Higher Level Synthesis Framework Implementation of

AES 128-bit Encryption Decryption

Afaq Younas — 051-16-124114

Muhammad Shaheer Aamir — 051 - 15 - 123404

Shah Nawaz — 051 - 15-123450

Mr. Arslan Majid

A Final Year Project Report is

Submitted in Partial Fulfilment of the

Requirements for the Degree of

Bachelor of Science in Telecommunication & Networks

Department of Computing & Technology

Iqra University, Islamabad Campus

April 2019

Certificate

We hereby accept the work contained in this report titled: ***Voice Encryption over IP Network***, as a confirmation to the required standards for the partial fulfillment of the degree of Bachelors of Telecommunication and Networks.

Internal Examiner

External Examiner

Project Supervisor

Head of Department

Declaration

We hereby declare that this work, neither whole nor in part, has been copied from any source. It is further declared that I have prepared this report entirely on the basis of my personal efforts made under the sincere guidance of teachers especially my supervisor Mr. Arslan Majid. If any part of this thesis is proved to be copied out from any source or found to be reproduction of some other, I will stand by the consequences. No portion of the work presented has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Afaq Younas

Muhammad Shaheer Aamir

Shah Nawaz

Dedication

We dedicate this project to our parents who did financial and moral support. We would also like to dedicate this project to our supervisor Mr. Arslan Majid.

Acknowledgements

We would definitely like to offer our thanks to ALLAH for his help and blessing throughout our studies. We might also want to offer a huge thanks to our supervisor, mentor, Mr. Arslan Majid for helping and guiding us till the end of this project. His specialized and research guidance was fundamental to the finishing of this FYP and has shown our countless lessons and bits of knowledge on the operations of academic research. Lastly, we would extend our thanks to our family for their unconditional support and immeasurable love.

Abstract

Cryptography assumes a vital job in security of information transmission. The improvement of registering innovation forces more grounded necessities on the cryptography plans. In 2000, the Advanced Encryption Standard (AES) supplanted the DES to conquer the expanding prerequisites for security. In cryptography, the AES, likewise called as Rijndael, is a square figure that is received as an encryption standard by the USA government, which determines an encryption calculation fit for securing private and touchy data. This calculation is a symmetric square figure that can encode and unscramble the data. Encryption changes over information into a garbled structure known as figure content. Unscrambling of the figure content believers the information once again into its unique structure, that is called plaintext. The AES calculation is underpinning keys length of 128, 192, and 256 bits to encode and decode information in squares of 128 bits, along these lines the name moves toward becoming AES-128, AES-192 and AES256 individually. The equipment execution of the AES calculation can give elite, ease for explicit applications and dependability contrasted with its product partners.

As the need of more and more cryptographic systems is emerging there is also an area of concern of computational power and response in terms if speed. A frame work developed using a parallel processing network is developed which performs the encryption and decryption process at a much faster rate. Coding for the frame work was done by using a higher level synthesis tool. The synthesis tool then converted the blocks coded in C into synthesizable Verilog module. The modules were then tested in waveform analyzers and compared with standard open source Verilog implementation.

Experiment results revealed that our methodology produced correct output results whereas it achieved a slightly better speed due to its parallel processing design.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1. Overall Description

It is a research-based project about implementation of Voice Encryption and decryption using AES Algorithm.

1.1.1. Objectives

Project is about giving a speech signal as input to an Analog to Digital Converter (ADC) at the transmitter end which will covert voice into its digitized form and then voice samples will be stored it on FPGA Block rams. The voice will be decrypted at transmitter end. After sending it through a wired communication channel (Ethernet cable), it will be received on second FPGA the data will be decrypted and later convert from binary to speech signal through Digital to Analog Converter (DAC). We will work on two approaches namely

- Offline Method (recorded Voice Samples Stored in memory)
- Real-Time Method (Taking input from microphone and listening to output on Speaker)

Figure 1.1: General Overview

1.1.2. Problem Description

Encryption is an efficient method for protection of speech communications. Voice Encrypt or digitize the conversation at transmitter end and apply a cryptographic technique to the resulting bit-stream. In order to decipher the speech correct encryption scheme must be used. Voice Encryption helps us in private and confidential manner. It is nearly impossible to decrypt voice into its original form again. We will compare the efficiency and performance of standardized C/C++ implementations and open source Verilog codes with ours and compare the results.

1.1.3. Methodology

We will first write code for our algorithm in higher level language usually in c or C++.The C implementation will be synthesized into its equivalent Verilog implementation using higher level synthesis tool. Xilinx VIVADO HLS can convert into a higher language code into its parameterized Verilog modules. After that performance comparison will be done in terms of efficiency and latency.

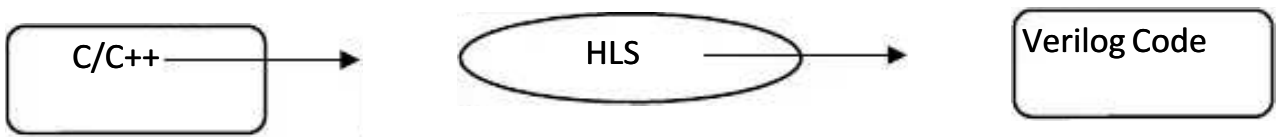


Figure 1.2: High Level Synthesis

1.1.4. Product Scope

Privacy is everyone's need nowadays. Through our project, we are devoted to make it better by figuring out about the performance and speed of AES Encryption Algorithm in Voice Encryption.

1.1.5. User Classes and Characteristics

A public key and private key will be used. It's a choice that whether public key or private key is being used on sender end and receiver end will use the alternate key. Private Key should not be compromised.

1.1.6. Operating Environment

The project will operate in Visual studio, ISE design and VIVADO HLS, including the hardware platform of Spartan 6.

1.1.7. Assumptions and Dependencies

Main limitations in **AES Algorithm** are that there are some input parameters like generating a 10 round key. It makes it hard to determine key. Details are given below:

- Message is encrypted with cipher key using XOR.
- There are 4 transformations applied to the encrypted message in each round which are Sub Bytes, Shift Rows, Mix Columns and Add Round Key.
- In the final round Mix Columns is not applied.
- For Sub Bytes, S-Box is used to provide confusion capability.

left (high-order) nibble \ right (low-order) nibble	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

Figure 1.4: S-Box

- For Shift Rows, we rotate 1 byte in 2nd row, 2 bytes in 3rd row and 3bytes in 4th row.
- In Mix Columns, each column obtained from Shift Rows is multiplied with Rijndael's Galois Field.

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

- Add Round Key involves adding the obtained state from Mix Columns with the round key using a special type of arithmetic.

1.2. External Interface Requirements

1.2.1. User Interfaces

User interface will be a C/C++ console using Microsoft Visual Studio.

1.2.2. Hardware Interfaces

Hardware interface that can be used for this project are minimum core i-3 laptop or computer for Microsoft Visual Studio and Xilinx VIVADO, a medium like Ethernet cable for IP and a SPARTAN 6 FPGA board for implementing it.

1.2.3. Software Interfaces

Software interface that can be used for this project is Xilinx VIVADO. It will exponent **e** in high-level synthesis tool, VIVADO is capable of converts **C** Code to equivalent RTL (register transfer level) Verilog code.

1.2.4. Communications Interfaces

Standard TCP/IP will be used for communication Interfaces. Communication standards can be used, such as FTP or HTTP.

1.3. System Features

The main System feature is performing encryption and decryption. System will perform encryption on first block of text and so on.

1.3.1. AES Encryption and Decryption Design Feature

1.4.1.1 Description and Priority

We take a 128-bit plaintext and 128-bit key for processing in Decryption block to generate a cipher text. Priority for the feature will be high.

1.4.1.2 Stimulus/Response Sequences

By processing we generate response number and their waveforms.

1.4.1.3 Functional Requirements

It will ask for 128-bit number. By getting 128-bit number and key system will be able to create Private key to Cipher.

REQ-1: 128-bit plain text
REQ-2: 128-bit key

1.4. Nonfunctional Requirements

1.4.1. Performance Requirements

AES Algorithm Matrix for Intel Dual-Core i7-4500U 1.80GHz is:

Bits	Time
128	Less than 2 sec
192	16 sec
256	35 min
260	1 hour

1.4.2. Safety Requirements

Main safety requirements include damage, harm or possible loss that could result from the use of this product is increasing temperature of device. Safeguards or actions that must be taken are to keep device on a proper temperature.

1.4.3. Security Requirements

The security requirement that is needed is that Private Key is not compromised it must be a secret.

1.5. Scenarios

There are three scenarios of AES system design 128 bit, 192 and 256-bit numbers. In this system design we can only use 128-bit number.

1.6. Report Structure

Overall structure of this report includes description of AES, implemented procedures and techniques. In chapter 1 we define methodology assumptions, dependences, system features and functional, non-functional requirements. Chapter 2 contains terminology, categorization of existing techniques, limitations and proposed improvements. Chapter 3 includes system design, Chapter 4 Implementation, Chapter 5 testing of implemented techniques, Chapter 6 Date Analysis of implemented and existing techniques and overall conclusion of whole project and ideas on future work in Chapter 7.

Chapter 2 Literature Review

2.1. Introduction

The uttermost important aspect in our daily life is communication. The main issue in our communication is security to preserve its integrity, availability and proper access control as well as confidentiality. Therefore, Communication protection is essential from misuse. Voice encryption and Decryption is a research-based project about implementation of

Encryption and Decryption of voice signal over an IP network using AES algorithm. Encryption is efficient method that is used for security of communications.

2.2. Related Works

2.2.1. Terminology

Existing Voice encryption techniques are implemented in C/C++ and Verilog. We can use these existing techniques for our research and convert the existing C/C++ code in Verilog by synthesizing it by VIVADO then comparing it to existing Verilog and C/C++ techniques by speed testing, complexity and security.

Figure 2.1: High Level Synthesis Terminology

2.2.2. Existing Implementations

As Rijndael has existed as the Advanced Encryption Standard since 2000, there are already many implementations of the algorithm. These implementations are available in many different programming languages. As the AES standard is open, organizations or users which wish to implement the Rijndael algorithm are free to do so.

2.2.3. Categorization of Existing Techniques/Works/Research

Encryption techniques as far as hardware implementations are concerned can be broken down into following categories

a. Implementation in C/C++

b. Implementation in Java

c. HDL based Implementations

d. GPU based implementation

2.2.4. Languages

As previously stated, this project requires the implementation of the Rijndael's algorithm in systems constructed with several different programming languages. Therefore it is important to examine the history and nature of these languages, so as to understand how they might be used and how cryptography might be used in a system developed in that language. The languages to be used in this project are Java, C, JavaScript and Perl.

2.2.4.1. Java

Java is an object-oriented language that was created and developed by software engineers at Sun Microsystems during the early 1990s. Code written in Java is compiled into byte code which can then be executed on the Java Virtual Machine. In practice this means that programs written in Java can be run on any computer which supports a JVM, regardless of the underlying hardware or operating system, which is a key feature of the language. While Java can be used to create desktop applications, it can also be used to create Java applets. These are applications which can be distributed over the internet and run within a web browser. Therefore a potential application for cryptography would be an applet which encrypts data on the client's computer before submitting it to a server.

2.2.4.2. C

The programming language C was originally designed in 1972 and used as the systems language for UNIX. It gradually evolved in the years following until in 1990 the American National Standards Institute (ANSI) set the standard for the language, known as ANSI C. C is a relatively low level programming language that gives the programmer more control over aspects such as memory allocation. Like Java, C has been widely used and has a variety of potential applications, but for this language, a suitable program might be one that encrypts or decrypts files from a command prompt.

2.2.4.3. Perl

Perl is a scripting language like JavaScript, and grew because of its strengths in text manipulation. Perl is often regarded as a general purpose programming language which can be used to automate a variety of different tasks. While there is no 'typical' Perl system, one which used cryptography might do so for the encryption of data being passed between other systems.

2.2.4.4. JavaScript

JavaScript is a scripting language that was created in the early 1990s. It is commonly used on the internet to provide client-side functionality in web pages. It has no technical relation to Java, although both languages have a similar syntax. A system using JavaScript on the internet might make use of cryptography when displaying encrypted content to a user.

2.2.5. Limitations/Gaps within Existing Techniques/Works

The implementations are done in C, C++, Perl, Java and JavaScript is purely for simulation purposes and have no relevance to hardware in general. The category C and Java involve no usage of hardware but in GPU based implementation are still in beginning stage. So we are left with one option that is HDL based implementation. Current project will minimize the gap of C and HDL as main algorithm blocks will be done in higher level language will be ported to synthesizable HDL so that they can function as per requirement.

• Key Creation Complexity

AES algorithm is limited because of 10 rounds of encryption.

2.2.6. Comparative Analysis of different AES implementations

Encryption has a strong presence in today's digital electronics with the frequent transmission and storage of sensitive data. AES as the standard encryption algorithm and it is commonly used as a fast solution to secure data. When designing VLSI systems, the task of balancing the area, power, and speed is a challenge; hardware encryption is no different. System requirements drive certain performance parameters to the forefront, identifying how to alter design implementations to meet performance requirements is not always apparent. Multiple resources in this research field have identified AES algorithm features of interest and discussed their impact a few of the design trade spaces, however, a single comparative analysis was lacking. This project explores six different AES features key size, mode specificity, round key storage, round unraveling, SBOX implementation, and pipelining. A summarized view of the resulting designs allows readers to quickly analyze how each of the six features impacts speed, power, area, latency and throughput.

2.3. Proposed Improvements in Existing Works

The implementation done in higher level language is just for simulation or observation and has no relevancy to hardware design constraints in general. So we will first take c blocks of the proposed encryption technology. VIVADO has the capability to convert the c language code into a synthesizable Verilog code. Figure 2.3 is the screenshot of a general VIVADO

HLS interface. In the image a C language will be converted into a register transfer level Verilog code.

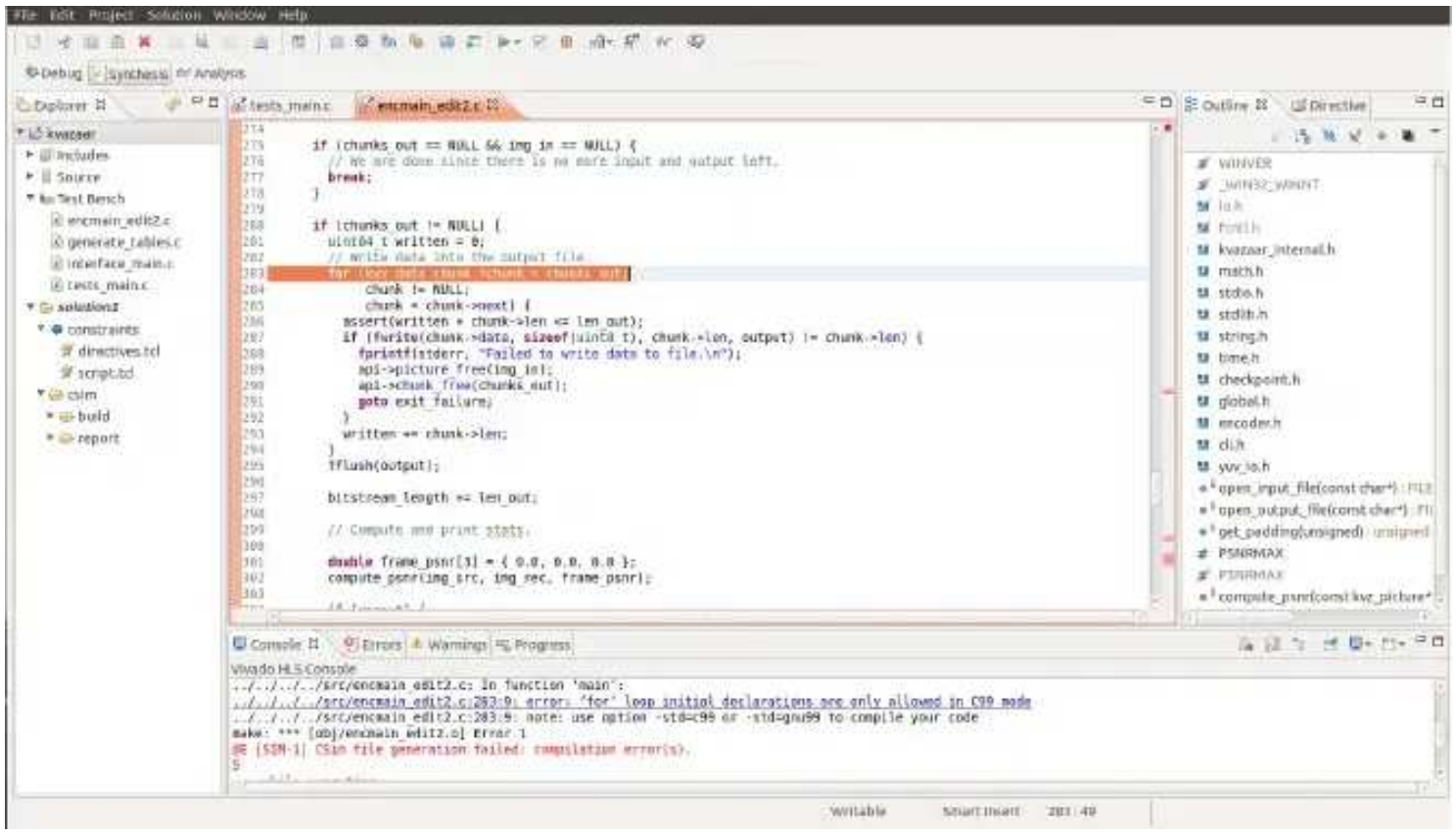


Figure 2.2: Starting screen of VIVADO HLS to open or create a project

2.4. Summary

Unwavering quality, openness, mystery and secretly of correspondence are the primary angles that would be kept up in voice security. Shielding voice frameworks from disturbance and adjustments just as the illicit access is the primary objective of voice security. A productive and secure correspondence framework for voice flags that will base on AES open key cryptosystem is structured in this work. The introduced cryptosystem is executed and its execution is assessed utilizing diverse voice quality measurements in both encryption and unscrambling forms.

Chapter 3 System Design

3.1. Introduction

Cryptography assumes an essential job in security of information transmission. The improvement of processing innovation forces more grounded prerequisites on the cryptography plans. In 2000, the Advanced Encryption Standard (AES) supplanted the DES to beat the expanding prerequisites for security. In cryptography, the AES, likewise called as Rijndael, is a square figure embraced as an encryption standard by the US government, which indicates an encryption calculation fit for securing delicate data. This calculation is a symmetric square figure which can encode and decode the data. Encryption changes over information into an ambiguous structure known as figure content. Decoding of the figure content believers the information again into its unique structure, that is called plaintext. The AES calculation is under-pins keys length of 128, 192, and 256 bits to scramble and decode information in squares of 128 bits, consequently the name progresses toward becoming AES-128, AES-192 and AES256 separately. The equipment execution of the AES calculation can give superior, minimal effort for explicit applications and dependability contrasted with its product partners.

3.1.1. Purpose

Out proposed project is about implementation of Voice Encryption and Decryption over IP Networks using AES Algorithm. Encryption is an efficient method for protection of speech communications. Voice Encrypt and digitize the conversation at transmitter end and apply a cryptographic technique to the resulting bit-stream. For decrypting the speech correct encryption scheme must be used. Voice Encryption helps us in private and confidential manner. It is nearly impossible to decrypt voice into its original form again.

3.1.2. System Overview

By giving a speech signal as input to an Analog to Digital Converter (ADC) at the transmitter end which will covert voice into its digitized form and then voice samples will be stored it on FPGA Block rams. The voice will be decrypted at transmitter end. After sending it through a wired communication channel (Ethernet cable), it will be received on second FPGA the data will be decrypted and later convert from binary to speech signal through Digital to Analog Converter (DAC). We will work on two approaches namely

- Offline Method (recorded Voice Samples Stored in memory)
- Real-Time Method (Taking input from microphone and listening to output on Speaker)
-

3.1.3. Design Map

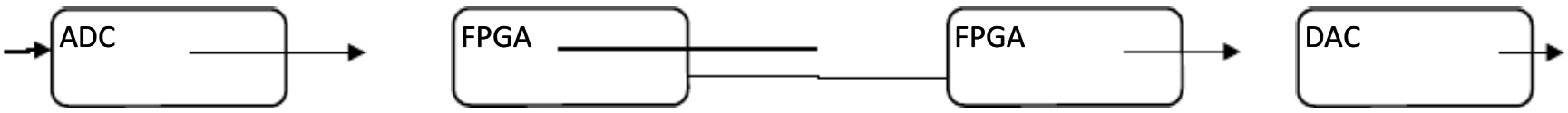


Figure 3.1: Design Map

3.1. Design Considerations

Design considerations like intended security and key's expected time were handled.

3.1.1. Assumptions

The main assumption of AES that the numerical problems are too large to be solved in a sensible time by attackers utilizing advance computer technology.

3.1.2. Constraints

Main constraint of **AES** Encryption and Decryption is that if number of computations exceeds from RAM storage or RAM storage is low then resource will also proceed from FPGA limits.

3.1.3. Design Methodology

Following Steps are followed for design methodology.

Step 1- Taking a C language code of AES encryption and decryption.
Step 2- Simulation
Step 3- Converting C language code to Verilog code.
Step 4- Comparing existing codes to new generating code.
Step 5- Performance and execution time checking.

3.1.4. Risks and Volatile Areas

No risk and volatile areas are involved.

3.2. Architecture

The architecture gives the high level design view of a system and gives a foundation for more specific design work.

3.2.1. Oerview

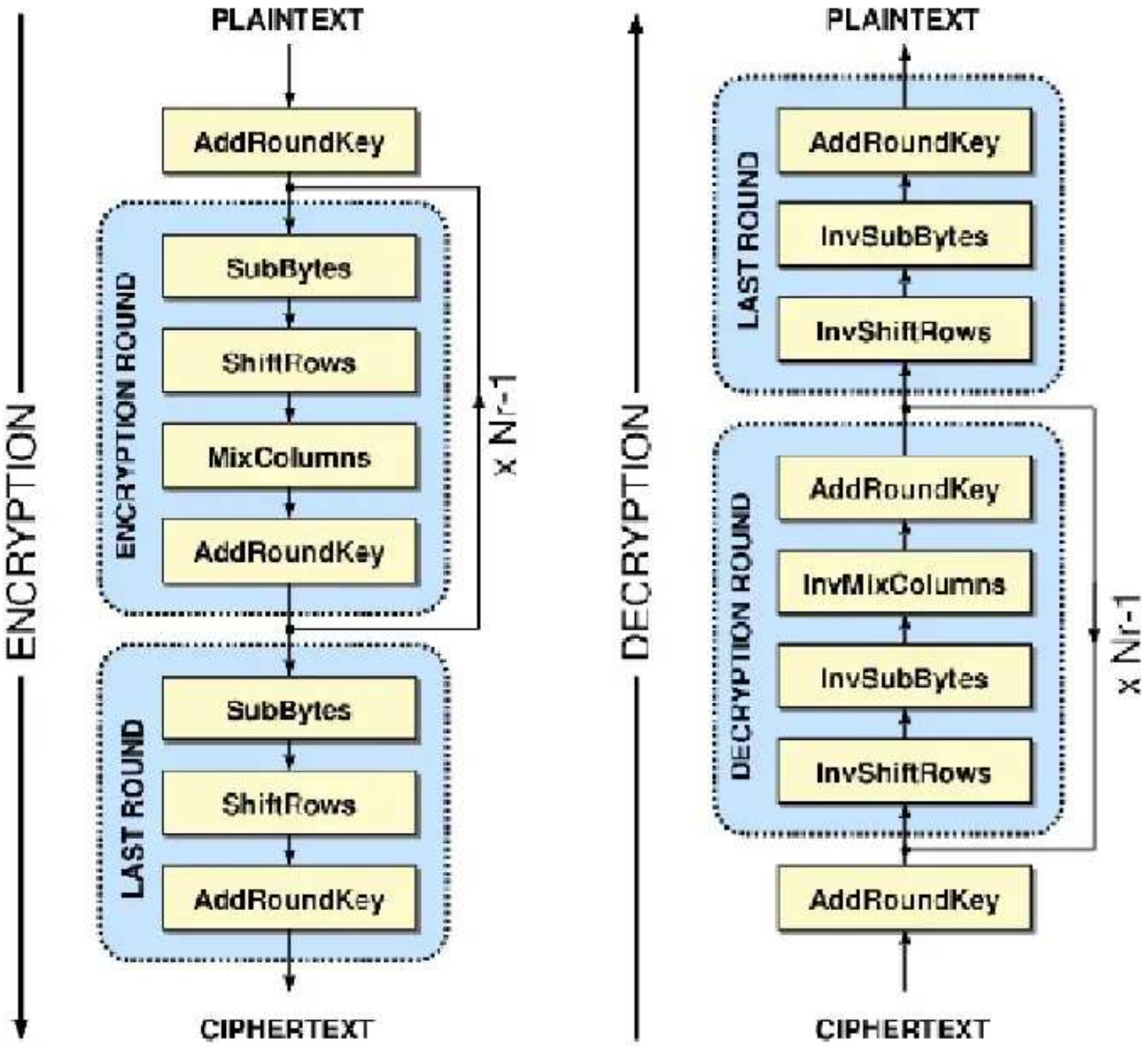


Figure 3.2: AES Architecture

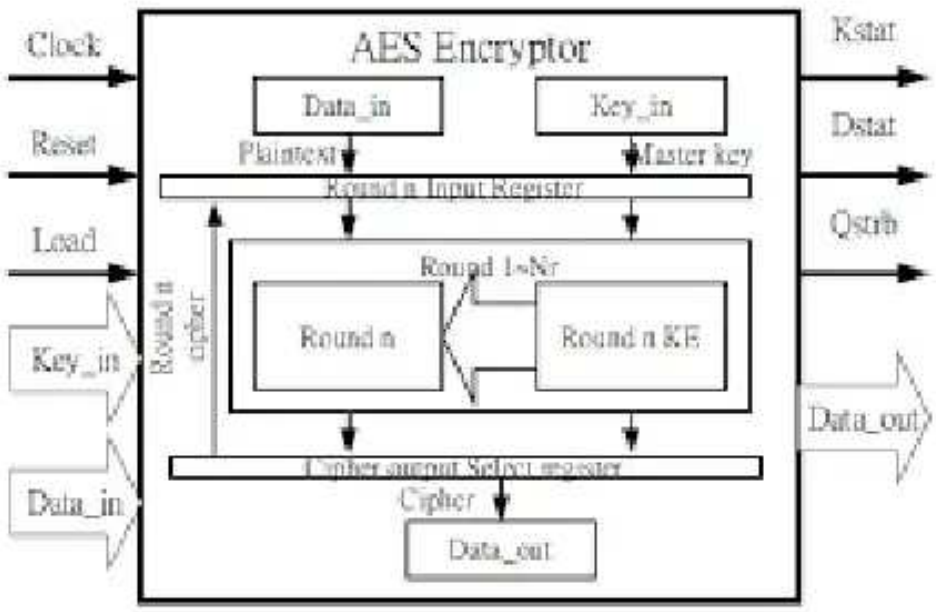


Figure 3.3: AES Encryption Description

3.2.2. Components, Subsystems or Modules 1 …N

Encryption Block

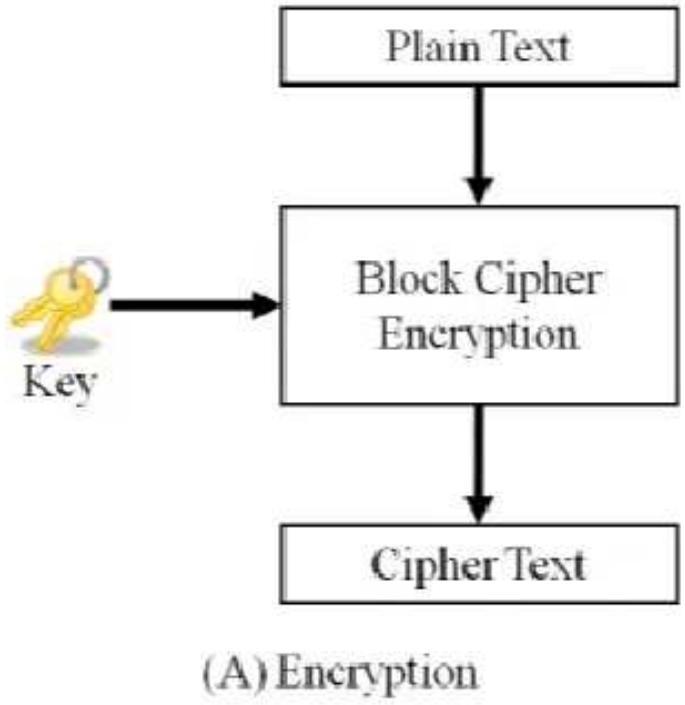


Figure 3.4: Encryption Block

Decryption Block

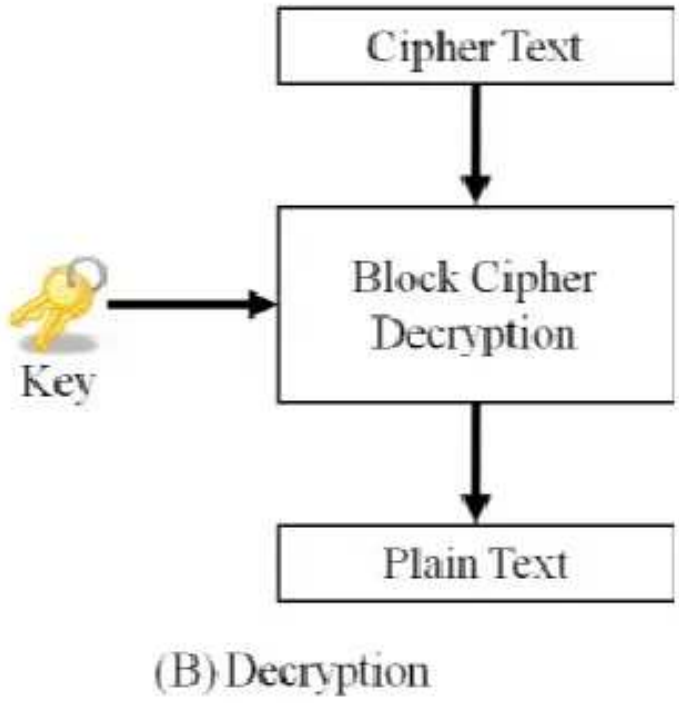


Figure 3.5: Decryption Block

3.2.3. Strategy 1…N

Followings are **AES** key generation algorithm steps:

1. From cipher key obtain the set of round keys.

2. Initialize the state array with the plaintext.

3. Add initial round key to the starting state array.

4. Perform nine rounds of state control.

5. Perform the tenth and final round of state manipulation.

6. Duplicate the final state array out as the encrypted data (cipher text).

Each round of the encryption procedure requires a series of steps to adjust the state array. These steps involve four operations:

- Sub Bytes
- Shift Rows
- Mix Columns
- XOR Round Key

AES Encryption algorithm:

1. Plain text M has been defined.

2. Cipher text C can be created using operations of state array.

AES Decryption algorithm:

1. Cipher text C has been received.

2. Plain text can obtain by performing nine full decryption rounds.

- XOR Round Key
- Inverse Mix Column
- Inverse Shift Rows
- Inverse Sub Bytes

Perform final XOR Round Key
The same round keys are used in the same order.

3.3. AES Modes of Operation

A mode of operation describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block. Five modes of operation of the AES algorithm were standardized: ECB (Electronic Code Book), CBC (Cipher Block Chaining), CFB (Cipher Feed Back), OFB (Output Feed Back) and CTR (Counter).

ECB (Electronic Code Book)

The ECB (Electronic Code Book) mode of operation is the simplest of all. A block scheme of this mode is presented in Figure 3.6.

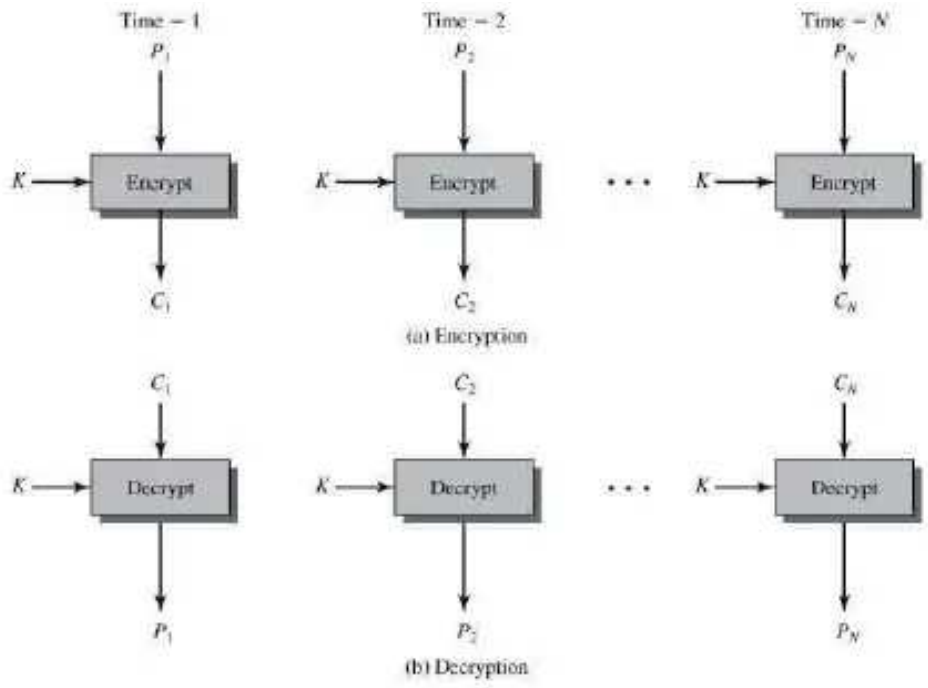
As it can be seen from Fig. 3.6, the plaintext message is divided in blocks (P1, P2, PN), where each block is encrypted separately with the same key (K). The results of the encryption are the encrypted messages C1, C2 and CN respectively.

If the size of the message is larger than n blocks, the last block is filled with padding. In this mode, if an error occurs in one of the blocks, it is not propagated to the other blocks, which is why decryption is possible in the blocks that don’t contain an error.

The encryption in this mode is deterministic, because identical P blocks will produce identical C blocks, which is why identical plaintext blocks or a message with the same beginning are

easily recognizable. Also, the ordering of the C blocks can be changed without the receiver noticing. In general, this mode is not recommended for encryption of data that is larger than one block.

Figure 3.6: Scheme of ECB



CBC (Cipher Block Chaining)

In order to provide cryptographic security, every encryption of the same plain text should result with a different cipher text. The CBC (Cipher Block Chaining) mode of operation (Fig. 3.7) provides this by using an initialization vector labelled as IV. The IV has the same size as the block that is encrypted.

Fig. 3.7 presents the encryption process. First, an XOR operation is applied to the plaintext block (P1) with the IV, and then an encryption with the key (K) is performed. Then, the results of the encryption performed on each block (C1, C2, … , CN-1) is used in an XOR operation of the next plaintext block PN which results in CN. In this way, when identical plaintext blocks are encrypted, a different result is obtained. Also, using a different IV for each new encryption, an identical message will always be encrypted differently. It should be emphasized that the same key K is used in each of the encryption blocks.

An error in one of the plaintext block will propagate in all the following blocks and will be manifested in the process of the description. Specifications is recommended that the Padding method 2 is used in case padding is needed with the CBC mode of operation because it provides protection from some of the known PA (Padding Attacks).

There are complex CBC attacks for which an unpredictable value of IV is needed in order to overcome them. In it is emphasized that the CBC mode of operation is safe from CPA (Chosen

Plaintext Attack) attacks (attacks in which the attacker chooses a set of plaintexts and is able to obtain respective cipher texts) only if the IV has a random value, but not if the IV is a nonce (a number that is not repeated). The CBC mode of operation, besides its vulnerability to PA attacks, is also easily susceptible to CCA (Chosen Cipher text Attack) attacks (where the attacker chooses a set of cipher texts and is able to obtain respective plaintexts). According to [3], the encryption key has to be changed whenever condition holds:

$$q << 2^{(n+1)/2}$$

In this equation, q is the number of blocks that should be encrypted and n is the number of bits in the encryption blocks.

In order to provide protection from CCA attacks in this mode of operation, it is necessary to use AE (Authenticated Encryption), where, besides the encryption, authentication is also performed.

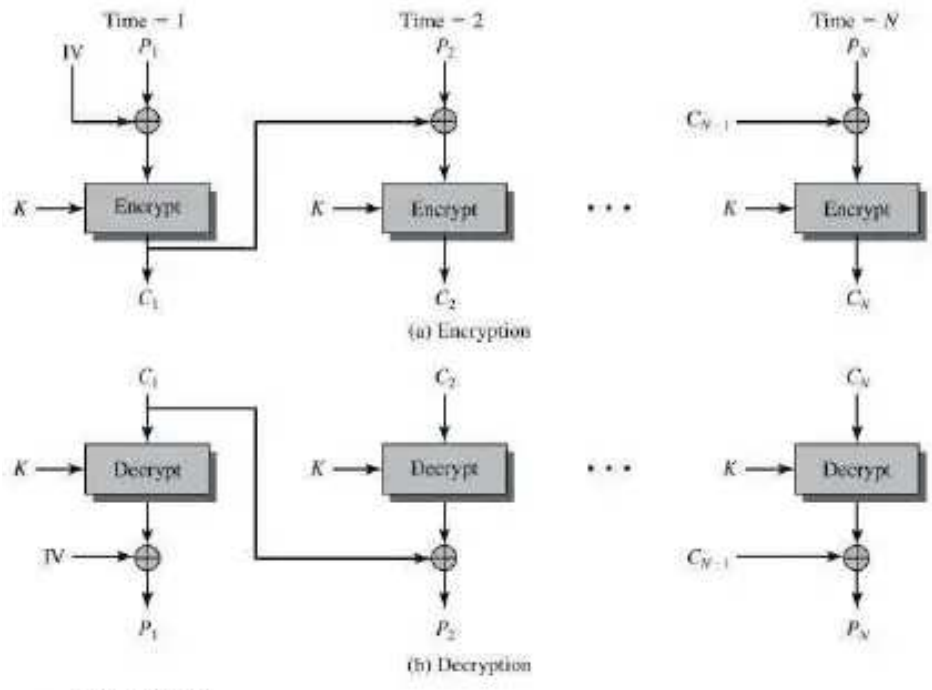


Figure 3.7: Scheme of CBC

CFB (Cipher Feed Back)

The CFB (Cipher Feedback) mode of operation allows the block encryptor be used as a stream cipher. The scheme of the CFB mode of operation is given in Fig. 3.8. As can be seen in Fig. 3, in the CFB mode of operation, at the beginning (at the first block) the encryption (uses an encryptor denoted with Encrypt) is performed by using an IV and an encryption key K. After that, the XOR operation between the encryption result (the output form the encryptor) and the plaintext block (P1) is performed. For all the other blocks, the encryption is performed over the result of the encryption of the previous blocks accordingly (C1, C2, ...). Then an XOR

is performed with the corresponding plaintext block (P2, P3, ...). In the beginning, the IV is placed in a shift register, the size of which can be e.g. 64 bits. The result of the encryption of the IV is again 64 bits. But, the XOR is applied to only a few bits (for example, s=8) of the encrypted IV with also s bits from the plaintext P1. The least significant bits from the IV that will not be used are discarded. The result C1 from the XOR operation is then placed at the rightmost position in the shift register from the next block, and the operation is repeated in the same manner. The encryption and decryption operations in the CFB mode of operation are the same operations. Also, an error in one block will propagate to the next block, which is manifested in the process of decryption.

In it is pointed out that the IV should be a unique identifier, e.g. a counter, whereas in it is stated that the value of the IV should be a nonce. The CFB mode of operation is safe form CPA attacks only if the IV has a random value, and is not safe if the IV is a nonce. Moreover, this mode of operation is not safe from CCA attacks. According to, the encryption key needs to be changed each time condition $q << 2^{(n+1)/2}$ holds.

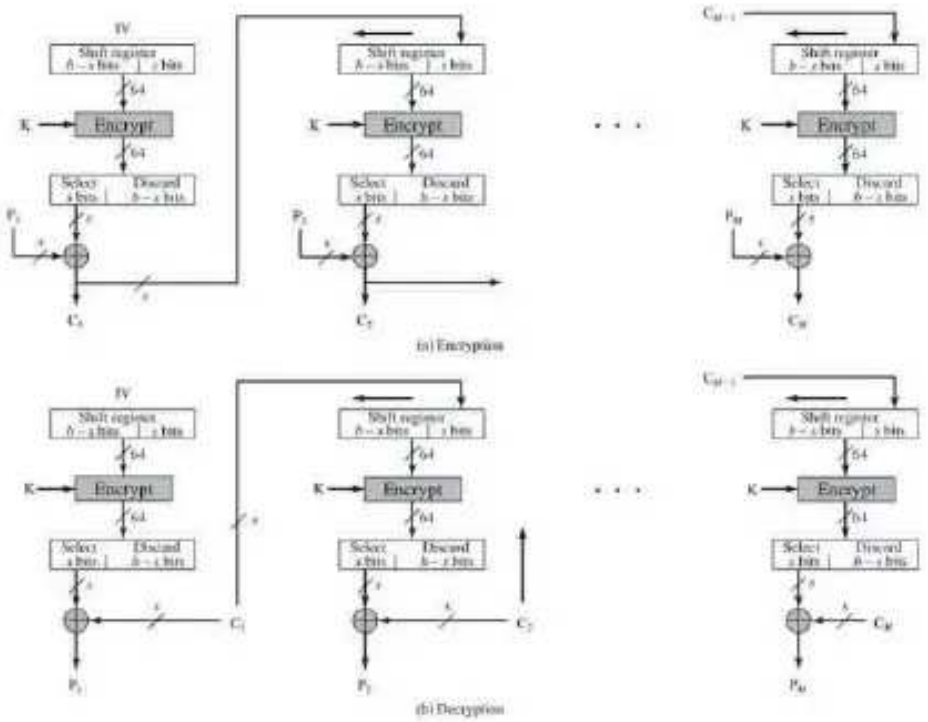


Figure 3.8: Scheme of CFB

OFB (Output Feed Back)

The OFB (Output Feedback) mode of operation (Fig. 3.9) also enables a block encryptor to be used as a stream encryptor. As shown in Fig. 3.9, the difference between the CFB and OFB mode is such that, in the case of an OFB, as an input for the shift register from the next block, the output from the encryptor (Encrypt) from the previous block is chosen. At the same time, the XOR operation with the s-bits of plain text P uses only s bits from the encryptor. Encryption and decryption are the same operation. If there is an error in a block during the

encryption, while performing the decryption, it will influence only a part of the plain text that will result from that block, i.e. there is a limited propagation of error. Therefore, this mode of operation is often used in communication through media that carry noise (for example, satellite communications).

According to, the IV should be a nonce. The guidelines given in suggest that the IV should be chosen randomly and used only once with the given encryption key K. security does not exist if the IV is a nonce, but the sequence generated by some counter is acceptable. The CFB mode of operation is vulnerable to attacks performed by modification of bits in the encrypted stream. To provide security in the OFB mode of operation, the encryption key K should be changed for every 2n/2 encryption blocks, where n is the number of bits in the block. But, OFB does not offer security from CCA attacks.

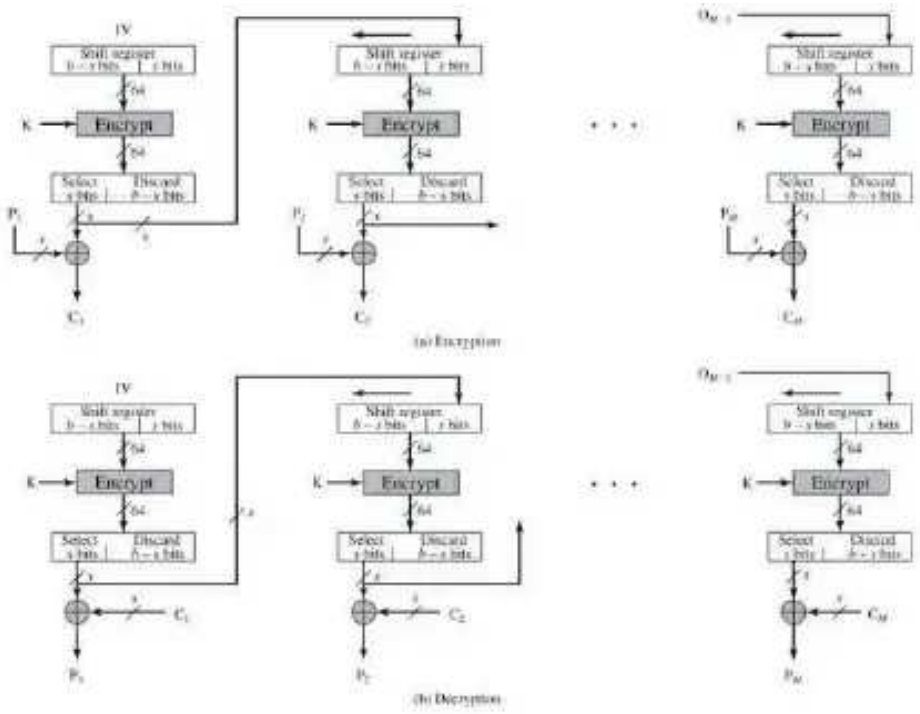


Figure 3.9: Scheme of OFB

CTR (Counter)

At the CTR (Counter) mode of operation, shown on Fig. 3.10, as an input block to the encryptor (Encrypt), i.e. as an IV, the value of a counter (Counter, Counter + 1, … , Counter + N 1) is used. The counter has the same size as the used block. As shown in Fig. 3.10, the XOR operation with the block of plain text (P1, P2, … , PN) is performed on the output block from the encryptor. All encryption blocks use the same encryption key K. If the last block of clear text PN has the number of bits smaller than the number of bits in the block, then only the s most significant bits for the XOR operation on the block PN are used from the output block of the encryptor. The remaining bits are discarded. Hence, as it is pointed out in [4], there is no need for adding bits (padding) to the last block. Values of the counters are independent from the output of the previous block; therefore, there is no propagation of error from one block

to another. Considering the independence of the blocks, this fact allows for parallelism in the encryption and the decryption, and there is also the possibility of preprocessing the values of the encryptors, which speeds up the process.

The encryption and decryption operations at the CTR mode of operation are the same. The counter sequence should be different for every block. On the other hand, it is suggested that the same value of the counter (Counter, Counter + 1, Counter + N - 1) and the same key K should not be used in the encryption of more than one block of data. If this requirement is not upheld, the plaintext can be revealed by performing the XOR operation on the two blocks of text encrypted by the same set of parameters. In that case, there is a complete breach of privacy. Usually the counter is initialized to some value, and then it is incremented by one for every block. Counter is a non-repeatable number on the order of 96 bits. The other 32 bits are zero at the beginning of the process, and then their values are incremented by 1 for every block. The guidelines found in recommend using a unique value for the Counter, chosen in a random manner. In order to ensure security with the CTR, the encryption key K should be changed for every 2n/2 blocks of encryption where n is the number of bits in a block [3]. In summation on the modes of operation in, the CTR mode is marked as the best choice among all the others.

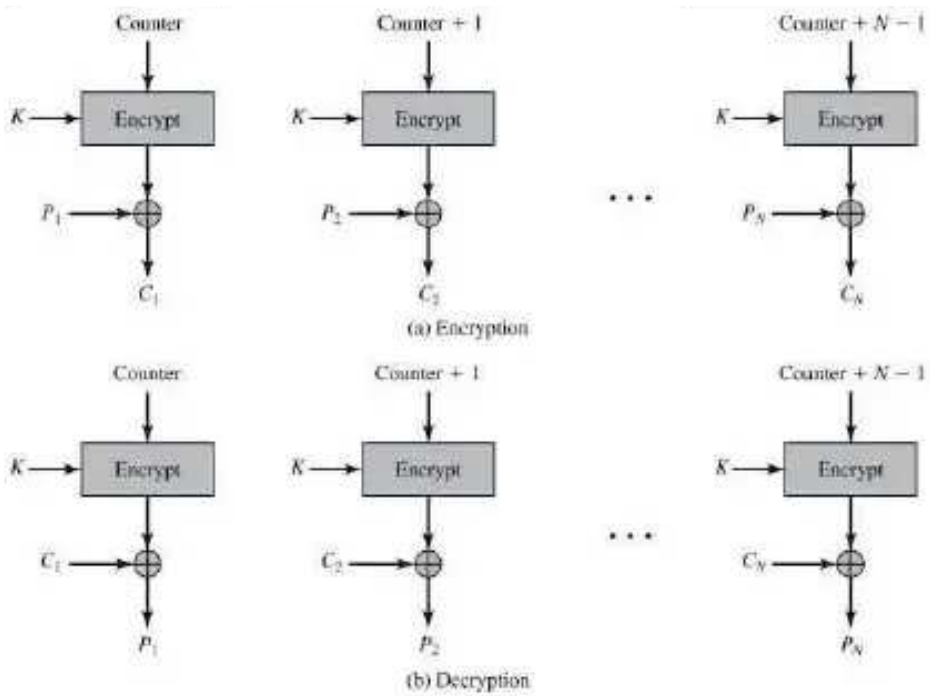


Figure 3.10: Scheme of CTR

3.4. Database Schema

3.4.1.3 New Fields(s)

Table Name	Field Name	Data Type	Allow Nulls	Field Description

Plain text	Plaintext	128 bit number	yes	128 bit number that will be given to encryption block
Key	Private key	128 bit number	yes	128 bit private number that will be given to cipher decryption block/

Table 3.1: Table of Fields

3.4.1.4 Fields Change(s)

Table Name	Field Name	What they do?
Sbox value	Sub_bytes	The changing input sequence after comparing it with the sbox 2 D matrix
Shift Rows	ShiftRows	Shift 1 byte in 2nd row, 2 bytes in 3rd row and 3 bytes in 4th row
Mix Colums	encMixCols	Each column obtained from Shift Rows is multiplied with Rijndael's Galois Field
Round Key	AddRoundKey	Obtained state from Mix Columns with the round key using a special type of arithmetic.

Table.3.2: Table of Variables

3.4.2 Data Migration

Existing data is migrating in new tables by analyzing of sub bytes and shift rows.

3.5. High Level Design

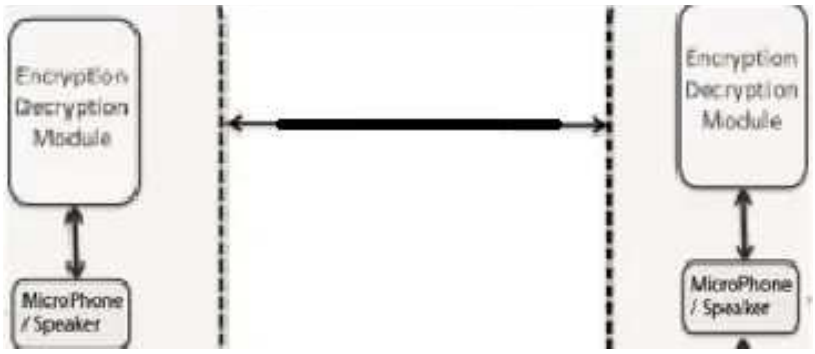


Figure 3.6: High Level Design

3.6. Low Level Design

Encryption

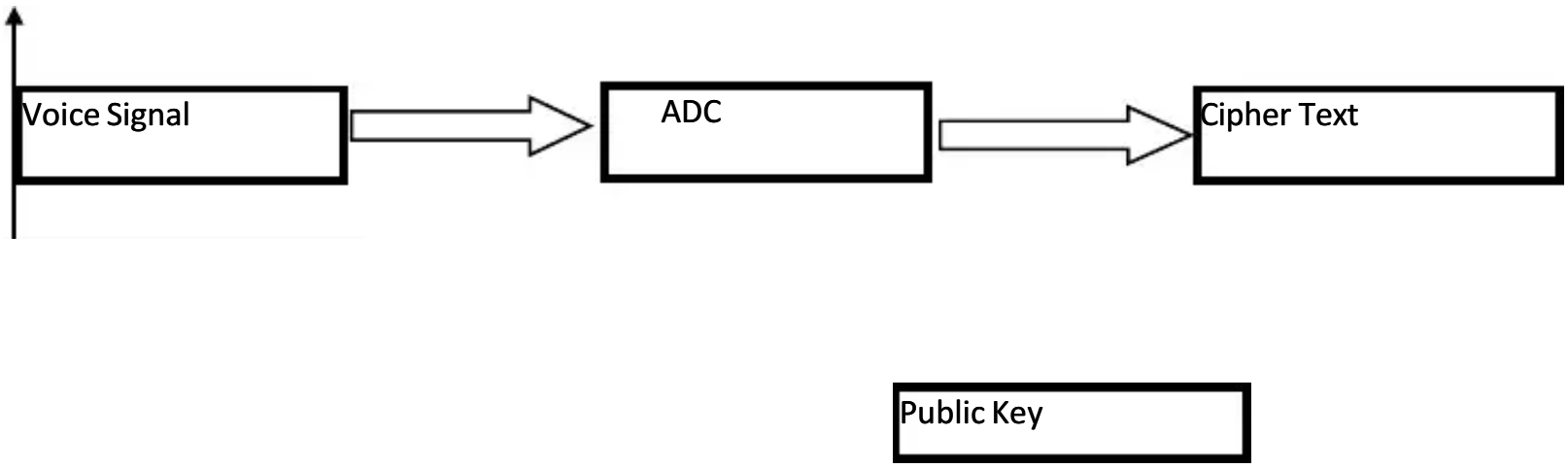


Figure 3.7: Low Level Design

At encryption block, a speech /voice signal is given as input to an Analog to Digital convertor (ADC) which will converted into binary and then voice samples will be stored it on FPGA Block RAMS then encrypted with public key. Then the cipher text it is sent through a medium.

3.6.1. Module 1…N

Module 1

SBOX

Main purpose of SBOX is 128 bit numbers is applied to each block by separated to 1 byte. The range of S-BOX is from 00-FF. Each byte will be computed by 2D table.

3.7. User Interface Design

At the GUI the process is generating the User Public key.

3.7.1. Screenshots 1… N

Major user interface screens the user will experience will be a C/C++ console using Microsoft Visual Studio.

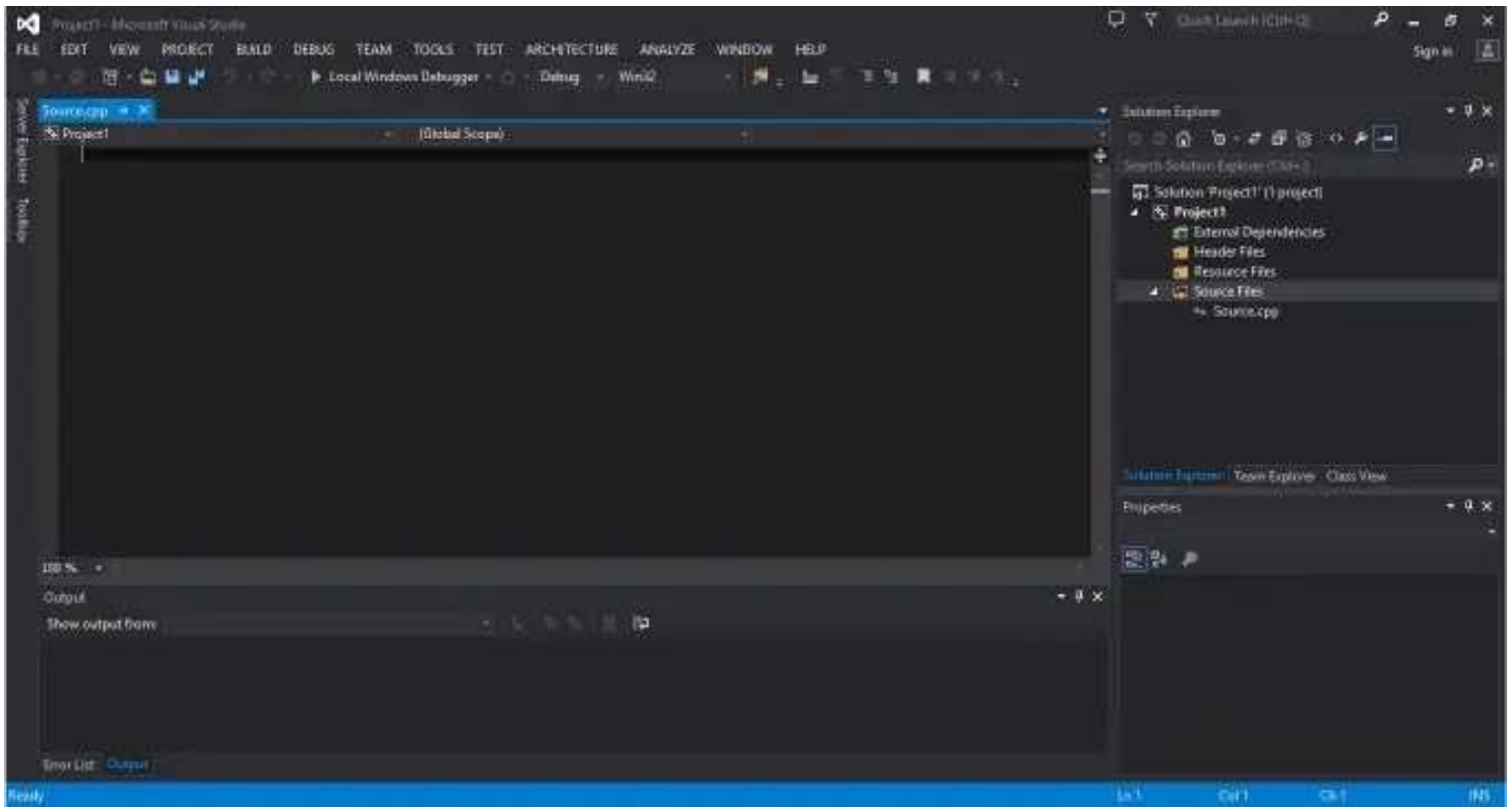


Figure 3.8: Visual Studio Interface

3.8. Summary

This structure accomplishes tradeoff between the speed and zone even without utilizing the BRAM. The information way is kept up to be 128 bits. The entire structure is performed with the assistance of Xilinx ISE and VIVADO HLS orchestrated with its apparatuses. The recreation and combination is finished focusing on the Spartan 6 FPGA. From the got exhibitions, we can reason that our proposed AES Architecture is appropriate to be utilized in asset obliged frameworks.

Chapter 4 Implementation

4.1. Discussion

Most of the challenging situations we faced during the implementation required finding the proper stability between flexibility, effectiveness and usability, while not compromising safety or security.

4.2. Development Methodologies

Existing Voice AES encryption techniques are implemented in C/C++ and Verilog. We can use these existing techniques for our research and to convert the existing techniques C/C++ code in Verilog then verification and comparison to existing Verilog and C/C++ techniques by speed testing, complexity and security.

Modules in C++ Code:

aes_ctr128_axis:
The module for the complete AES-128 CTR mode encryptor / decryptor, including key expansion. In this module has a 128-bit input data, 128-bit key and 256-bit of initialization vector. This module will XOR ECB (Electronic Code Book) output with plaintext for encryption and with cipher text for decryption.

aes_ecb128:

It is a complete AES-128 ECB mode encryptor, including key expansion. There are 10 rounds required for AES-128. It performs process of sub Bytes, shift Rows, mix Columns and add round key.

encMixCol:
Every input and output of different stages in this module are of 8 bits. In round 1 the four numbers of one column are modulo multiplied in Rijndael's Galois filed by a given matrix. The MixColumns step along with the ShiftRows steps is the primary source of diffusion in Rijndael.

keyExpEngine:
Key expansion range in this module is 32 bits. This module calculates the next round key for AES key expansion.

nonceCount:

Nonce and counter for AES counter mode. Initialization Vector will be loaded at power-on/reset or at first 16byte word (message block) of new plaintext/cipher text message. The last word of current plaintext/cipher text message is indicated by "last".
At the first word of a new message, the counter and nonce will load from init_vect. This allows the counter to be started at a non-zero value chosen by the user. The sizes (in bits) of the counter and nonce variables can be modified but the sum of their lengths must be 128.

rCon:

RCON lookup table for AES key expansion. Expansion of the given Cipher key into 11 partial keys, used in initial round, the 9 main rounds and the final round. The expanded key can be seen as an array of 32-bit columns numbers from 0 to 43. The first four columns are filled with the given Cipher key.

rotWord:

Word rotate function for AES key expansion.

shiftRows:

Input and output range in this module is 8 bits. It rotates over 1st byte, 2nd byte and 3rd byte.

subBytes:

In the Sub Bytes step, each byte in the matrix is updated using an 8-bit substitution box, the Rijndael S-box. This module provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF (28), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points, and also any opposite fixed points.

subWord:

SBOX function for key expansion.

4.3. Implementation Tools and Technologies

Microsoft Visual Studio was used to analyze the existing techniques; The C implementation will be synthesized into its equivalent Verilog implementation using higher level synthesis tool. VIVADO HLS can convert into a higher language code into its parameterized Verilog modules. A Xilinx ISE test tool was used for testing, verification and comparison of new generated technique with existing techniques.

Figure 4.1: Vivado HLS Interface

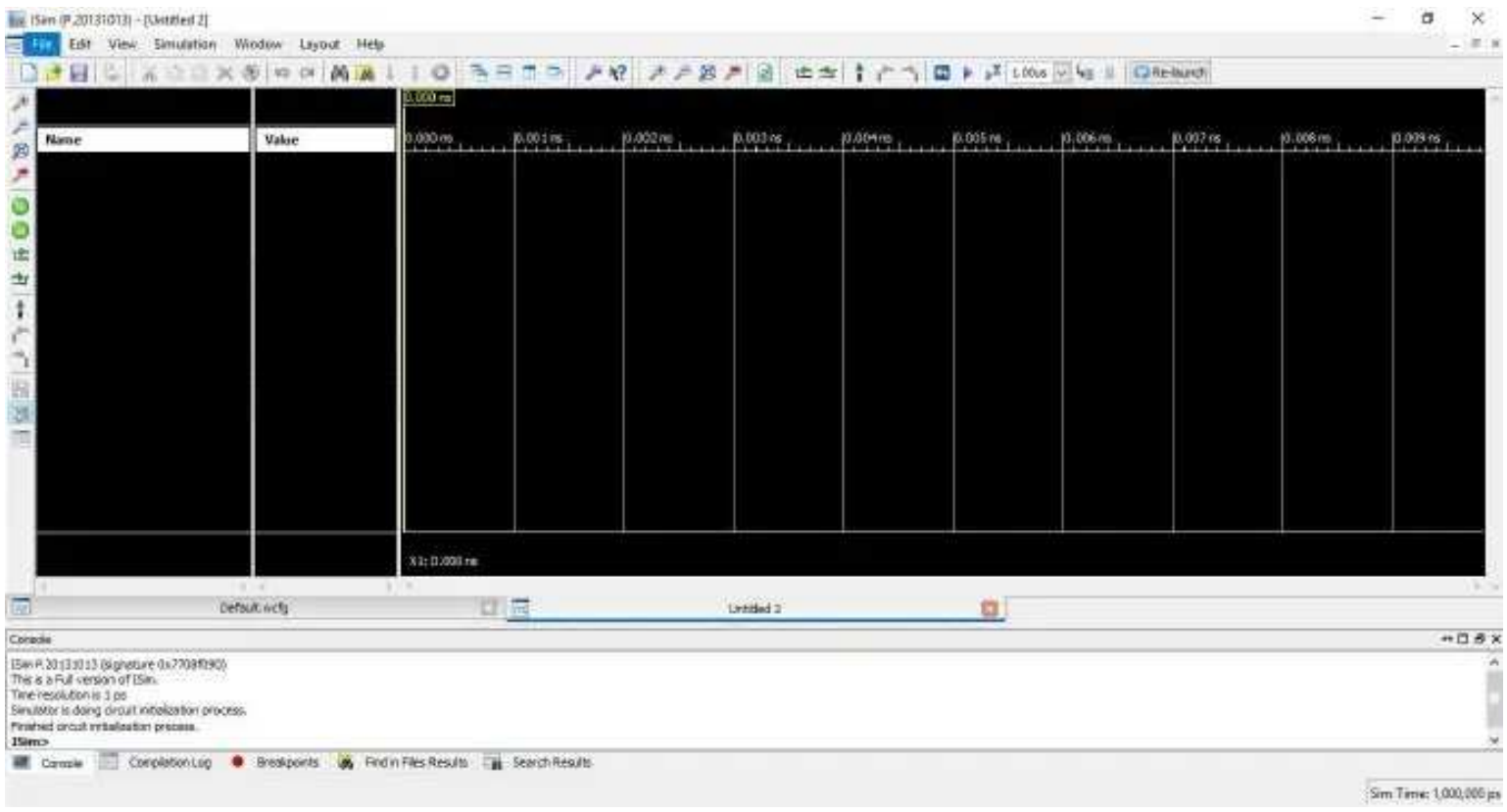


Figure 4.2: ISIM Interface

4.4. Summary

In this project we have implemented AES technique in C/C++ and C implementation will be synthesized into its equivalent Verilog implementation using higher level synthesis tool. We use existing techniques forverification and comparison to implemented Verilog and C/C++ techniques by speed testing, complexity and security.

Chapter 5 Testing

5.1. Testing Techniques Employed for This Project

Major testing technique employed for this project is ISIM. Xilinx ISIM is a Hardware Description Language (HDL) simulator that permits to perform purposeful and timing simulations for VHDL, Verilog and mixed VHDL/Verilog designs.

5.2. Test Cases

There must be a defined methodology to ensure an implementation meets the published standard or not. In simple words, one should be able to validate that given a plaintext string and a specific key, the output of the encryption algorithm is the correct one.

Such a methodology does exist. So, in this project testing cases includes Input values and key management. The data we gave is **"d4e0b81e27bfb44111985d52aef1e530"** in Hexadecimal and the program returned correct value.

5.3. Test Results

The test results that are obtained by testing elements (Input values and key) using Xilinx ISIM testing techniques is correct input/output values. Test results from test cases are given below:

In the **Mix Columns** step, the four bytes of every section of the state are joined utilizing an invertible direct change. The Mix Columns work accepts four bytes as information and four bytes for output, where each info byte influences every one of the four output bytes. Together with Shift Rows, Mix Columns gives assemble in the cipher.

During this activity, every section is multiplied by the known matrix that is for the 128-bit key is

The multiplication activity is characterized as: multiplication by 1 implies no change, multiplication by 2 implies moving to the left side, and multiplication by 3 implies moving to the left side and afterward performing XOR with the underlying un-shifted values. After moving, a conditional XOR with 0x1B will be performed if the shifted value is bigger than 0xFF.

Every section is treated as a polynomial over Galois Field (2^8) and is then multiplied modulo x4+1 with a fixed polynomial c(x) = 0x03 · x3 + x2 + x + 0x02. The coefficients are shown in their hexadecimal equivalent of the binary representation of bit polynomials from Galois Field (2)[x].

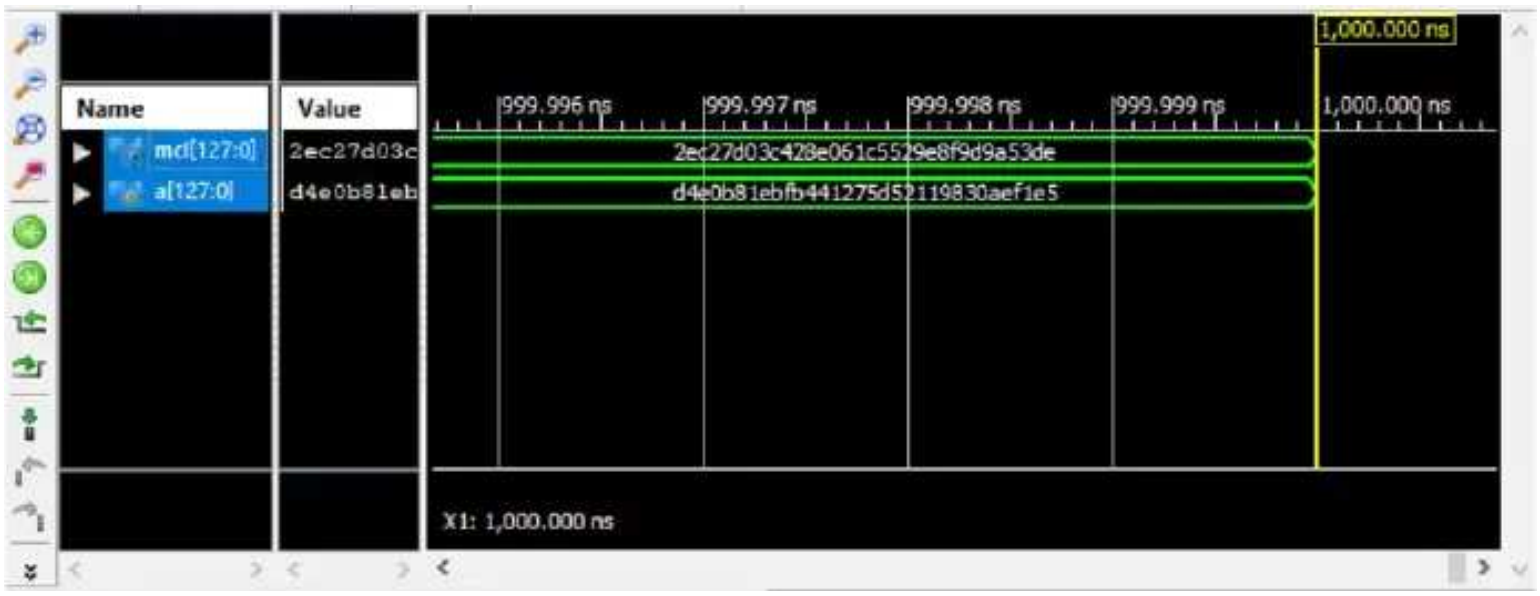


Figure 5.1: Result of Mix Columns Block Wave Diagram

The **S-Box** is produced by deciding the inverse multiplication of a given number in Rijndael's Galois Field. The inverse multiplication is then changed using the following matrix know as affine transformation matrix:

$$\begin{bmatrix} 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \\ 11111000 \\ 01111100 \\ 00111110 \\ 00011111 \end{bmatrix}$$

This relative change can likewise be determined by the accompanying calculation:

1. Store the inverse multiplication of the info number in two 8-bit unsigned impermanent factors: **s** and **x**
2. Rotate the one-bit **s** value to the left side; if the estimation of **s** had a high piece (eight bit from the left) of one, make the lower bit of **s** one; generally, the low bit of **s** is zero.
3. XOR the estimation of x with the estimation of s, storing estimated value in x

4. For three additional cycles, repeated stages two and three; stages two and three are completed a sum of multiple times.

5. The estimation of "x" will currently have the changed value.

After the relative change is done, XOR the received value with the decimal number 99 (the hexadecimal number).

Here is the code that performs above algorithms.

This code will generate the AES S-box, which is represented below with hexadecimal form.

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

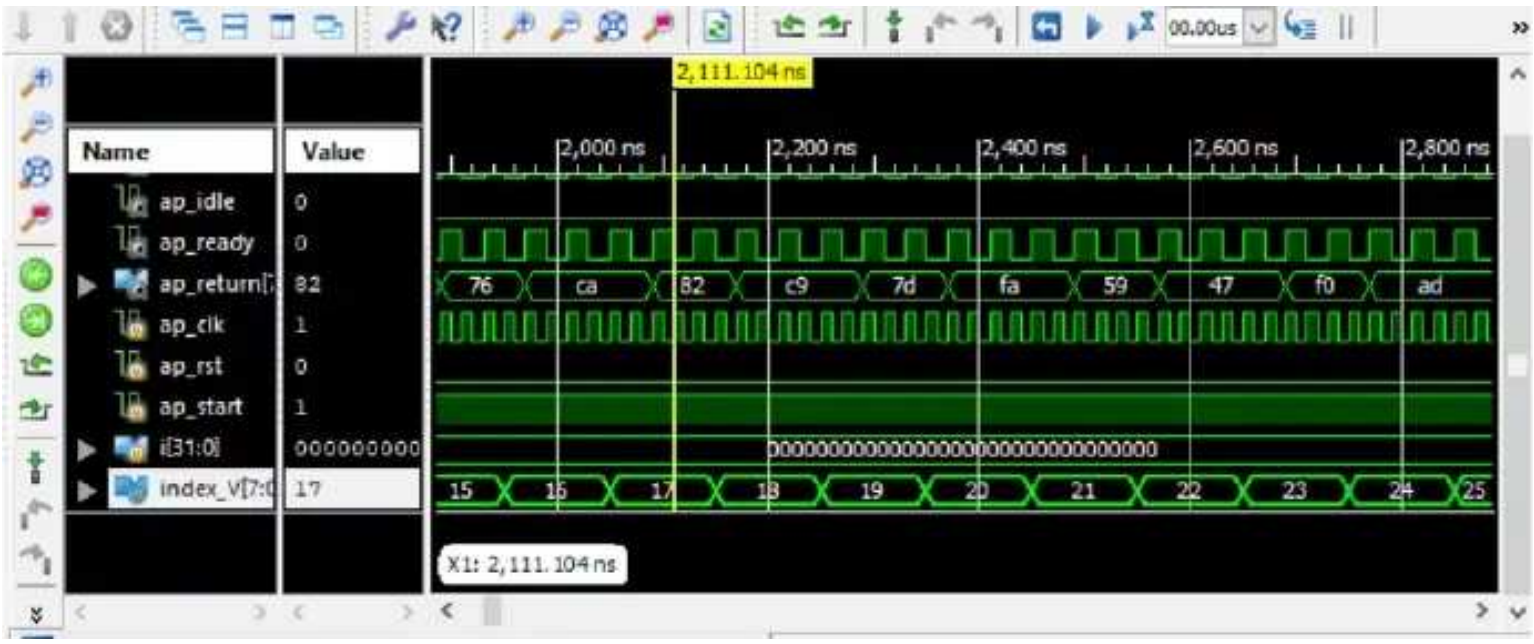
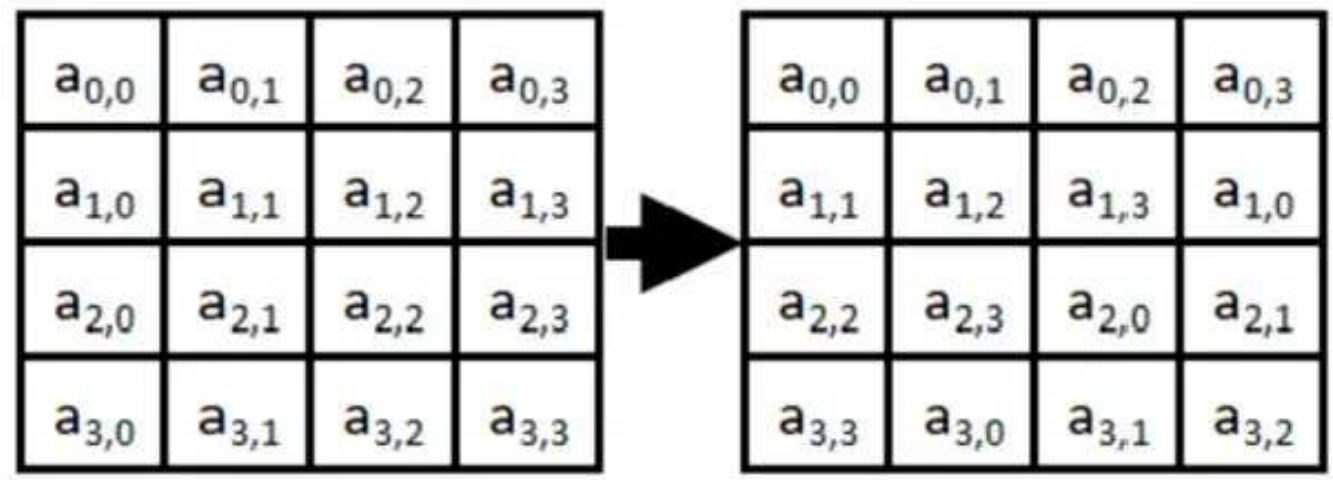


Figure 5.2: Result of S-Box Block Wave Diagram

In the **Shift Rows** period of AES, each column of the 128-bit interior state of the cipher is moved. The column in this stage refers to the standard representation of the inward state

in AES, which is a 4x4 matrix where every cell contains a byte. Bytes of the inner state are set in the matrix row wise over lines from left to right and down columns.

In the Shift Rows activity, every one of these lines is moved to left side by a set sum: their column number beginning with zero. The top column isn't moved in any way, the following line is moved by one, etc. This is outlined in the Figure underneath.



In the Figure, the main number in every cell alludes to the line number and the second alludes to the segment. The highest line (row 0) does not move by any means, push 1 moves left by one, and up to so on.

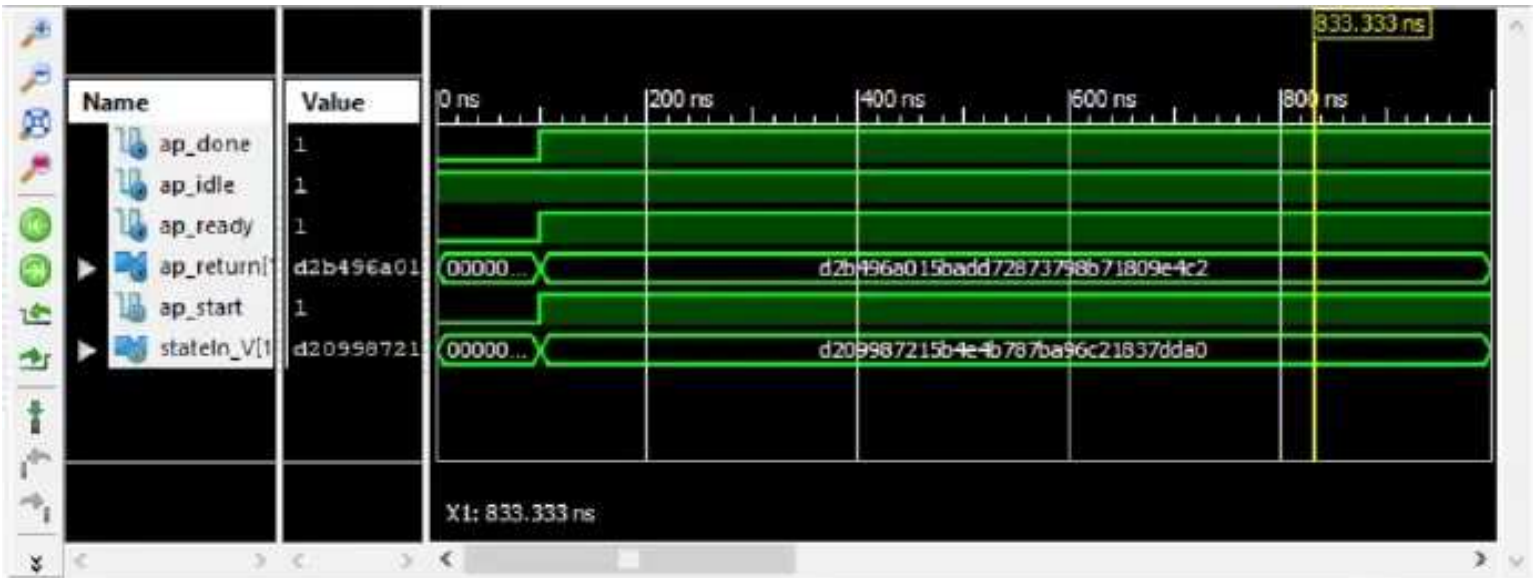


Figure 5.3: Result of Shift Rows Block Wave Diagram

In **Sub Bytes** period of AES includes the contribution to bytes and going each through a Substitution Box or S-Box. In contrast to DES, AES utilizes a similar S-Box for all bytes. The AES S-Box actualizes reverse augmentation in Galois Field 2^8. The AES S-Box is appeared in the Table underneath.

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

To read above Table, the byte input is divided into two 4-bit parts. The first half decides the row and the second half decides the column. For instance, the S-Box change of 35 or 0x23 can be found in the cell at the crossing point of the row named 20 and the column named 03. In this manner decimal 35 ends up 0x26.

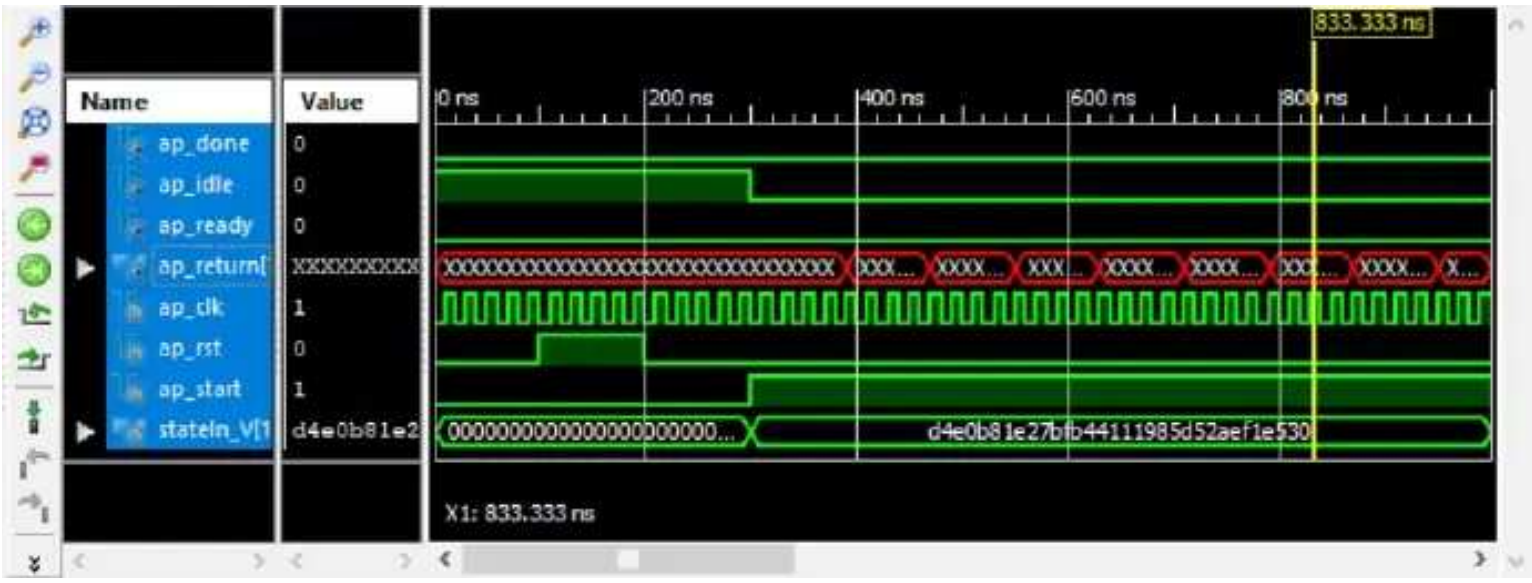


Figure 5.4: Result of Sub Bytes Block Wave Diagram

5.4. Summary

The summary of this chapter tells us about test results in ISIM by testing input of 128 bits' plain text, key and their wave diagrams.

Chapter 6 Data Analysis and Results

6.1. The Empirical Study Methodology

We implemented AES algorithm in C language, and generating C code into Verilog using VIVADO HLS and testing and comparing it by existing implemented codes.

6.1.1. Metrics for Result Comparison

We take 128-bit in both generated and existing code. By evaluating resulting we analyze and compare their values are equal or not.

Proposed Verilog implementation

LUT			FF			IOB's		
Used	Avail	%	Used	Avail	%	Used	Avail	%
2001	53200	3%	1604	3183	50%	520	200	260%

Table 6.1: Table of C-Based Code

Reference AES Design module

LUT			FF			IOB's		
Used	Avail	%	Used	Avail	%	Used	Avail	%
9387	53200	17%	0	9387	0%	384	200	192%

Table 6.2: Table of Verilog-Based Code

6.1.2. Performance Evaluation

By comparing and testing the values of our generated and existing code we can say that our hardware is efficient co it uses on 3% of hardware. Existing codes block circuits are combinational and our generated code blocks circuits are sequential because they're using Flip Flop.

6.2. Statistical Evaluation

By analyzing and comparing lookup tables of existing and generated techniques we can find that existing technique uses 17% of lookup table and generated technique uses 3% of lookup tables.

6.3. Summary

The summary of the chapter tells us about the testing techniques that are being used during the completion of project. From the scratch till the end of the project from the existing works to the comparison with the results generated from newly work-done. Test cases basically tells us about the Lookup table, flip-flops and IOB's that are being used in the project their previous results and new results are being compared in order to differentiate.

Chapter 7 Conclusions and Future Work

7.1. Contributions

7.1.1. Contribution 1

Our first contribution in this project was to find the issues of conversion of floating point to fix point implementation. Usually the code in C is processors based which is usually used by the floating-point.

On the other hand, FPGA's have fixed point implementation. If they are converted into floating point, then they consume a lot of resources. This phenomenon was observed in the project. We used ZYNQ-702 architecture as it supports floating point numbers.

7.1.2. Contribution 2

Analysis of clock without predefined pipelines had an effect in the latency. The performance of the circuit was checked with VIVADO HLS directives which were responsible for the creation of pipelines.

7.2. Findings

Overall project was a good learning curve-one of the key findings of the project was the VIVADO capability to optimize the hardware. VIVADO takes an array and process it in the form of shift registers. Furthermore, the synthesis tool employed usage for dedicated DSP units. DSP unit have multiple accumulators which VIVADO mapped them not so efficiently as we would have liked.

We observed in our finding was much better than existing techniques in manner of hardware utilization and time complexity.

7.3. Future Work

The AES algorithm is most widely used algorithm for different security based applications. Security of the AES algorithm can be expanded by utilizing biometric framework for creating key.

As a future work, we are going to continue this research in order generating more secure key by biometric implementation and to get the maximum encryption speed in limited implementation area. Moreover, some artificial intelligence techniques can be used for simulating the AES encryption and crypto analysis processes.

7.3.1. Improvements in the existing System

Existing system used 128-bit scheme. It can be further improved by increases 128 bits to 192 and 256-bit scheme. System performance can be further increased by applying pipe lines stages in between modules.

7.3.2. Further System Designs

We need to have an improved mechanism for floating point to fixed point implementation. We haven't implemented any parallel processing. For parallel processing we can use the concept of loop unrolling. Loop unrolling identifies independent sections in for loop. By using this concept, we can further increase the performance of the system.

References

[1] Abdullah, A. M., & Aziz, R. H. H. (2016, June). New Approaches to Encrypt and Decrypt Data in Image using Cryptography and Steganography Algorithm., International Journal of Computer Applications, Vol. 143, No.4 (pp. 11-17).

[2] Singh, G. (2013). A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. International Journal of Computer Applications, 67(19).

[3] Gaj, K., & Chodowiec, P. (2001, April). Fast implementation and fair comparison of the final candidates for Advanced Encryption Standard using Field Programmable Gate Arrays. In Cryptographers' Track at the RSA Conference (pp. 84-99). Springer Berlin Heidelberg.

[4] Stallings, W. (2006). Cryptography and network security: principles and practices. Pearson Education India.

[5] Yenuguvanilanka, J., & Elkeelany, O. (2008, April). Performance evaluation of hardware models of Advanced Encryption Standard (AES) algorithm. In Southeastcon, 2008. IEEE (pp. 222-225).

[6] Lu, C. C., & Tseng, S. Y. (2002). Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter. In Application-Specific Systems, Architectures and Processors, 2002. Proceedings. The IEEE International Conference on (pp. 277-285).

[7] Mohamed, A. A., & Madian, A. H. (2010, December). A Modified Rijndael Algorithm and it's Implementation using FPGA. In Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on (pp. 335-338).

[8] Pramstaller, N., Gurkaynak, F. K., Haene, S., Kaeslin, H., Felber, N., & Fichtner, W. (2004, September). Towards an AES crypto-chip resistant to differential power analysis. In Solid-State Circuits Conference, 2004. ESSCIRC 2004. Proceeding of the 30th European IEEE (pp. 307-310).

[9] Deshpande, H. S., Karande, K. J., & Mulani, A. O. (2014, April). Efficient implementation of AES algorithm on FPGA. In Communications and Signal Processing (ICCSP), 2014 IEEE International Conference on (pp. 1895-1899).

[10] Nadeem, H (2006). A performance comparison of data encryption algorithms," IEEE Information and Communication Technologies, (pp. 84-89).

[11] Diaa, S., E, Hatem M. A. K., & Mohiy M. H. (2010, May) Evaluating the Performance of Symmetric Encryption Algorithms. International Journal of Network Security, Vol.10, No.3, (pp.213-219).

[12] Jain, R., Jejurkar, R., Chopade, S., Vaidya, S., & Sanap, M. (2014). AES Algorithm Using 512 Bit Key Implementation for Secure Communication. International journal of innovative Research in Computer and Communication Engineering, 2(3).

[13] Selmane, N., Guilley, S., & Danger, J. L. (2008, May). Practical setup time violation attacks on AES. In Dependable Computing Conference, 2008. EDCC 2008. Seventh European (pp. 91-96). IEEE.

[14] Berent, A. (2013). Advanced Encryption Standard by Example. Document available at URL http://www. networkdls. com/Articles/AESbyExample. pdf (April 1 2007) Accessed: June.

[15] Benvenuto, C. J. (2012). Galois field in cryptography. University of Washington.

[16] Lee, H., Lee, K., & Shin, Y. (2009). Aes implementation and performance evaluation on 8-bit microcontrollers. arXiv preprint arXiv:0911.0482.

[17] Padate, R., & Patel, A. (2014). Encryption and decryption of text using AES algorithm. International Journal of Emerging Technology and Advanced Engineering, 4(5), 54-9.

[18] Reddy, M. S., & Babu, Y. A. (2013). Evaluation of Microblaze and Implementation of AES Algorithm using Spartan-3E. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2(7), 3341-3347.

8	P. Hodgers, K. H. Boey, M. O'Neill. "Variable window power spectral density attack", 2011 IEEE International Workshop on Information Forensics and Security, 2011 Publication	<1%
9	web.cs.dal.ca Internet Source	<1%
10	Submitted to Study Group Australia Student Paper	<1%
11	www.ijircce.com Internet Source	<1%
12	Shashank P. Wankhade, A.N. Bandal. "Security for Automation in Internet of Things Using One Time Password", 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), 2017 Publication	<1%
13	Jayanta Kumar Pal, J. K. Mandal, Somsubhra Gupta. "Composite Transposition Substitution Chaining Based Cipher Technique", 2008 16th International Conference on Advanced Computing and Communications, 2008 Publication	<1%
14	etheses.bham.ac.uk Internet Source	<1%

15	vssut.ac.in Internet Source	<1%
16	media.proquest.com Internet Source	<1%
17	"Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014", Springer Nature, 2015 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		