cipher back to the original form. The Advanced Encryption Standard is being made available for use by Federal agencies within the context of a total security program consisting of physical security procedures, good information management practices, and computer system/network access controls.

Data encryption (cryptography) is utilized in various applications and environments. The specific utilization of encryption and the implementation of the AES will be based on many factors particular to the computer system and its associated components. In general, cryptography is used to protect data while it is being communicated between two points or while it is stored in a medium vulnerable to physical theft.

Communication security provides protection to data by enciphering it at the transmitting point and deciphering it at the receiving point. File security provides protection to data by enciphering it when it is recorded on a storage medium and deciphering it when it is read back from the storage medium. In this the key must be available at the transmitter and receiver simultaneously during communication.

2. LITERATURE SURVEY
The literature survey focuses its attention towards AES, particularly to utilize under low power consumption, high security, better performance and improved efficiency. The implementation feasibility in VLSI environment is also studied and analyzed in depth.

2.1 Fault Analysis in AES-CBC Algorithm Using Hamming Code for Space Applications
National institute of standard and technology (2001) presented computer security. Two FIPS publications already prove the modes of operation for two particular block cipher algorithms . Four of these modes are equivalent to the ECB, CBC, CFB, and OFB modes with the Triple DES algorithm (TDEA) as the underlying block cipher. For any given key, the block cipher algorithm of the mode consists of two function that are inverses of each other.

Francois-Xavier Standaert, Gael Rouvroy, Jean-Jacques Quisquater, and Jean-Didier Legat presented (2004) discussed about the Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware. It addressed various approaches for efficient FPGA implementations of the Advanced Encryption Standard algorithm. In implementation of block ciphers, several strategies can produce effective designs. Inherent constraints of FPGAs were taken into account in order

to define an efficient methodology. Inside these architectures, the authors proposed algorithmic optimizations for the substitution box, and also efficient combinations between the diffusion layer and the key addition. [25] Farhadian.A and Aref.M.R (2009) presented efficient method for simplifying and approximating the s-boxes based on power functions . In this paper cipher algorithms, power functions over finite fields and special inversion functions have an important role in the S-box design structure. A new systematic efficient method is introduced to crypt analyze such S-boxes. This method is very simple and does not need any heuristic attempt and can be considered as a quick criterion to find some simple approximations. Using this new method, approximations can be obtained for advanced encryption standard (AES) like S-boxes, such as AES, Camellia, and Shark and so on. Finally as an application of this method, a simple linear approximation for AES S-box is presented.

Akashi Satoh, Sumio Morioka, Kohji Takano, and Seiji Munetoh (2011) presented a Compact Rijndael Hardware Architecture with S-Box Optimization . Encryption and decryption data paths are combined and all arithmetic components are reused. An extremely small size of 5.4 K gates is obtained for a 128-bit key Rijndael circuit using a 0.11-µm CMOS standard. In order to minimize the hard-ware size, the order of the arithmetic functions were changed, and the encryption-decryption data paths are efficiently combined with respect to cell library. Logic optimization techniques such as factoring were applied to the arithmetic components, and gate counts were reduced.

Ashkan Masoomi and Roozbeh Hamzehiyan (2012) presented a new approach for detecting and correcting errors in satellite communications based on Hamming Error Correcting Code . A novel model to detect and correct Single Event Upsets in on-board implementations of the AES algorithm was based on hamming error correcting code. Single Even Upset (SEU) faults occur in the on-board during encryption due to radiation. Some of the AES modes like ECB, CBC, OFB, CFB and CTR performances have been analyzed. From that, CTR mode has been recommended as the optimum choice for satellite applications. 26 Ramesh Babu, George Abraham and Kiransinh Borasia (2012) presented a Review on Securing Distributed Systems Using Symmetric Key Cryptography . It was used to evaluate the importance of Symmetric Key Cryptography for Security in Distributed Systems. Two symmetric key cryptographic algorithms DES and AES were commonly used. These two algorithms were evaluated on the parameters such as key size, block size, number of iterations. From the literatures reviewed of various implementations and analysis of both the algorithms, it can be concluded that AES algorithm has over-shadowed the DES algorithm in many areas.

Karri, R., Wu, K., Mishra, P., and Kim, Y. (2002) Concurrent Error Detection Schemes for Fault-Based Side-Channel Cryptanalysis of Symmetric Block Ciphers . They presented algorithm level, round level, and operation level CED (Concurrent Error Detection) architectures for symmetric block ciphers. The algorithm was independent and can be applied to almost any symmetric block cipher. The proposed scheme introduced moderate area overhead and interconnects complexity to achieve permanent as well as transient fault tolerance. This approach assumes that the key RAM, comparator, or both encryption and decryption modules simultaneously are not under attack or faulty.

2.2 A Secure Implementation of Nonlinear AES S-Box and Fault Analysis in AES-CM Algorithm Using Hamming Code for Space Applications.

Ross Anderson, Eli Biham, Lars Knudsen (1999) presented a Proposal for the Advanced Encryption Standard . Its design is highly conservative, yet still allows a very efficient implementation. With a 128-bit block size and a 256-bit key, it is as fast as DES on the market leading Intel Pentium/MMX platforms yet we believe it to be more secure than three-key triple-DES. The linear transformations were just bit 27 permutations, which were applied as rotations of the 32-bitwords in the bit slice implementation. The authors also considered replacing the XOR operations by seemingly more complex operations, such as additions. Finally cognate algorithms with the same structure as Serpent but with block sizes of 64, 256 and 512 bits.

A. J. Elbirt, W. Yip, B. Chetwynd, C. Paar (2000) presented an FPGA implementation and performance evaluation of the AES block cipher candidate algorithm finalists . Reprogrammable devices such as Field Programmable Gate Arrays (FPGAs) are highly attractive options for hardware implementations of encryption algorithms as they provide cryptographic algorithm agility, physical security, and potentially much higher performance than software solutions. The implementations of each algorithm will be compared in an effort to determine the most suitable candidate for hardware implementation within commercially available FPGAs. A design methodology was established which in turn led to the architectural requirements for a target FPGA. The best speed-optimized implementations were identified for each AES finalist in both non-feedback and feedback modes.

Pawel Chodowiec, Kris Gaj, Peter Bellows and Brian Schott (2001) presented Experimental Testing of the Gigabit IPSec-Compliant Implementations of Rijndael and Triple DES Using SLAAC-1V FPGA Accelerator Board . Full implementations of the new Advanced Encryption Standard, Rijndael, and the older American federal standard, Triple DES, were developed and experimentally tested using the SLAAC-1V FPGA accelerator board, based on Xilinx Virtex 1000 devices.

Demonstration of a capability to enhance our circuit to handle the encryption and decryption throughputs of over 1 Gbit/s regardless of the chosen algorithm. For Rijndael in the basic iterative architecture, the results for encryption and decryption are different, with decryption slower than encryption by about 13% in experimental testing. The IPSec-compliant encryption/decryption units of the new Advanced Encryption Standard - Rijndael and 28 the older encryption standard Triple DES have been developed and tested experimentally. Khoa Vu, David Zier (2003) presented FPGA Implementation AES for CCM Mode encryption using Xilinx Spartan-II. In this paper discusses a possible FPGA implementation of the AES algorithm specifically for the use in CCM Mode Encryption . It investigates the possibility of creating an off-chip AES system for CCM so that the process can be speed up. The AES implementation on the FPGA is a viable solution for improving the speed and processing power of CCM Mode Encryption. A much larger FPGA or ASIC would be preferred, for both encryption and decryption could be implemented as well as some pipelining of processes. Although the design was implemented, it was intended to be interfaced with a micro-controller/microprocessor running the CCM Mode Encryption.

Xinmiao Zhang and Keshab K. Parhi (2004) presented high speed VLSI architectures for the AES algorithm. A novel high-speed architecture for the hardware implementation of the Advanced Encryption Standard (AES) algorithm . Composite field arithmetic is employed to reduce the area requirements, and different implementations for the inversion in subfield GF (2^4) are compared. In order to explore the advantage of sub-pipelining further, the SubBytes/InvSubBytes is implemented by combinational logic to avoid the unbreakable delay of LUTs in the traditional designs. Fully Sub-pipelined encryptor/decryptors using other key lengths can be implemented by adding more copies of round units and modifying the key expansion unit slightly. Expected throughput is slightly lower than the encryptor-only implementations.

Davidd Hwang, Patricks Chaumont, Kristiri, and Ingrid Verbauwhede (2006) discussed about securing embedded systems Education, pp. 134-161, 2006 .. A 29

top-down, multi abstraction layer approach for embedded security design reduces the risk of security flaws, letting designers maximize security while limiting area, energy, and computation costs. Embedded systems are essentially processor-based devices. Embedded devices pose severe resource constraints on the security architecture in terms of memory, computational capacity, and energy operating under resource-constrained conditions. At the algorithm level, a designer must select or design both cryptographic and application-specific algorithms for implementation on the embedded device.

Joo, M.K., Kim, J.H. and Choi, Y.H., “A fault tolerant architecture for symmetric block ciphers,” in Proceedings of 11th Asian Test Symposium, 2002. (ATS 2002), pp. 212-217, 2002 . Symmetric key encryption algorithm with low-Energy consumption is required to the applicable sensor networks. Though sensor network provides various capabilities, it is unable to ensure the secure authentication between sensor nodes. Eventually it causes the losing reliability of the entire network and many secure problems. In this paper, a solution of reliable sensor networks to analyze the communication efficiency through measuring performance of AES-128 CBC algorithm which is selected by default in sensor networks by plaintext size and cost of operation per hop according to the network scale was proposed. Finally the results from proposed was proved to be better than existing system.

Bertoni, G., Breveglieri, L., Koren, I., Maistri, P., and Piuri, V (2002) presented Concurrent fault detection for a hardware implementation of the Advanced Encryption Standard (AES) . It is very important not only to protect the encryption/decryption process from random faults. It will also protect the encryption/decryption circuitry from an attacker who may maliciously inject faults in order to find the encryption secret key. They presented a parity prediction algorithm which was designed for each of the four round transformations employed by 30

the Encryption module. Since all byte elements of the output state for each transformation was computed in parallel, they did the same with the output parity bits. and they showed that the proposed scheme leads to very efficient and high coverage fault detection.

Bertoni, G., Breveglieri, L., Koren, I., Maistri, P., and Piuri, V presented Error analysis and detection procedures for a hardware implementation of the advanced encryption standard . They presented two fault detection schemes. The first was a redundancy-based scheme while the second uses an error detecting code. The latter was a novel scheme which leads to very efficient and high coverage fault detection. It was based on the use of parity codes, exhibits very good fault coverage, limited hardware overhead cost, and short detection latency. For single bit faults and multiple bit faults of odd order, it proved that (under reasonable assumptions) the fault coverage of the parity-based detection technique was good compared with the existing one.

Breveglieri, L. , Koren, I. , and Maistri, P. ,“An operation-centered approach to fault detection in symmetric cryptography ciphers,” IEEE Transactions on Computers, vol. 56, no. 5, pp. 635-649, 2007 . An operation-centered approach to fault detection in symmetric cryptography ciphers is presented. They proposed a general framework for error detection in symmetric ciphers based on an operation-centered approach. They first enumerated the arithmetic and logic operations included in the cipher and analyze the efficiency and hardware complexity of several error-detecting codes for each such operation. They

recommended an error-detecting code for the cipher as a whole based on the operations it employs. The trade-off between the checkpoint frequency and the error coverage was also evaluated.

Wu, K., Ramesh, K., Kuznetsov, G. in Proceedings of International Test Conference 2004 (ITC 2004), pp. 1242-1248, presented Low cost concurrent error 31 detection for the advanced encryption standard . In this they presented a new low-cost concurrent checking method for the Advanced Encryption Standard (AES) encryption algorithm. In this method, the parity of the 128-bit input is determined and modified step-by-step into the parity of the 128-bit output according to the processing steps of the AES encryption. For the parity preserving AES steps Shift-Rows and Mix-Column no parity modifications are necessary. The modified parity is compared in any round with the actual parity of the outputs of the round. The method detects technical faults and deliberately injected faults during normal operation.

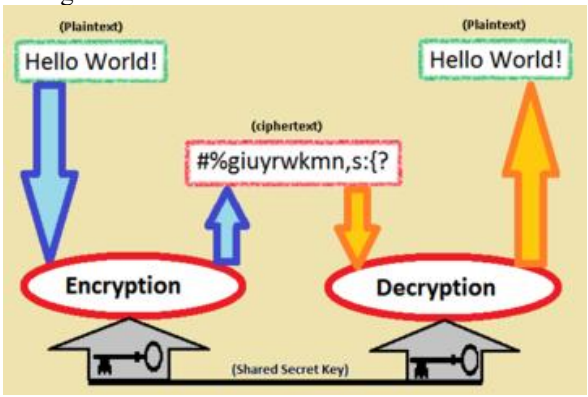
Yen, C. H. and Wu, B.F., "Simple error detection methods for hardware implementation of advanced encryption standard" . IEEE Transactions on Computers, vol. 55, no. 6, pp. 720-731, 2006. In this they proposed a simple, symmetric, and high-fault-coverage error detection scheme for AES. Although the erroneous bits were diffused in AES, this work used the linear behavior of each operation in AES to design a detection scheme. It is possible to use only one 8-bit register for storing the parities during hardware implementation. This error detection may also be used in encryption-only or decryption-only designs. Because of the symmetry of the proposed detection scheme, the encryption and decryption circuit can share the same error detection hardware. The proposed schemes can be applied in the implementation of AES against differential fault attacks and can be easily implemented in a variety of structures, such as 8-bit, 32-bit, or 128-bit structures.

3. AES (ADVANCED ENCRYPTION STANDARD)

The selective application of technological and related procedural safeguards is an important responsibility of every Federal organization in providing adequate security to its electronic data systems. This publication specifies a cryptographic algorithm, the Advanced Encryption Standard (AES) which may be used by Federal organizations to protect sensitive data. Protection of data during transmission or while in storage may be necessary to maintain the confidentiality and integrity of the information represented by the data.

The algorithms uniquely define the mathematical steps required to transform data into a cryptographic cipher and also to transform the cipher back to the original form. The Advanced Encryption Standard is being made available for use by Federal agencies within the context of a total security program consisting of physical security procedures, good information management practices, and computer system/network access controls.

Data encryption (cryptography) is utilized in various applications and environments. The specific utilization of encryption and the implementation of the AES will be based on many factors particular to the computer system and its associated components. In general, cryptography is used to protect data while it is being communicated between two points or while it is stored in a medium vulnerable to physical theft. Communication security provides protection to data by enciphering it at the transmitting point and deciphering it at the receiving point. File security provides protection to data by enciphering it when it is recorded on a storage medium and deciphering it when it is read back from the storage medium. In this the key must be available at the transmitter and receiver simultaneously during communication.



Advanced Encryption Standard (AES)

Cryptography is probably the most important aspect of communication security becoming increasingly important as a basic building block for computer security. Cryptographic systems are characterized along three independent dimensions:

1. The type of operations used for transforming plaintext to ciphertext: All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext is mapped into another element, and transposition, in which element in the plaintext are rearranged. The fundamental requirement is that no information be lost. Most systems referred to as product systems, involve multiple stages substitutions and transpositions.
2. The number of keys used: If both sender and receiver use the same key, the system is referred to as symmetric, single-key, secret-key, or conventional encryption. If the sender and receiver use different keys, the system is referred to as asymmetric, two-key, or public-key encryption.

The way in which the plaintext is processed: A block cipher processes the input one block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along. Before beginning, we define some

terms. A symmetric encryption scheme has five ingredients:

- Plaintext: This is the original intelligible message or data that is fed into the algorithm as input.
- Encryption Algorithm: The encryption algorithm performs various substitutions and transformations on the plaintext.
- Secret key: The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.
- Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertext. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.
- Decryption algorithm: This is essential the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

Advanced Encryption Algorithm (AES)

The Advanced Encryption Algorithm (AES), a symmetric block cipher algorithm that can processes blocks of 128-b, using cipher keys with lengths of 128,192 and 256-bits. The input and output for the AES algorithm each consist of sequence of 128-b (digit with values of 0 or 1). These sequences will sometimes refer to as blocks and the number of bits they contain will be referred to as their length. Internally, the AES algorithm's operation is are performed on a two-dimensional array of bytes called the state as shown in the Figure.1. The state consists of four rows of bytes, each containing Nb bytes, where Nb is the block length. Here Nb = 4, which reflects the number of 32-b words (number of columns) in the state.

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

Figure.1 State Array

For AES algorithm, the length of the cipher key, K, is 128,192 or 256 bits. The key length is represented by Nk = 4, 6, or 8, which reflects the number of 32-b words (number of columns) in the cipher key. The number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by Nr, where Nr = 10 when Nk = 4, Nr = 12 when Nk = 6 and Nr = 14 when Nk = 8. The only key-block-round combination that confirms to the standard is given in Figure.2. Here we consider Nr = 10 and Nk = 4 for design of the architecture.

	Key Length (Nk words)	Block Size (Nb words)	Number of Rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Figure.2 Key-Block-Round combination

The Figure.3 shows the complete structure of AES algorithm (both encryption and decryption process). In FIPS 197 standard, the block is depicted as square matrix of bytes. The block is copied into state array, which is modified at each stage of encryption or decryption processes. Similarly, the 128-b key is depicted as a square matrix of bytes. The key is then expanded into an array of key schedule words; each word is of 4-bytes and the total key scheduled is 44 words for the 128-b key input.

The Cipher and Decipher process is explained in the pseudo code in Figure.4 and Figure.5. The individual transformation – Byte substitution, Shift row, Mix column and Add round key- processes the state and is described in the following subsection. As shown in the Figure.4, all Nr rounds are identical with the exception of the final round, which does not include Mix column transformation and as such same in the decipher process shown in the Figure.5. Let us now see the detailed description of each of the four stages used in AES. For each stage both encryption and decryption transformation will be explained. This is followed by a discussion of key expansion process used in AES.

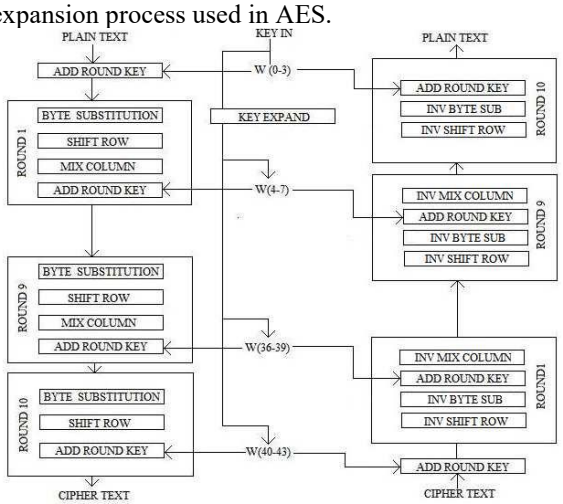


Figure.3 AES Encryption and Decryption

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)]
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])

   for round = 1 step 1 to Nr-1      // round 1 to 9
      SubBytes(state)
      ShiftRows(state)
      MixColumns(state)
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)                   // last round
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

Figure.4 Pseudo Code for AES Cipher Process

```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   for round = Nr-1 step -1 downto 1      // round 9 to 1
      InvShiftRows(state)
      InvSubBytes(state)
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
      InvMixColumns(state)
   end for

   InvShiftRows(state)                    // last round
   InvSubBytes(state)
   AddRoundKey(state, w[0, Nb-1])

   out = state
end
```

Figure.5. Pseudo Code for AES Decipher Process

1. **Byte Substitution**

Encryption and Decryption Transformation

The substitute byte transformation, called the byte sub, is a simple table lookup. The process is shown in the Figure.6. AES defines a 16X16 matrix of byte values, called an S-Box that contains a permutation of all possible 256 8-b values which is shown in the Table.1. Each individual byte of the state is mapped into a new byte in the following way: The leftmost 4-b of the byte value is used as a row value and the rightmost 4-b are used as a column value. These row and column values serve as indexes into the S-Box to select a unique 8-b output value. For example, the hexadecimal value {95} references row 9, column 5 of the S-Box, which contains the value {ad}. Accordingly, the value {95} is mapped into the value {ad}. The process of byte substitution is same for the decryption process but it makes use of inverse S-Box as shown in the Table.2, which is applied to each byte of the state [2].

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Table.1 AES S-Box

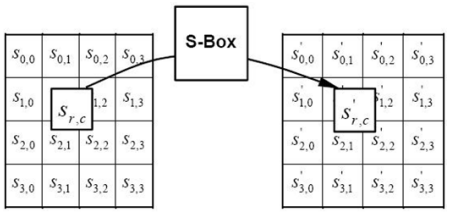


Figure.6 AES Byte Substitution Operation

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Table.2 AES Inverse S-Box

Shift Row

In this step, a normal left circular shift operation is done where in the first row is not altered and the next three rows are moved by 1, 2, 3 bytes respectively [4]. Conceptually, this is shown in the Figure.7.

Whereas in decryption transformation, the rows are subjected to right circular shift wherein the first row is untouched and row 2, 3 and 4 are shifted by 1, 2 and 3 bytes respectively. This process is shown in the Figure.8.

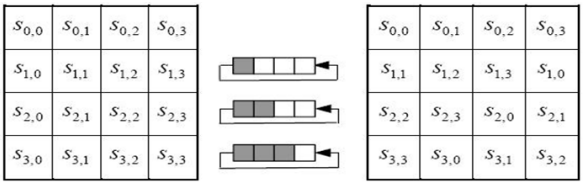


Figure.7 AES Shift Row operation

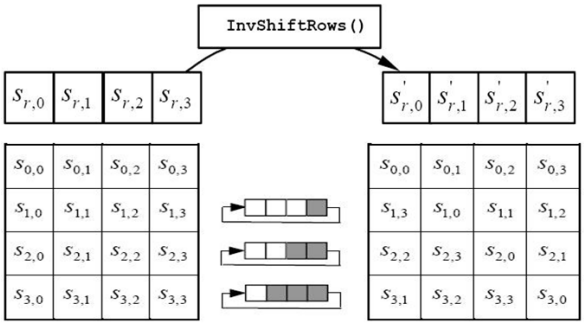


Figure.8 AES Inverse Shift Row operation

Mix Column

The mix column transformation operates on the state column-by-column, treating each column as a four-term polynomial. The column is considered as polynomial over GF (28) and multiplied with modulo x4+1 with a fixed polynomial a(x), given by

a(x) = {03}x3+{01}x2+{01}x+{02}
Let s`(x) = a(x) x s(x):

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

As a result of this multiplication, the four bytes in a column are replaced by the following: And the illustrates of Mix column transformation is shown in the Figure.9 for the encryption process.

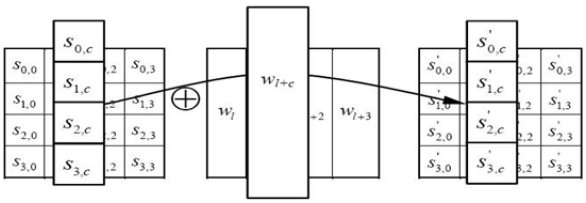


Figure.10 AES Add Round Key Transformation

$$s'_{0,c} = (\{0e\} \bullet s_{0,c}) \oplus (\{0b\} \bullet s_{1,c}) \oplus (\{0d\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$

$$s'_{1,c} = (\{09\} \bullet s_{0,c}) \oplus (\{0e\} \bullet s_{1,c}) \oplus (\{0b\} \bullet s_{2,c}) \oplus (\{0d\} \bullet s_{3,c})$$

$$s'_{2,c} = (\{0d\} \bullet s_{0,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0e\} \bullet s_{2,c}) \oplus (\{0b\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{0b\} \bullet s_{0,c}) \oplus (\{0d\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0e\} \bullet s_{3,c})$$

While for the decryption process the Inv mix column is inverse of the mix column transformation, where in it is multiplied with fixed polynomial a-1(x), given by
a-1(x) = {0b}x3+{0d}x2+{09}x+{0e}
Let s`(x) = a-1(x) x s(x):
As a result of this multiplication, the four bytes in a column are replaced by the following:

Add Round Key

In the Add round key transformation, the 128-b of the state is bitwise XORed with128-b of the round key. As shown in the Figure.10, the operation is viewed as a column wise operation between the 4-bytes of a state column and one word of the round key; it can also be viewed as a byte level operation. The Add round key operation in decryption transformation is same, as XOR operation is its own inverse [2], [4].

Key Expansion

The AES key expansion algorithm takes as input a 4-word (16-byte) key and produces a linear array of 44 words (176 bytes). This is sufficient to provide a 4-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher. The following pseudocode describes the expansion [6], [15]:

```
KeyExpansion (byte key[16], word w[44])
{
word temp
for (i = 0; i < 4; i++
  w[i] = (key[4*i],key[4*i+1],key[4*i+2],key[
for (i = 4; i < 44; i++)
  {
temp = w[i-1];
if (i mod 4 = 0)
temp = SubWord (RotWord (temp)) XOR Rcon[i/4];
  w[i] = w[i-4] XOR temp
    }
}
```

The key is copied into the first four words of the expanded key. The remainder of the expanded key is filled in four words at a time. Each added word w[i] depend on the immediately preceding word, w[i-1], and the word four positions back, w[i-4]. In three out of four cases, a simple XOR is used. For a word whose position in the w array is a multiple of 4, a more complex function is used.

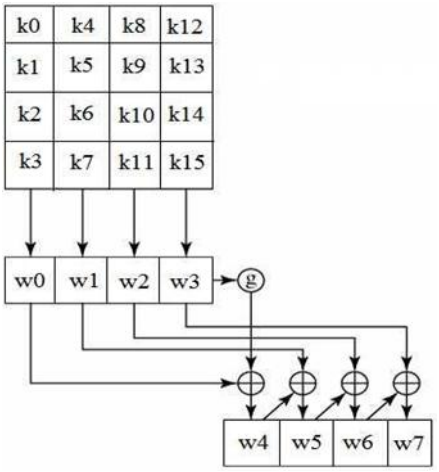


Figure.11 AES Key Expansion

Figure.11 illustrates the generation of the first eight words of the expanded key, using the symbol g to represent that complex function. The function g consists of the following sub functions:
1. RotWord performs a one-byte circular left shift on a word. This means that an input word [b0, b1, b2, b3] is transformed into [b1, b2, b3, b0].
2. SubWord performs a byte substitution on each byte of its input word, using the S-box.
3. The result of steps 1 and 2 is XORed with a round constant, Rcon[j] which is given in the Table.3.

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

Table.3 Round Constant Rcon[j]

4. RESULTS

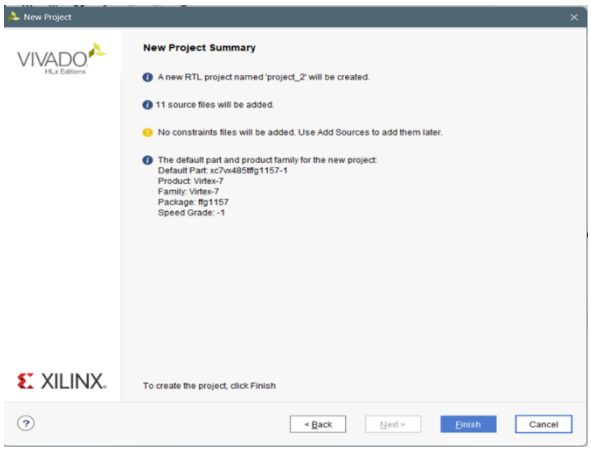


Figure 7.1 – New project summary

Figure 7.1 shows the **New Project Summary** window in the Xilinx Vivado HLx Editions software before the project is created. This summary provides an overview of the project configuration, including the project name (project_2), the number of source files to be added (11), and a notification that no constraint (.xdc) files have been included. It also displays the selected target FPGA device specifications: **Virtex-7 (xc7vx485tffg1157-1)** with package **ffg1157** and **speed grade -1**. After verifying these settings, the user can click the **Finish** button to create the project.

Figure 7.2: Project Summary in Xilinx Vivado

Figure 7.2 shows the Project Summary window in Xilinx Vivado for the AES design project. It displays the project settings, target FPGA device, synthesis and implementation status, and confirms that both processes were completed successfully with a few warnings.

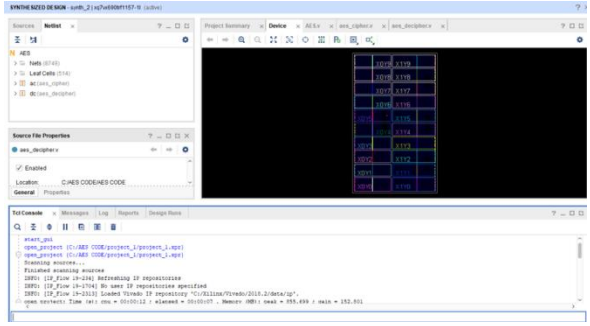


Figure 7.3: Post-Synthesis Design View in Xilinx Vivado

Figure 7.3 presents the post-synthesis view of the AES design in Xilinx Vivado. The synthesized netlist displays the top-level AES module along with its encryption (aes_cipher) and decryption (aes_decipher) submodules. The device view illustrates the synthesized design mapped onto the target FPGA device. The source file properties and TCL console confirm that the synthesis process was completed successfully without errors, making the design ready for implementation.

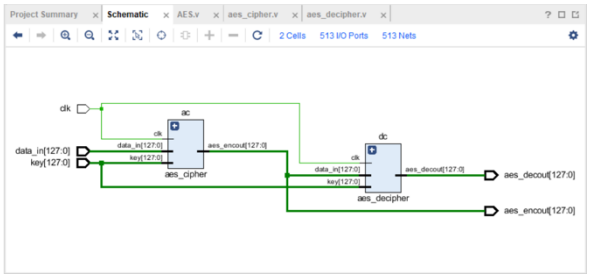


Figure 7.4: Schematic Representation of the AES Encryption and Decryption Modules

Figure 7.4 illustrates the synthesized schematic of the AES design generated in Xilinx Vivado. The diagram shows the interconnection between the aes_cipher and aes_decipher modules, where the encrypted output (aes_encout) from the encryption module is used as the input to the decryption module. Both modules share the same clock and 128-bit encryption key. The schematic verifies the correct data flow and connectivity of the complete AES encryption–decryption system.

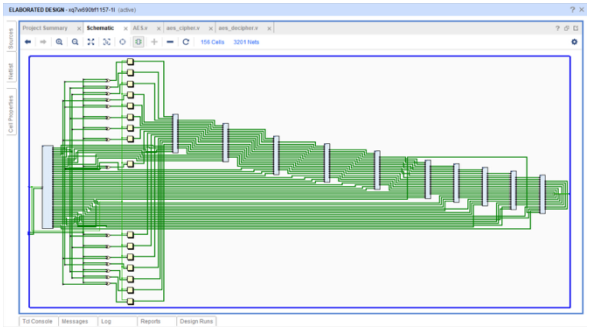


Figure 7.5: Elaborated Schematic of the AES Design in Xilinx Vivado

Figure 7.5 illustrates the elaborated schematic of the AES design generated in Xilinx Vivado before synthesis. It presents the internal logic structure, signal paths, and interconnections between the functional blocks of the AES module. The schematic provides a detailed view of the data flow through the encryption and decryption circuitry, enabling verification of the design hierarchy and connectivity. This elaborated view helps ensure that the HDL design is functionally correct prior to synthesis and implementation.

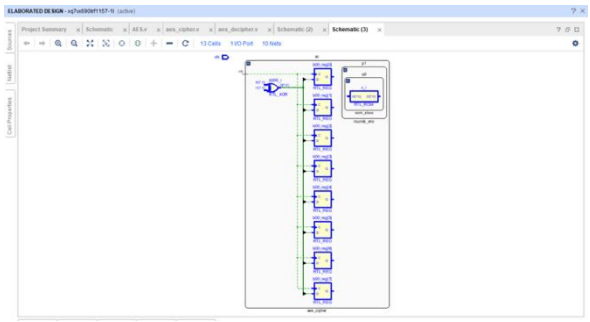


Figure 7.6: Elaborated Schematic of the AES Cipher Module

Figure 7.6 shows the elaborated schematic of the aes_cipher module generated in Xilinx Vivado. The schematic illustrates the internal architecture of the encryption module, including the XOR logic, register blocks (RTL_REG), and the S-box (RTL_ROM) used in the encryption rounds. It also depicts the signal flow and interconnections between the functional blocks. This view helps verify the correctness of the AES cipher architecture before synthesis and implementation.

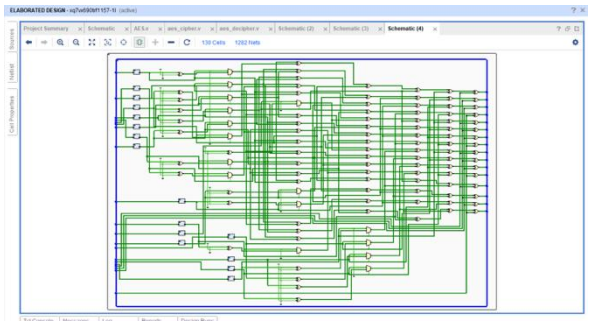


Figure 7.7: Elaborated Internal Logic Schematic of the AES Cipher Module

Figure 7.7 illustrates the detailed elaborated logic schematic of the AES cipher module generated in Xilinx Vivado. The schematic displays the internal combinational logic, gates, and signal interconnections that implement the AES encryption algorithm. It provides a low-level representation of the data paths and logic operations synthesized from the Verilog HDL design. This view is useful for verifying the internal functionality and correctness of the encryption circuitry before synthesis and implementation.

TECHNOLOGY SCHEMATIC

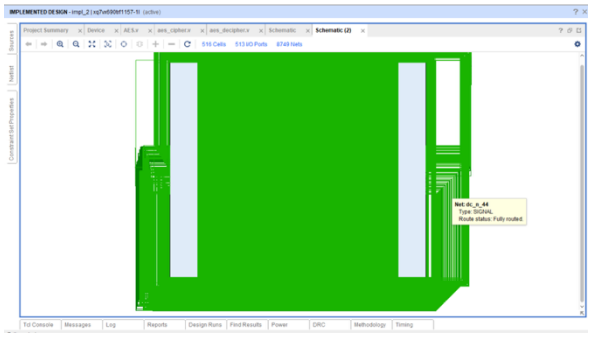


Figure 7.8: Implemented Design and Routing of the AES Module

Figure 7.8 illustrates the implemented design of the AES module after placement and routing in Xilinx Vivado. The routing view shows the interconnections of the synthesized logic mapped onto the target FPGA device using LUTs and other FPGA resources. The highlighted signal paths indicate that the design has been fully routed, confirming successful implementation. This technology schematic verifies that the AES design has been physically realized on the FPGA and is ready for bitstream generation and hardware deployment.

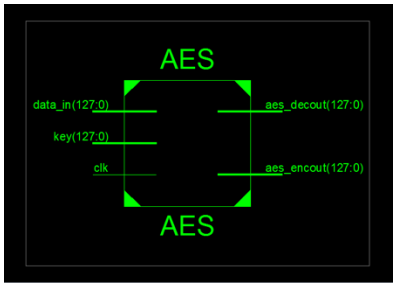


Figure 7.9: RTL Schematic of the AES Design

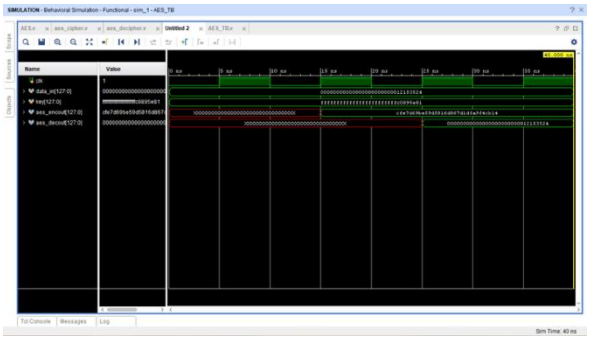


Figure 7.10: Behavioral Simulation Waveform of the AES Design

Figure 7.10 shows the behavioral simulation waveform of the AES design generated in Xilinx Vivado. The waveform illustrates the clock signal, 128-bit input data, encryption key, encrypted output (aes_encout), and decrypted output (aes_decout). The results confirm

that the input plaintext is successfully encrypted and subsequently decrypted back to its original value, verifying the correct functional operation of the AES encryption and decryption modules.

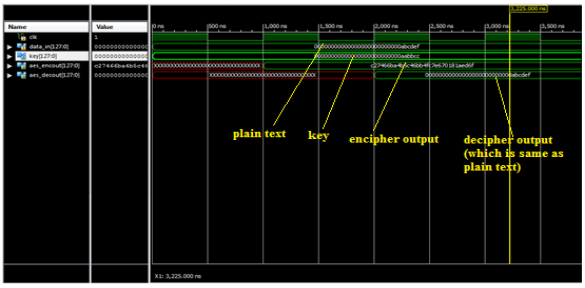


Fig: Simulated Waveforms of AES

5. CONCLUSION & FUTURE WORK

In this project, we have given 128 bits input and 128 bits security key and observed how it is delivered at the output with security. In this project there is no revealing of the original message to the hackers. The original message can be revealed to only sender and the receiver. So in future, any propriety information can be transmitted securely by using this project (military or banking purposes) and additionally AES is more secured .Modern applications of AES cover a wide variety of applications, such as secure internet, electronic financial transactions, remote access servers, cable modems, secure video surveillance and encrypted data storage. The future scope of our project is to extend 128 bits inputs to n bits (n is any integer value).

REFERENCES

[1] Madakam, Somayya, R. Ramaswamy, and Siddharth Tripathi. "Internet of Things (IoT): A literature review." Journal of Computer and Communications 3, no. 05 (2015): p.164.

[2] Wang, Yong, Garhan Attebury, and Byrav Ramamurthy. "A survey of security issues in wireless sensor networks." IEEE Communications Surveys Tutorial (2006).

[3] Veeramallu, B., S. Sahitya, and Ch LavanyaSusanna. Veeramallu, B., S. Sahitya, and Ch LavanyaSusanna. "Confidentiality in Wireless sensor Networks." International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-6, January 2013.

[4] Eisenbarth, Thomas, and Sandeep Kumar. "A survey of lightweightcryptography implementations." IEEE Design & Test of Computers 24.6 (2007).

[5] Banik, Subhadeep, Andrey Bogdanov, and Francesco Regazzoni. "Exploring energy efficiency of lightweight block ciphers." International Conference on Selected Areas in Cryptography. Springer, Cham, 2015.

[6] Bogdanov, Andrey, et al. "PRESENT: An ultra-lightweight block cipher." CHES. Vol. 4727. 2007.

[7] Borghoff, Julia, et al. "PRINCEa low-latency block cipher for pervasive computing applications." International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin Heidelberg, 2012.

[8] Beaulieu, Ray, et al. "The SIMON and SPECK lightweight block ciphers." Design Automation Conference (DAC), 52nd ACM/EDAC/IEEE. IEEE, 2015.

[9] Suzaki, Tomoyasu, et al. "TWINE: A Lightweight Block Cipher for Multiple Platforms." Selected Areas in Cryptography. Vol. 7707. 2012.

[10] Li, Wei, et al. "Security analysis of the LED lightweight cipher in the internet of things." Jisuanji Xuebao(Chinese Journal of Computers) 35.3(2012): p.434-445.

[11] Shibutani, Kyoji, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. "Piccolo: An ultra-lightweight blockcipher." In CHES, vol. 6917, pp. 342-357. 2011.

[12] Wu, Wenling, and Lei Zhang. "LBlock: a lightweight block cipher." In Applied Cryptography and Network Security, pp. 327-344. Springer Berlin/Heidelberg, 2011.

[13] Daemen, Joan and Rijmen, Vincent. "The design of Rijndael: AES-the advanced encryption standard.", Springer Science & Business Media, 2013.

[14] Descriptions of SHA-256, SHA-384, and SHA-512.
http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384- 512.pdf.

[15] Al Hasib, Abdullah, and Abul Ahsan Md Mahmudul Haque. "A comparative study of the performance and security issues of AES and RSA cryptography." Third International Conference on Convergence and Hybrid Information Technology, 2008. Vol.2.

[16] Feldhofer, Martin, Johannes Wolkerstorfer, and Vincent Rijmen. "AES implementation on a grain of sand." IEE Proceedings-Information Security 152, no. 1 (2005): p.13-20.

[17] Moradi, Amir, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. "Pushing the limits: a very compact and a threshold implementation of AES." In Eurocrypt, vol. 6632, pp. 69-88. 2011.

[18] Hocquet, Cdric, Dina Kamel, Francesco Regazzoni, Jean-Didier Legat, Denis Flandre, David Bol, and Franois-Xavier Standaert. "Harvesting the potential of nano-CMOS for lightweight cryptography: an ultra-lowvoltage
65 nm AES coprocessor for passive RFID tags." Journal o Cryptographic Engineering 1, no. 1 (2011): p.79-86.

[19] Kerckhof, Stphanie, Franois Durvaux, Cdric Hocquet, David Bol, and Franois-Xavier Standaert. "Towards green cryptography: a comparison of lightweight ciphers from the energy viewpoint." Cryptographic Hardware and Embedded SystemsCHES 2012 (2012): p.390-407.

[20] Batina, Lejla, et al. "Dietary recommendations for lightweight block ciphers: power, energy and area analysis of recently developed architectures."

International Workshop on Radio Frequency Identification: Security and Privacy Issues. Springer, Berlin, Heidelberg, 2013.
[21] Banik, Subhadeep, Andrey Bogdanov, and Francesco Regazzoni. "Exploring the energy consumption of lightweight blockciphers in FPGA." International Conference on ReConFigurable Computing and FPGAs (ReConFig), 2015 , pp.1-6.
[22] Kong, Jia Hao, Li-Minn Ang, and Kah Phooi Seng. "A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments." Journal of Network and Computer Applications 49 (2015): p.15-50.
[23] Wenceslao Jr, Felicisimo V., et al. "Modified AES Algorithm Using Multiple S-Boxes." The Second International Conference on Electrical, Electronics, Computer Engineering and their Applications (EECEA2015). 2015.
[24] Kawle, Pravin, et al. "Modified Advanced Encryption Standard." International Journal of Soft Computing and Engineering (IJSCE) 4 (2014).