IMPLEMENTATION OF ADVANCED ENCRYPTION STANDARD TARGETED AT FPGA ARCHITECTURES

A PROJECT REPORT

Submitted by

NITHIN SHYAM S

PRADEEP KUMAR T

RAHUL S

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH,

COIMBATORE - 641 062

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2021

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**IMPLEMENTATION OF ADVANCED ENCRYPTION STANDARD TARGETED AT FPGA ARCHITECTURES"** is the bonafide work of **NITHIN SHYAM S (715517106026), PRADEEP KUMAR T (715516106028) and RAHUL S (715516106034)** who carried out the project work under my supervision.

SIGNATURE
Dr. P. Vetrivelan M.S., Ph.D.,
HEAD OF THE DEPARTMENT
Associate Professor
ECE Department
PSG Institute of Technology
And Applied Research,
Coimbatore - 641 062.

SIGNATURE
Mr. K. Paldurai M.E.,
SUPERVISOR
Assistant Professor
ECE Department
PSG Institute of Technology
And Applied Research,
Coimbatore - 641 062.

Submitted for the Anna University Viva-Voce examination held on 06/08/2021.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We would like to thank the **Management** of PSG Institute of Technology and Applied Research for providing us with excellent facilities for the completion of this project.

We are grateful to our Secretary, **Dr. P. V. Mohanram**, of PSG Institute of Technology and Applied Research for giving us support throughout the project.

We would like to thank **Dr. G. Chandramohan**, Principal of PSG Institute of Technology and Applied Research for encouraging us throughout the project.

We are thankful to **Dr. P. Vetrivelan**, Associate Professor and Head, Department of Electronics and Communication Engineering, for his constant support throughout the project.

It is our pleasure to thank our Project Guide **Mr. K. Paldurai**, Assistant Professor, Department of Electronics and Communication Engineering, for helping us complete the project successfully.

We also would like to thank all the faculty members and staff of the Electronics and Communication Engineering Department for their kind cooperation and encouragement during the course of this work.

ABSTRACT

Data has become an invaluable commodity in this digital era. All organizations depend on data to track their growth and to serve their end goal better. The data has to be kept confidentially as data is as important as money in many cases. Hackers continue to pose a threat by stealing data. Cybercrime is predicted to inflict damages totaling $6 trillion USD globally in 2021 and is estimated to cost the world $10.5 trillion by 2025. Cryptography played a very important role in the World Wars and the cracking of the Germans' crypto system by the British led to their ultimate downfall.

From the 1970s, scientists have incorporated cryptography into computers and all digital communications equipment. The Data Encryption Standard was standardized in 1977 and served well until the 1990s when it was finally cracked and deemed insecure. So, the National Institute of Standards and Technology made a call for a better algorithm and in 2001, the Advanced Encryption Standard was introduced. AES comes in 3 different key sizes 128, 192 and 256-bit lengths. The 128-bit version was implemented is several designs focusing on specific purposes and predominantly used for nearly 20 years. With the current trend of technological advancement, it can be broken in a few years. NIST claims that AES-128 is fine at the very least up to 2030. The AES can be programmed in software or built with pure hardware. However, Field Programmable Gate Arrays (FPGAs) offer a quicker and more customizable solution. Our project compares the 128-bit and 256-bit AES algorithms implemented on FPGA architectures. The implemented design handles both encryption and decryption and was tested on the Xilinx xc7z020-clg484-1 FPGA using the Xilinx Vivado 2019.1 software in Verilog language.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF TABLES	vii
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	x
1	**INTRODUCTION**	1
	1.1 NEED FOR THE PROJECT	1
	1.2 CRYPTOGRAPHY	2
	1.3 OBJECTIVES OF CRYPTOGRAPHY	3
	1.4 CLASSIFICATION OF CRYPTOGRAPHY	3
	1.5 ORGANIZATION OF CHAPTERS	4
2	**LITERATURE REVIEW**	5
3	**ENCRYPTION**	7
	3.1 SYMMETRIC KEY OR SECRET KEY SYSTEM	7
	3.1.1 Stream Cipher	8
	3.1.2 Block Cipher	10
	3.2 ASYMMETRIC KEY OR PUBLIC KEY SYSTEM	11
4	**ADVANCED ENCRYPTION STANDARD**	12
	4.1 MATHEMATICS INVOLVED	13
	4.1.1 Galois Field	13
	4.1.2 2^8 Galois Field Transformation	14
	4.1.3 Finite Field Addition and Subtraction	14
	4.1.4 Finite Field Multiplication and Multiplicative Inverse	16
	4.2 THE STATE	18

4.2.1 The State as an Array of Columns 19
4.3 STEPS INVOLVED IN AES ALGORITHM 20
4.3.1 SubBytes (Substitute Bytes) 22
4.3.2 Shift Rows 24
4.3.3 Mix Columns 26
4.3.4 Add Round Key 28
4.3.5 Key Expansion 29
4.4 SPECIFICATIONS OF AES 31
4.5 CRYPTANALYSIS RESULTS OF AES 31
4.6 AES ENCRYPTION (ROUND BY ROUND) 32
4.7 AES DECRYPTION (ROUND BY ROUND) 35
5 IMPLEMENTATION 38
5.1 AES-128 RESULTS 42
5.2 AES-256 RESULTS 44
6 CONCLUSION & FUTURE SCOPE 47
6.1 CONCLUSION 47
6.2 FUTURE SCOPE 47

REFERENCES 49

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
4.1	AES Relation Between Key Length and Number of Rounds	31
5.1	xc7z020-clg484-1 FPGA Specifications	38

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
1.1	Cryptography Classification	4
3.1	Symmetric Encryption	7
3.2	Stream Cipher	8
3.3	Block Cipher	10
3.4	Asymmetric Encryption	11
4.1	State Matrix	19
4.2	AES General Structure	20
4.3	S-Box	22
4.4	SubBytes Transformation	23
4.5	Inverse S-Box	24
4.6	Shift Rows Transformation	25
4.7	Inverse Shift Rows Transformation	26
4.8	Mix Columns Matrix	26
4.9	Inverse Mix Columns Matrix	27
4.10	Mix Columns Transformation	27
4.11	Add Round Key Transformation	28
4.12	Block Diagram of Key Expansion	29
4.13	Functional Representation of Key Expansion	30
4.14	RCon/RC for 10 rounds	30
4.15	AES Encryption Block Diagram	32
4.16	AES-128 Encryption Example	33, 34
4.17	AES Decryption as an inverse of Encryption	35
4.18	AES-128 Decryption Example	36, 37
5.1	Zedboard	39
5.2	Zedboard Block Diagram	40
5.3	Testcases	41
5.4	AES-128 Waveform	42

5.5	AES-128 implemented in Hardware	42
5.6	Performance of AES-128 Design	43
5.7	Resource Utilization of AES-128 Design in Hardware	44
5.8	AES-256 Waveform	44
5.9	AES-256 implemented in Hardware	45
5.10	Performance of AES-256 Design	45
5.11	Resource Utilization of AES-256 Design in Hardware	46

LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
BRAM	Block Random Access Memory
BUFG	Global Buffer
DES	Data Encryption Standard
ECC	Elliptic Curve Cryptography
FF	Flip Flop
FIPS	Federal Information Processing Standards
FPGA	Field-Programmable Gate Array
GF	Galois Field
IEC	International Electrotechnical Commission
IO	Input/Output
ISO	International Organization for Standardization
LED	Light Emitting Diode
LUT	Look-up Table
MITM	Meet-In-The-Middle
NIST	National Institute of Standards and Technology
RAM	Random Access Memory
RCon/RC	Round Constant
RSA	Rivest-Shamir-Adleman
S-BOX	Substitution Box
SoC	System on Chip
VHDL	High-Speed Integrated Circuit Hardware Description Language
XOR	Exclusive OR

CHAPTER 1

INTRODUCTION

Data confidentiality has become a very essential part of any form of digital communication. Data is gold in today's age. 463 exabytes of data will be generated each day by humans as of 2025. Technologies such as Big Data, IoT, Machine learning and Artificial Intelligence produce huge amounts of data to be stored and processed. The processed result is very crucial for the advancement of any project or organization. The data has to be communicated so that it provides help to others in need. Most of the times, it needs to be sent through the internet which is an insecure channel. Take for instance, theft of medical records can be devastating. So, to make sure only the intended recipient gets access to the shared data, cryptography, to be more precise, encryption is used. It makes sure that the data which is sent through an insecure channel is scrambled and doesn't make sense to any interceptor. The data is unscrambled at the receiver's end.

1.1 NEED FOR THE PROJECT

The Advanced Encryption Standard, also referred to as Rijndael, is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001. It is a widely adopted encryption standard and has been poplar since its introduction. AES-128 is the most popular version in use today. This project implements the 256-bit key length version in FPGA and compares it with the 128-bit implementation and provides the differences in performance and resource utilization. This data is useful for estimation purposes when moving from 128-bit encryption to 256-bit encryption system.

1.2 CRYPTOGRAPHY

The word “cryptography” comes from the Greek work “kryptos” meaning hidden or secret. It is the study of secure communications techniques that allow only the sender and the intended recipient of a message to view its contents. It involves encryption, which is the act of scrambling ordinary text into what's known as ciphertext and decryption, where the ciphertext is unscrambled into ordinary text.

Ancient Egyptians were known to use these methods in complex hieroglyphics, and Roman Emperor Julius Caesar is credited with using one of the first modern ciphers. Prior to the early 20th century, cryptography was mainly concerned with linguistic and lexicographic patterns. Since then, the emphasis has shifted, and cryptography now makes extensive use of mathematics, including aspects of information theory, computational complexity, statistics, combinatorics, abstract algebra, number theory, and finite mathematics generally.

Cryptography is best known as a way of keeping the contents of a message secret. Confidentiality of network communications, for example, is of great importance for e-commerce and other network applications. However, the applications of cryptography go far beyond simple confidentiality. In particular, cryptography allows the network business and customer to verify the authenticity and integrity of their transactions used the primitive call digital signature. If the trend to a global electronic marketplace continues, better cryptographic techniques will have to be developed to protect business transactions. Sensitive information sent over an open network may be scrambled into a form that cannot be understood by a hacker or eavesdropper. This is done using a mathematical formula, known as an encryption algorithm, which transforms the bits of the message into an unintelligible form. The intended recipient has a decryption algorithm for extracting the original message. There are many examples of information on open networks, which need to be

protected in this way, for instance, bank account details, credit card transactions, or confidential health or tax records.

1.3 OBJECTIVES OF CRYPTOGRAPHY

- Confidentiality of the message: only the authorized recipient should be able to extract the content of the cypher. In addition, obtaining information about the content of the message (such as a statistical distribution of certain characters) should not be possible, once the cryptographic analysis becomes easier.
- Message integrity: the recipient must be able to determine if the message was altered during transmission.
- Authentication of the sender: the recipient should be able to identify the sender and verify if it was him who sent the message.
- Irrevocability of the sender: it should not be possible to deny the authorship of the message.

1.4 CLASSIFICATION OF CRYPTOGRAPHY

- Symmetric Key Cryptography aka Secret Key Cryptography - It is an encryption system where the sender and receiver of message use a single common key to encrypt and decrypt messages. Symmetric Key Systems are faster and simpler but the problem is that sender and receiver have to somehow exchange key in a secure manner.
- Hash Functions - There is no usage of any key in this algorithm. A hash value with fixed length is calculated as per the plain text which makes it impossible

for contents of plain text to be recovered. Many operating systems use hash functions to encrypt passwords.

- Asymmetric Key Cryptography aka Public Key Cryptography - Under this system a pair of keys is used to encrypt and decrypt information. A public key is used for encryption and a private key is used for decryption. Public key and Private Key are different. Even if the public key is known by everyone the intended receiver can only decode it because he alone knows the private key.

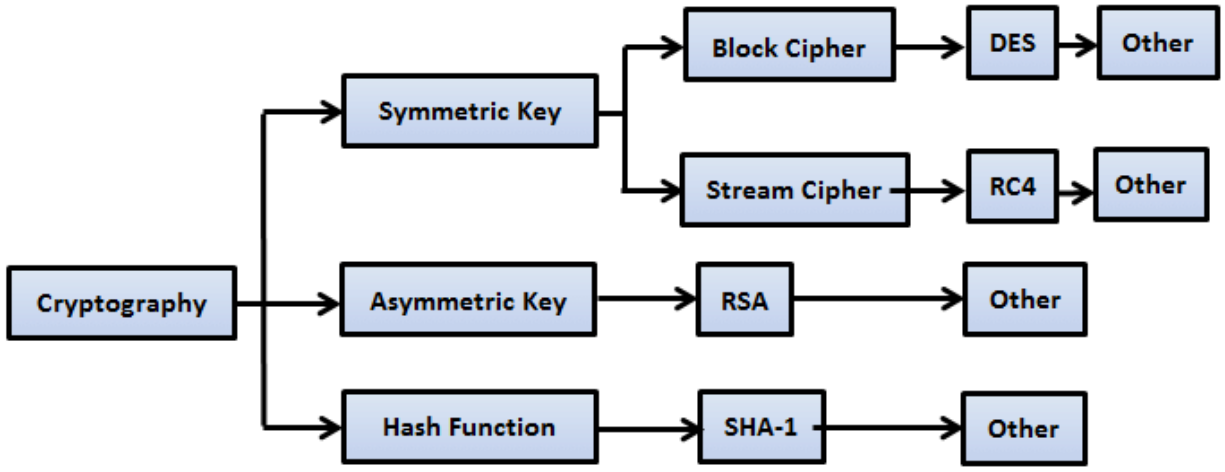


Fig 1.1 Cryptography Classification

1.5 ORGANIZATION OF CHAPTERS

Chapter 2 details the knowledge gathered from literature survey. Encryption and its types are described in Chapter 3. Chapter 4 goes into detail about the AES algorithm itself, about its history and its structure with an example. The features of implantation and the results obtained are shown in Chapter 5. The project's conclusion and future scope are present in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

The NIST published the specifications of AES in the FIPS publication 197 on November 26, 2001. It specifies the structure for AES in detail. Implementations of the algorithm with different key sizes is also discussed. The procedures to perform the transformations, Substitute Bytes, Shift Rows, Mix Columns, Add Round Key and Key Expansion are provided along with pseudo codes. The standard S-box is given. The equivalents of the transformation used to decrypt are also provided with their pseudo codes.

The research done by Pawel Chodowiec and Kris Gaj in "Very Compact FPGA Implementation of the AES Algorithm" provides a beautiful insight into the implementation of AES in FPGA. It has been implemented in the Spartan II FPGA focusing on compaction. Encryption, decryption and key schedule are all implemented using small resources of only 222 Slices and 3 Block RAMs. The performance is 4 clock cycles per round with a throughput of 166 Mbps.

The work done by Shuang Chen, Wei Hu, Zhenhao Li in "High Performance Data Encryption with AES Implementation on FPGA" three solutions for high throughput or low latency. The maximum throughput achieved is 31.296 Gbps which has a latency of 4.09 ns. For optimization, loop unrolling has been implemented partially or fully in the designs.

In "Efficient Implementation of AES Algorithm in FPGA Device" by Swinder Kaur and Prof. Renu Vig, speed is increased by processing multiple rounds simultaneously but at the cost of increased area. Algorithmic optimization techniques have also been used which includes exclusion of shift row stage and on

the fly round key generation. Their design consumes 6279 Slices and 5 B-RAMs and outputs data with a throughput of 1.18 Gbps.

Encryption and Decryption have been designed and implemented for the AES-128 algorithm in “Efficient Implementation of AES Algorithm On FPGA”. The implementation uses VHDL language and RTL structures are developed and simulated in Xilinx ISE 14.1 Project Navigator.

“FPGA Implementation of AES Algorithm” by Mr. Atul M. Borkar , Dr. R. V. Kshirsagar and Mrs. M. V. Vyawahare takes a look at the simulation od AES-128 with speed optimization. They have achieved a throughput of 352 Mbps (51 clock cycles). VHDL was the programming language used.

CHAPTER 3

ENCRYPTION

Encryption is the process through which data is encoded so that it remains hidden from or inaccessible to unauthorized users. It helps protect private information, sensitive data, and can enhance the security of communication between client apps and servers. This process converts the original representation of the information, known as plaintext, into an alternative form known as ciphertext. Ideally, only authorized parties can decipher a ciphertext back to plaintext and access the original information.

There are two classes of algorithm in encryption, an asymmetric key and symmetric key.

3.1 SYMMETRIC KEY OR SECRET KEY SYSTEM

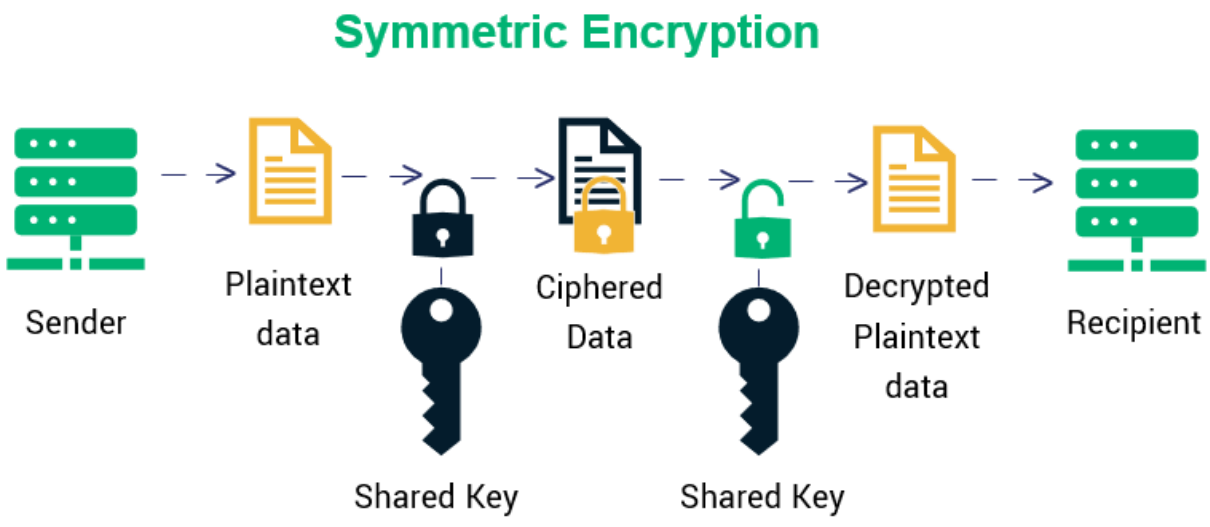


Fig. 3.1 Symmetric Encryption

In a symmetric or private key algorithm, in the ordinary case, the communication only uses only one key. A user Alice sends the secret private key K_c to another user Bob user before the start of the communication between them. Both sides use the same secret key to encrypt and decrypt the exchanged information. Data Encryption Standard (DES), Advanced Encryption Standard (AES) and Blowfish are examples of symmetric key algorithm.

Symmetric Key Cryptography is further classified into stream cipher and block cipher.

3.1.1 STREAM CIPHER

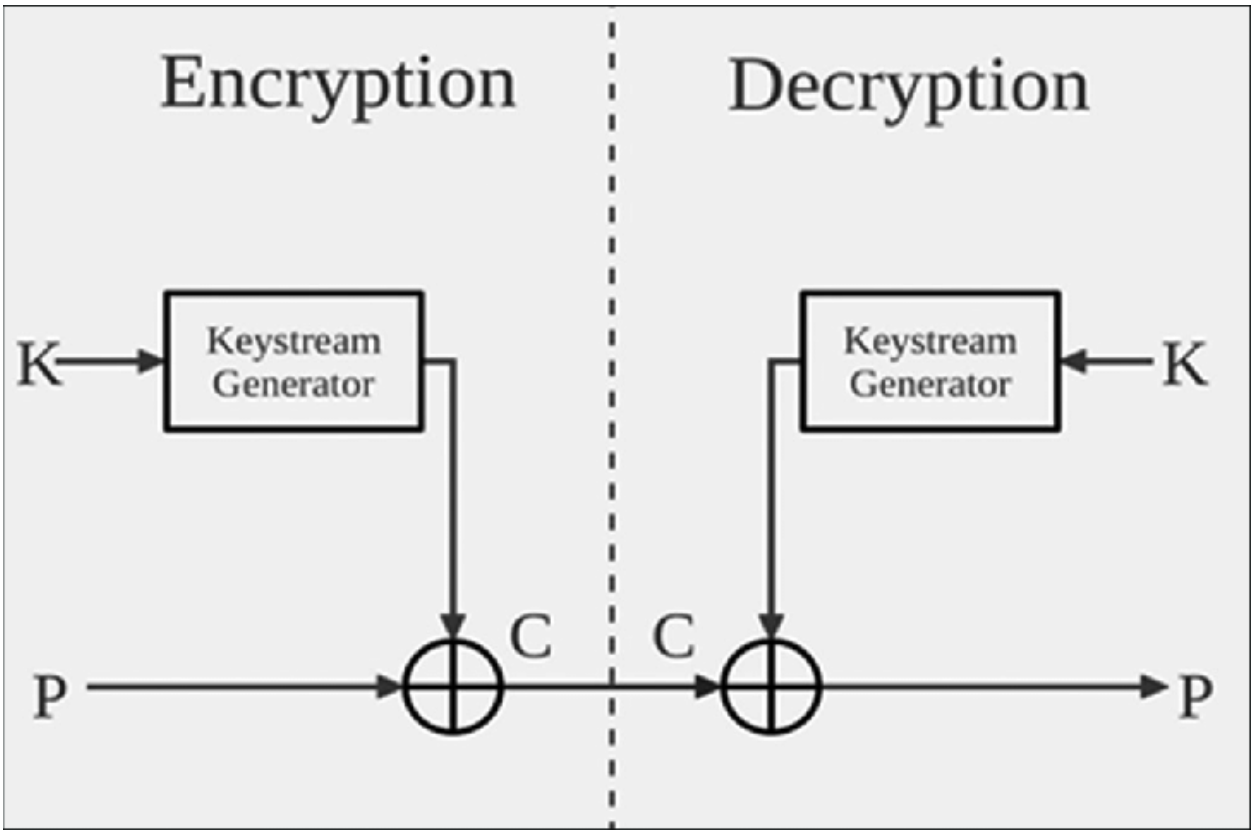


Fig 3.2 Stream Cipher

A stream cipher is a method of encrypting text in which a cryptographic key and algorithm are applied to each binary digit in a data stream, one bit at a time. They are designed to be extremely fast. They are faster than block ciphers. This method is not much used in modern cryptography. A keystream is used which is generated from a random seed by a pseudo-random generator. Encryption is accomplished by combining the key stream with the plaintext, usually with the bitwise XOR operation. Decryption is performed by applying the reverse transformation to the cipher text using the same secret key. The encryption of any particular plaintext with a block cipher will result in the same cipher text when the same key is used. With a stream cipher, the transformation of these smaller plaintext units will vary, depending on when they are encountered during the encryption process. The one-time pad is the most secure encryption possible digitally.

Characteristics of Stream Cipher:

- Symmetric encryption
- Usually implemented in hardware
- Encrypts by operating on a continuous data stream
- Stream cipher treats the message as a stream of bits and performs mathematical functions on them individually.
- Statistically unpredictable
- Much faster than any block cipher

3.1.2 BLOCK CIPHER

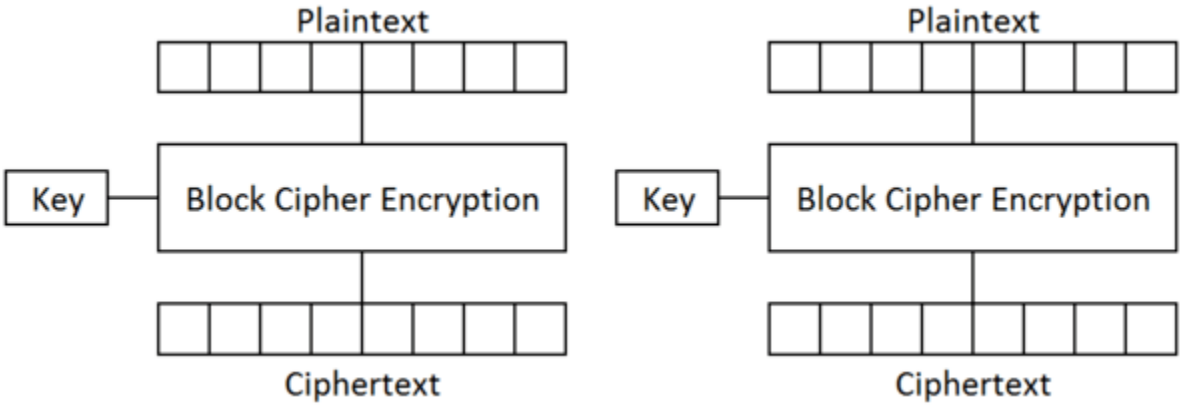


Fig. 3.3 Block Cipher

In a block cipher, the plaintext is broken into blocks of a set length and the bits in each block are encrypted together. This Transformation takes place under the action of a user-provided secret key. Decryption is performed by applying the reverse transformation to the cipher text block using the same secret key. The block lengths can be 128, 192 or 256 bits. Examples of block cipher are AES, DES and Blowfish.

Characteristics of Block Cipher:

- Breaks the plaintext into blocks and encrypts each with the same algorithm
- The properties of a cipher should contain confusion and diffusion
- Uses a Feistel network or a Substitution-Permutation network
- Are more suitable for software implementations, because they work with blocks of data which is usually the width of a data bus (64 bits).

3.2 ASYMMETRIC KEY OR PUBLIC KEY SYSTEM

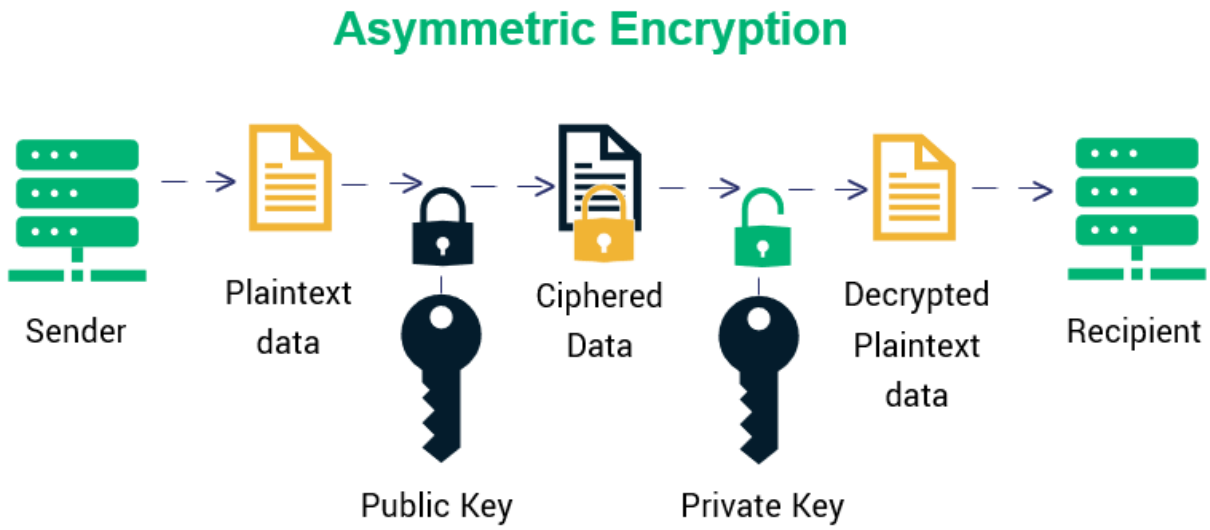


Fig. 3.4 Asymmetric Encryption

Asymmetric encryption involves the use of multiple keys for data encryption and decryption. To be exact, the asymmetric encryption method comprises two encryption keys that are mathematically related to each other. These keys are known as the public key and private key. As a result, the asymmetric encryption method is also known as “public key cryptography”.

In communication between Alice and Bob, Alice uses the public key K_e of Bob to encrypt the message, in a way that only B (neither A) can decrypt this message using his private key K_d. This system is also used to sign a message digitally. Rivest-Shamir-Adleman (RSA) is widely used asymmetric key algorithm for decades and Elliptic Curve Cryptography (ECC) as an alternative to RSA which offers highest security with small bit length of key.

CHAPTER 4

ADVANCED ENCRYPTION STANDARD

DES was deprecated in 2002 as it was proved insecure. This is mainly due to its smaller 56-bit key size. DES Keys can be broken in 56 hours using the EFF DES cracker - Deep Crack. The algorithm is believed to be practically secure in the form of Triple DES (3DES), although there are theoretical attacks. In recent years, the cipher has been superseded by the Advanced Encryption Standard (AES).

The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.

In January 1997, the National Institute of Standards and Technology (NIST) announced the initiation of an effort to develop the AES and made a formal call for algorithms on September 12, 1997. After reviewed the results of this Preliminary research, the algorithms MARS, RC6TM, Rijndael, Serpent and Two fish were selected as finalist. And further reviewed public analysis of the finalist, NIST has decided to propose Rijndael as the new Advanced Encryption Standard (AES) on 2nd October 2000. It was adopted by the U.S. government in 2001. In the summer of 2001, AES replaced the aging DES as the Federal Information Processing Standard (FIPS). It is also the standard for block cipher in ISO/IEC 18033-3. DES is seen as reaching the end of its life, as cracking of its cipher is seen to be more tractable on current computer hardware. The AES algorithm will be used for many applications within the government an in the private sector.

AES is a subset of the Rijndael block cipher developed by two Belgian cryptographers, Vincent Rijmen and Joan Daemen, who submitted a proposal to NIST during the AES selection process. Rijndael is a family of ciphers with different key and block sizes. For AES, NIST selected three members of the Rijndael family, each with a block size of 128 bits, but three different key lengths: 128, 192 and 256 bits.

4.1 MATHEMATICS INVOLVED

This section provides a brief introduction to the fundamental mathematical concepts of finite fields needed to understand. Several operations in AES are defined at byte level, with bytes representing elements in the finite field $gf(2^8)$. Other operations are defined in terms of 4-byte words.

4.1.1 GALOIS FIELD

The elements of Galois Field $gf(p^n)$ is defined as

$$gf(p^n) = (0, 1, 2, \ldots, p-1) \cup$$

$$(p, p+1, p+2, \ldots, p+p-1) \cup$$

$$(p^2, p^2+1, p^2+2, \ldots, p^2+p-1) \cup \ldots \cup$$

$$(p^{n-1}, p^{n-1}+1, p^{n-1}+2, \ldots, p^{n-1}+p-1)$$

where $p \in P$ and $n \in Z^+$. The order of the field is given by p^n while p is called the characteristic of the field. On the other hand, *gf*, as one may have guessed it, stands for Galois Field. Also note that the degree of polynomial

of each element is at most n − 1.

4.1.2 2^8 GALOIS FIELD TRANSFORMATION

The byte value in AES is represented as a set of bits (0 or 1) and is represented as the collection of bits separated by comma as {b7, b6, b5, b4, b3, b2, b1, b0}. These bytes are interpreted as finite field elements using polynomial representation as

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

Example:

The byte with hexadecimal value '85' (binary 01010101) corresponds with polynomial

$$x^6 + x^4 + x^2 + 1$$

4.1.3 FINITE FIELD ADDITION AND SUBTRACTION

An addition in Galois Field is pretty straightforward. Suppose *f*(p) and *g*(p)

are polynomials in $gf(p^n)$. Let $A = a_{n-1}a_{n-2} \ldots a_1a_0$, $B = b_{n-1}b_{n-2} \ldots b_1b_0$,

and $C = c_{n-1}c_{n-2} \ldots c_1c_0$ be the coefficients of *f*(p), *g*(p), and *h*(p) = *f*(p) +

g(p) respectively. If a_k, b_k, and c_k are the coefficients of p^k in *f*(p), *g*(p), and *h*(p) respectively then

$$c_k = a_k + b_k \pmod p$$

Similarly, if *h*(p) = *f*(p) - *g*(p) then

$$c_k = a_k - b_k \pmod p$$

where $0 \leq k \leq n - 1$. Since computer works in $gf(2^8)$, if a_k and b_k refer to the k^{th} bit in the bytes we wish to add then c_k, the k^{th} bit in the resulting byte, is given by

$$c_k = a_k + b_k \pmod 2$$

Since $0 + 1 = 1 + 0 = 1 \pmod 2 = 1$ and $0 + 0 = 0 \pmod 2 = 1 + 1 = 2 \pmod 2 = 0$, we may think of addition as exclusive-or operation which is also known as XOR operation. That is, XOR operation returns 0 if both entries are equal and returns 1 otherwise which also means that subtraction and addition is the same in Galois Field whose characteristic field is 2. Due to the nature of Galois Field, addition and subtraction of two bytes will not go any bigger than 11111111 = 255, the biggest value one byte can store, and is therefore a safe operation.

Example:

Suppose we are working in $gf(2^8)$, then 83 + 249 is

$$83 + 249 = (2^6 + 2^4 + 2^1 + 2^0) + (2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^0)$$

$$= 2^7 + 2 \cdot 2^6 + 2^5 + 2 \cdot 2^4 + 2^3 + 2^1 + 2 \cdot 2^0$$

$$= 27 + 25 + 23 + 21$$

$$= 169$$

Alternatively, from binary numeral system perspective,

$83 + 249 = 01010011 + 11111001$

$= 10101010$

$= 169$

and the results coincide.

4.1.4 FINITE FIELD MULTIPLICATION AND MULTIPLICATIVE INVERSE

Multiplication in Galois Field, however, requires more tedious work. Suppose $f(p)$ and $g(p)$ are polynomials in $gf(p^n)$ and let $m(p)$ be an irreducible polynomial (or a polynomial that cannot be factored) of degree at least n in $gf(p^n)$. We want $m(p)$ to be a polynomial of degree at least n so that the product of two $f(p)$ and $g(p)$ does not exceed 11111111 = 255 as the product needs to be stored as a byte. If $h(p)$ denotes the resulting product then

$$h(p) = (f(p) \cdot g(p)) \pmod{m(p)}$$

On the other hand, the multiplicative inverse of $f(p)$ is given by $a(p)$ such that

$$(f(p) \cdot a(p)) \pmod{m(p)} = 1$$

Note that figuring out the product of two polynomials and the multiplicative inverse of a polynomial requires both reducing coefficients modulo p and reducing polynomials modulo m(p). The reduced polynomial can be calculated easily with long division while the best way to compute the multiplicative inverse is by using Extended Euclidean Algorithm.

Example:

$\{53\} \bullet \{CA\}$

$(x^6 + x^4 + x + 1) \bullet (x^7 + x^6 + x^3 + x) =$

$(x^{13} + x^{12} + x^9 + x^7) + (x^{11} + x^{10} + x^7 + x^5) + (x^8 + x^7 + x^4 + x^2) + (x^7 + x^6 + x^3 + x)$

$= x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x$

and

$(x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x) \% (x^8 + x^4 + x^3 + x + 1)$

$= (11111101111110 \bmod 100011011)$

$= \{3F7E \bmod 11B\}$

$= \{01\} = 1$ (decimal),

which can be demonstrated through long division (shown using binary notation, since it lends itself well to the task. Notice that exclusive OR is applied in the example and not arithmetic subtraction, as one might use in grade-school long division.

11111101111110 (mod) 100011011

^100011011

1110000011110

^100011011

110110101110

^100011011

10101110110

^100011011

0100011010

^100011011

00000001

(The elements {53} and {CA} are multiplicative inverses of one another since their product is 1.)

4.2 THE STATE

AES breaks data into states or matrix of bytes of pre-determined size and encrypt every state independently of each other. Putting together bytes into a state has no cryptographic significance, yet it is an important process without whom other operations cannot be conducted. In Rijndael with a 128-bit key, an array or a matrix with 4 rows and 4 columns is formed where each entry is a byte or 8 bits so that there are essentially 16 bytes in total. Additionally, we may also think of a 128-bit state space as a vector in $gf(2^8)^{16}$ where each component of the vector is a byte.

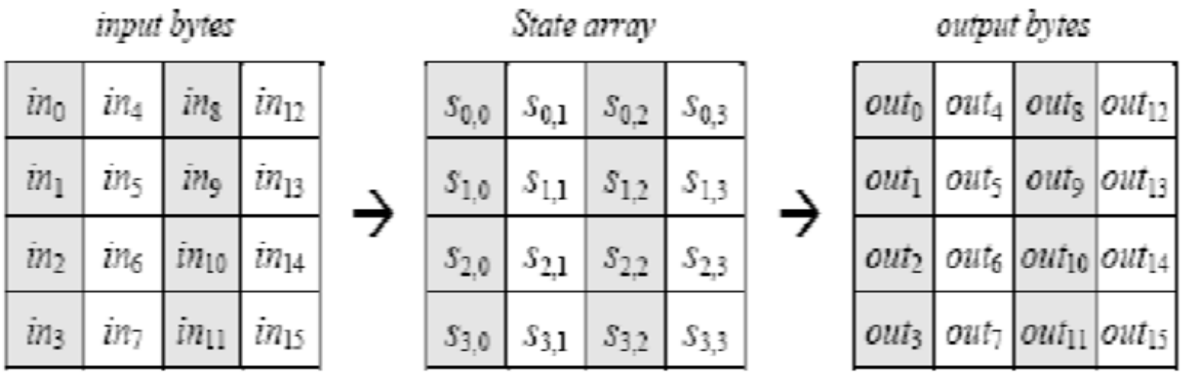


Fig. 4.1 State Matrix

4.2.1 THE STATE AS AN ARRAY OF COLUMNS

The four bytes in each column of the State array form 32-bit words. The state can hence be interpreted as a one-dimensional array of 32-bit words (columns) w_0, w_1, w_2 and w_3.

$$w_0 = s_{00}\ s_{10}\ s_{20}\ s_{30}$$

$$w_1 = s_{01}\ s_{11}\ s_{21}\ s_{31}$$

$$w_2 = s_{02}\ s_{12}\ s_{22}\ s_{32}$$

$$w_3 = s_{03}\ s_{13}\ s_{23}\ s_{33}$$

4.3 STEPS INVOLVED IN AES ALGORITHM

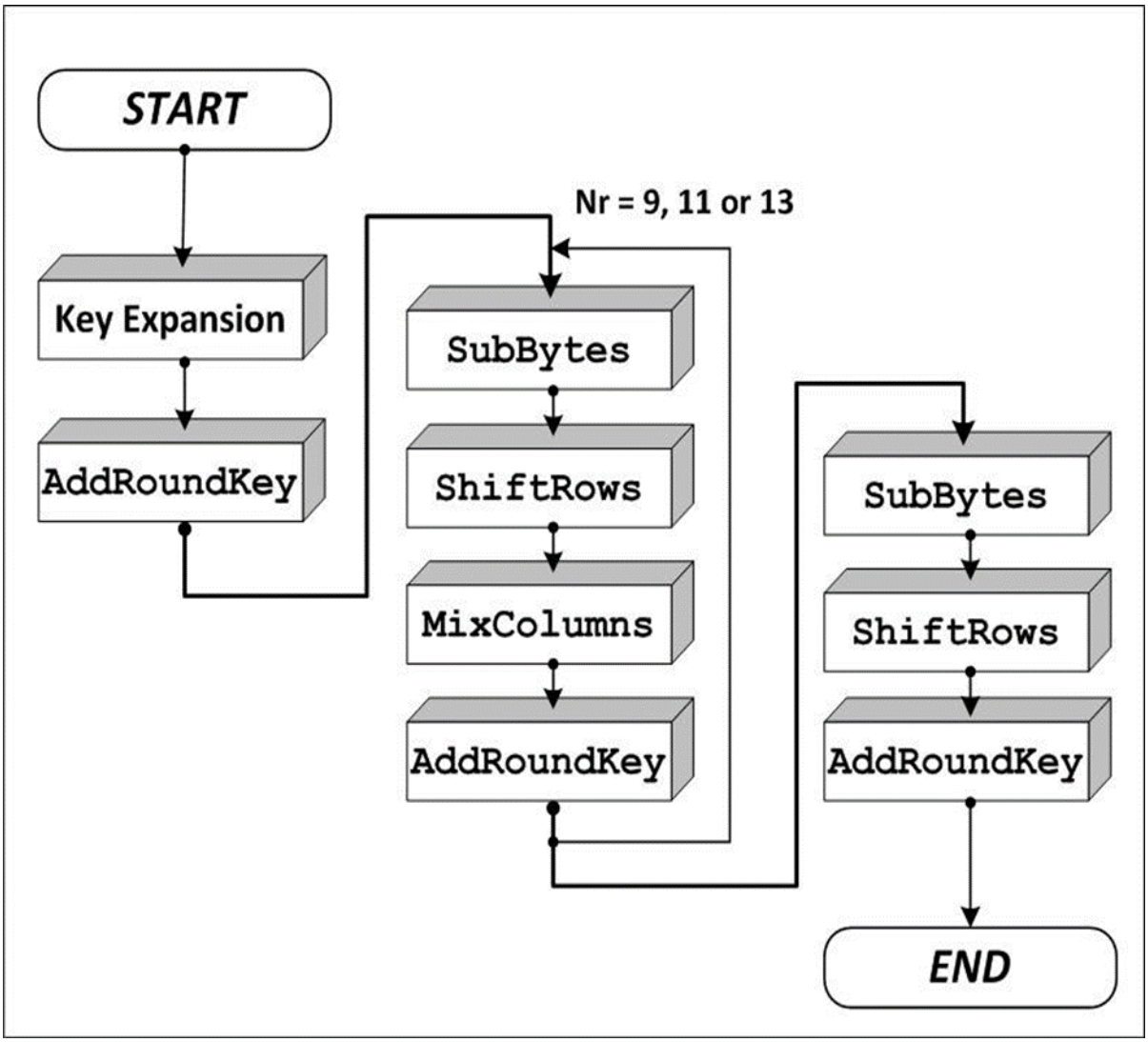


Fig. 4.2 AES General Structure

1) Key Expansions — round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.

2) Initial Round

AddRoundKey—each byte of the state is combined with a block of the round key using bitwise XOR.

3) Rounds

a) SubBytes

b) ShiftRows

c) MixColumns

d) AddRoundKey

4) Final Round

a) SubBytes

b) ShiftRows

c) AddRoundKey

4.3.1 SUBBYTES (SUBSTITUTE BYTES)

In this method each byte, each element in the matrix is replaced using the Rijndael's S-Box (Substitution Box). Each byte is replaced with its multiplicative inverse.

	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

Fig. 4.3 S-Box

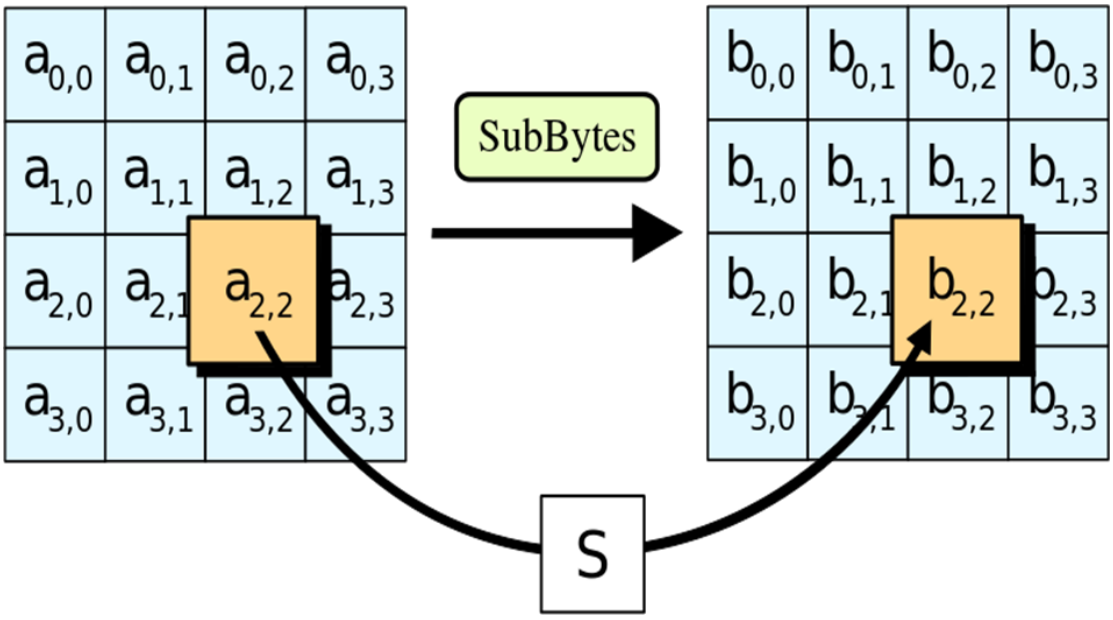


Fig. 4.4 SubBytes Transformation

For inverting the SubBytes operation while decrypting, substitute the bytes in the state matrix using the following s-box(inverse s-box).

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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	al	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	9b	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

Fig. 4.5 Inverse S-Box

4.3.2 SHIFT ROWS

ShiftRows is perhaps the simplest operation to scramble data. Like the name suggest, ShiftRows shifts row n to the left by n−1 unit, so that the first row remains unchanged while the second row is shifted to the left by 1, third row by 2, and fourth row by 3.

That is, if we let S and S' to denote the state before and after performing ShiftRows then if S' is given by

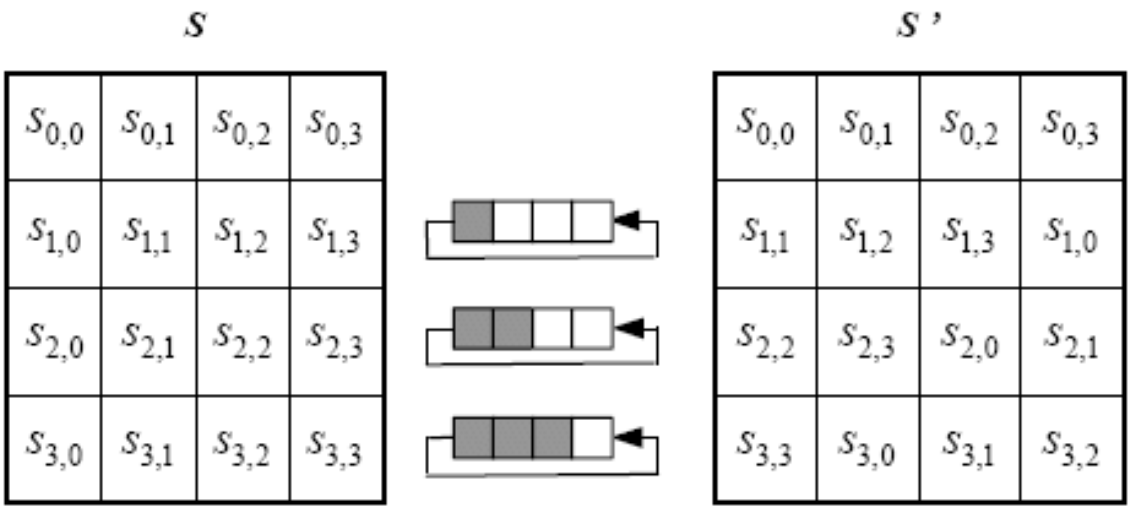


Fig. 4.6 Shift Rows Transformation

For inverting the operation, the same sequence of operations is performed but the shifting is done to the right.

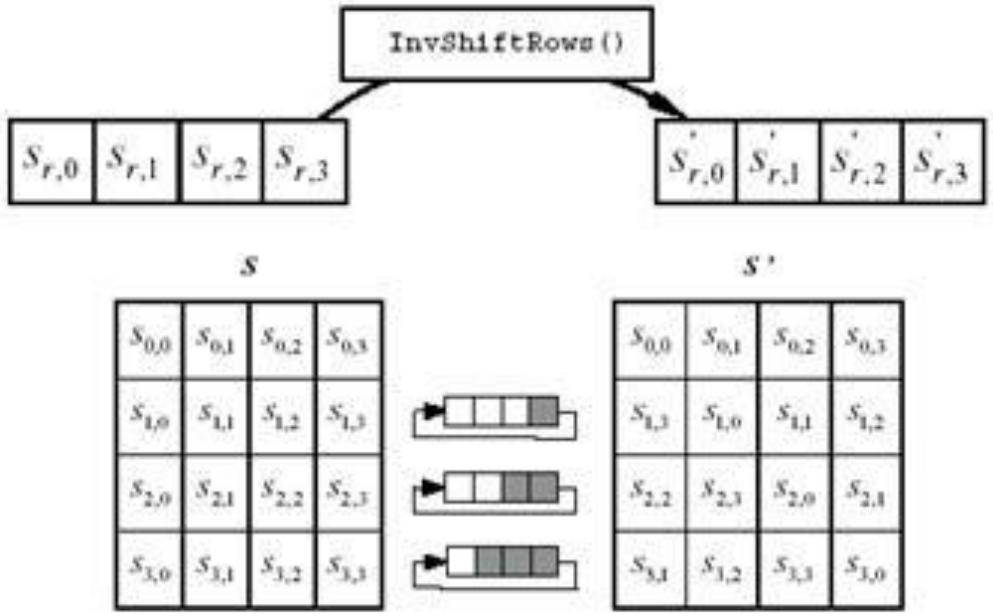


Fig. 4.7 Inverse Shift Rows Transformation

4.3.3 MIX COLUMNS

The next method messes with columns instead of rows. MixColumns takes each column in the state and perform linear transformation on it. Since each column has 4 rows, the matrix transformation has to be 4 × 4 and is defined as follow

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

Fig. 4.8 Mix Columns Matrix

And since linear transformation has an inverse, this operation is invertible. The matrix used to revert the state back to its original standing is given by

$$\begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix}$$

Fig. 4.9 Inverse Mix Columns Matrix

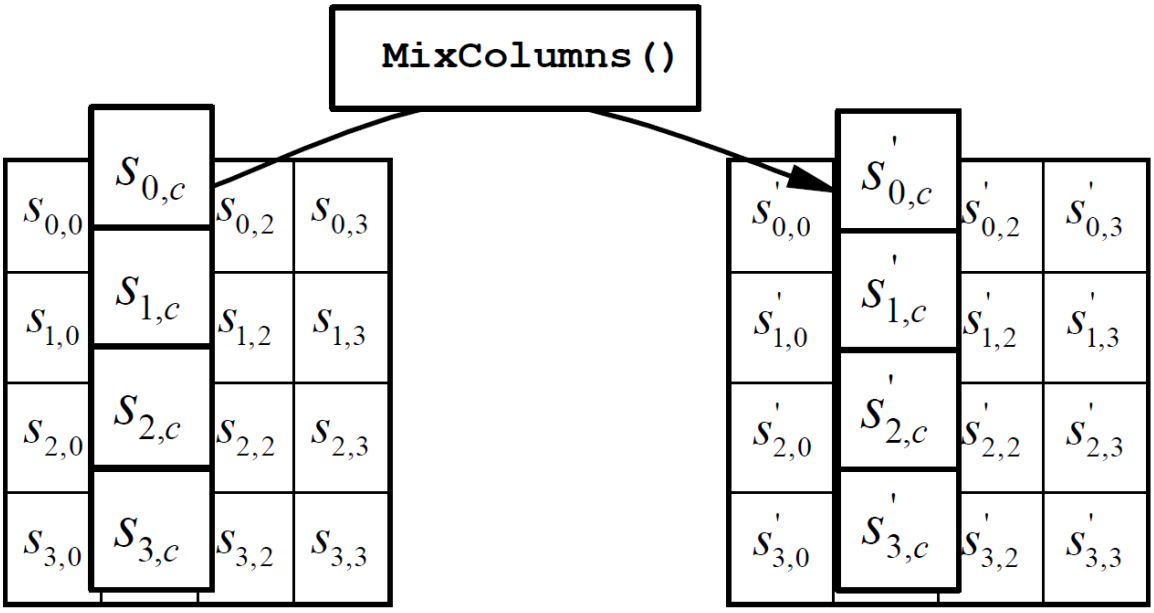


Fig. 4.10 Mix Columns Transformation

4.3.4 ADD ROUND KEY

AddRoundKey is the most important stage as it is the stage that provides uniqueness to the encryption thus is inevitably a complex operation by itself. The output of AddRoundKey fully depends on the key or the password specified by the user. In this stage a subkey, which is the same size as the state, is computed from the main key using Rijndael's Key Schedule. Once a subkey is derived, the sum of the subkey and the state is calculated.

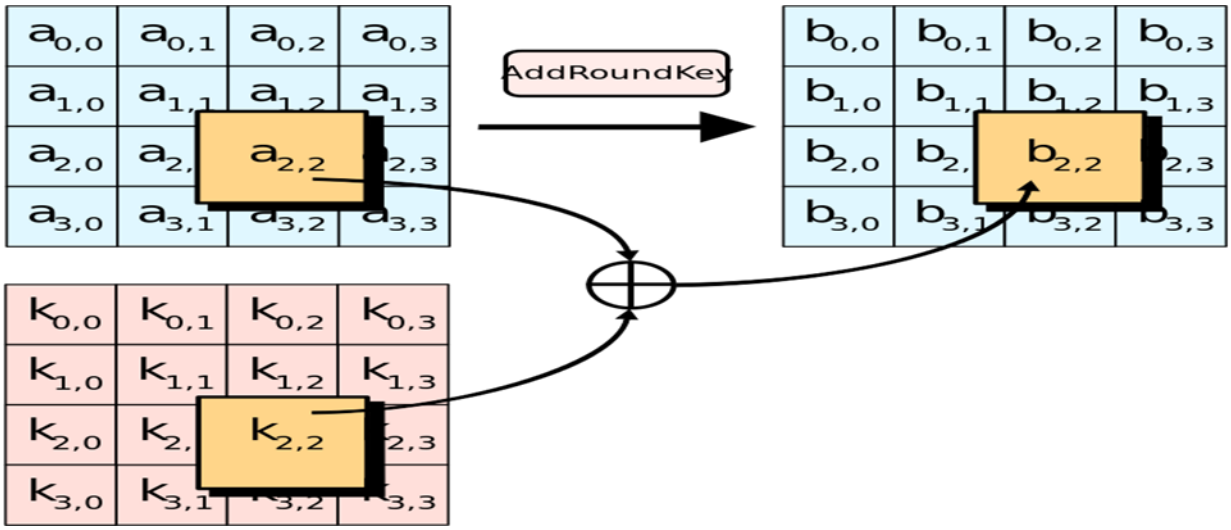


Fig. 4.11 Add Round Key Transformation

4.3.5 KEY EXPANSION

This process consists of three parts: Rotate, RCon or RC (Round Constant), and SubByes. The first part is to rotate or to shift the bytes that form the keyword 8 bits to the left, which is similar to what happens to the second row in ShiftRows. The second part is to apply sub-operation called RCon; And the third part is to perform Rijndael's S-Box whose details have been dissected in SubBytes. Next step in this operation is to expand the main key until we have enough subkeys.

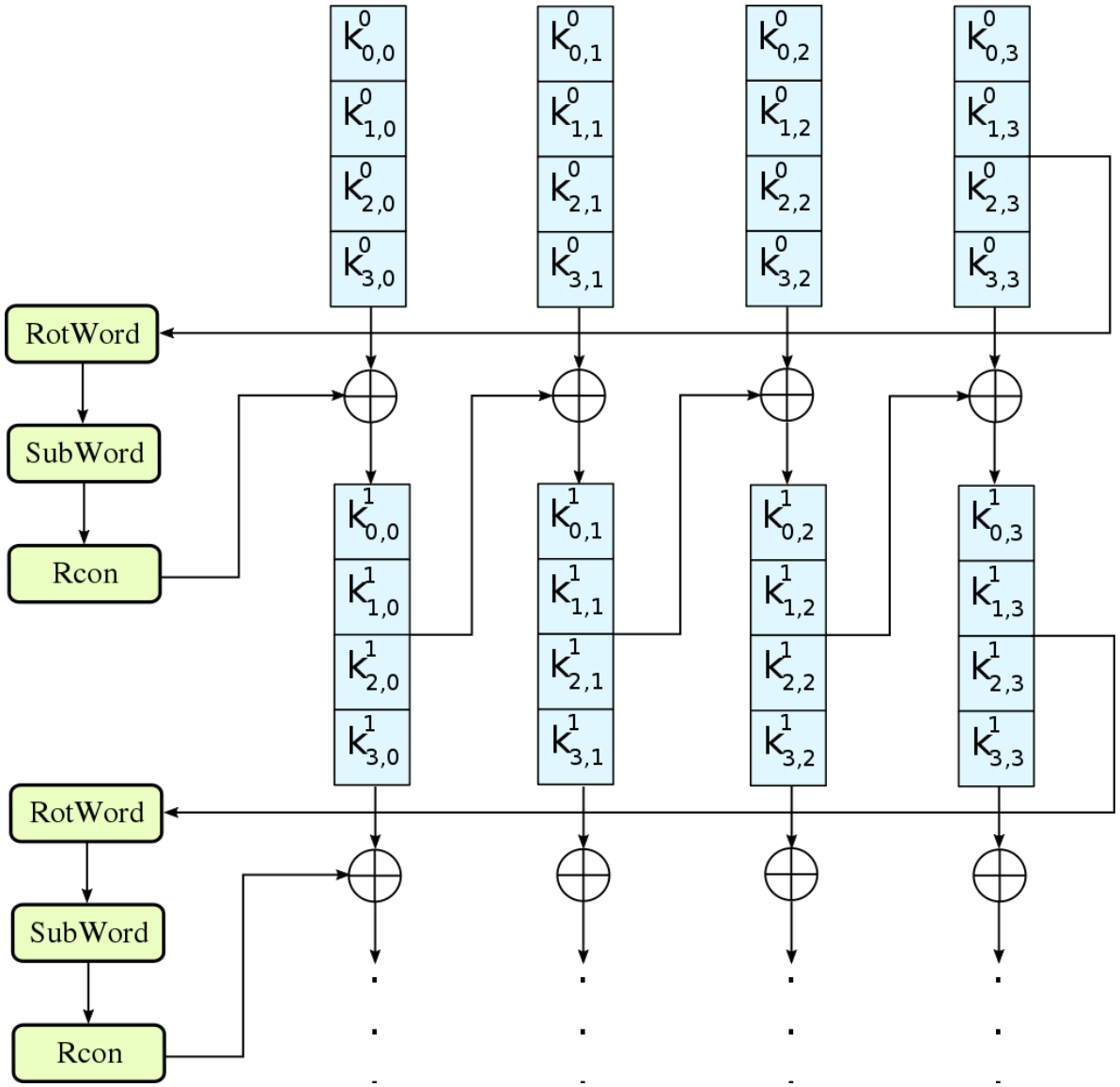


Fig. 4.12 Block Diagram of Key Expansion

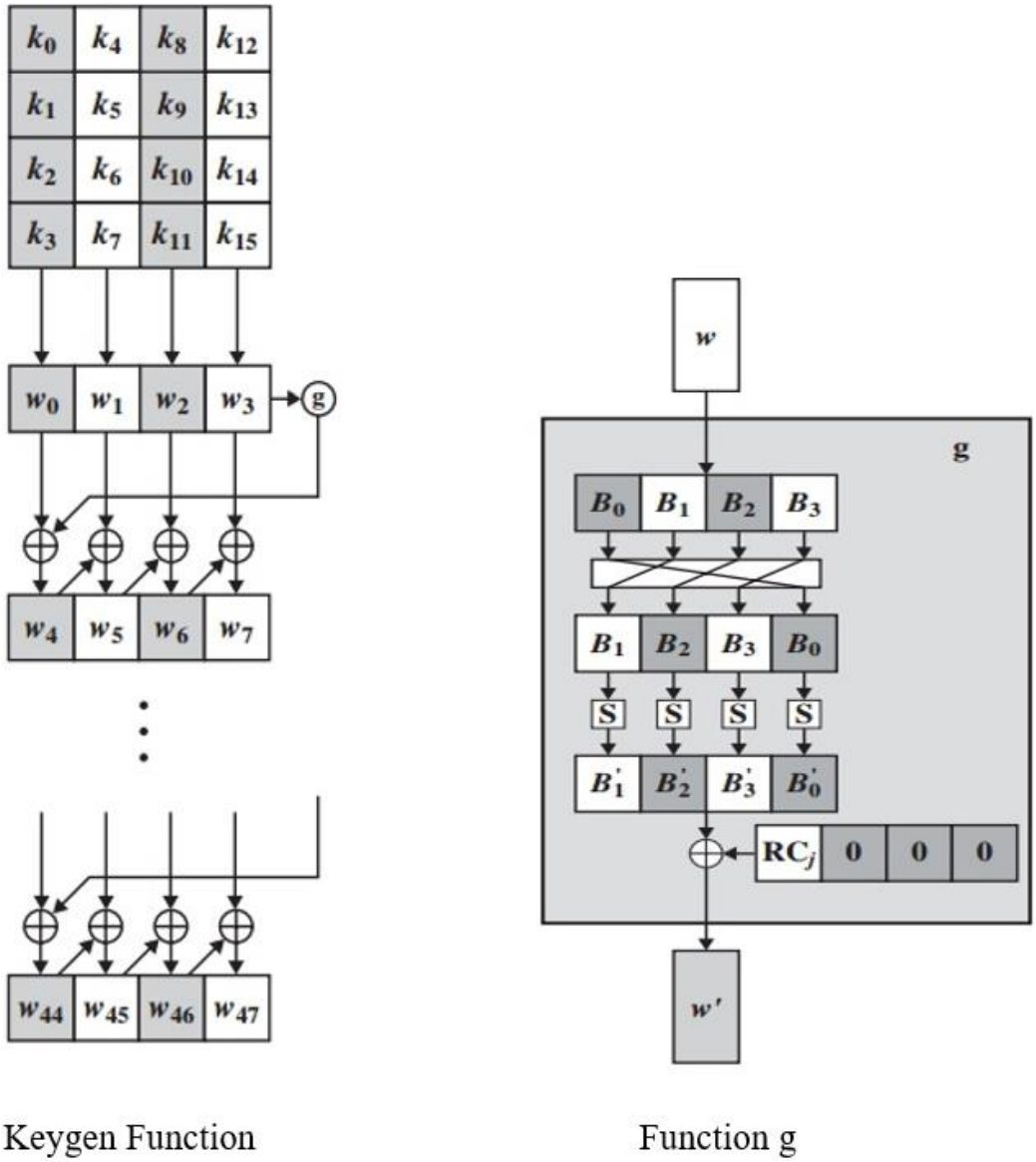


Fig. 4.13 Functional Representation of Key Expansion

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

Fig. 4.14 RCon/RC for 10 rounds

4.4 SPECIFICATIONS OF AES

- The length of the input block, the output block and the State is always 128 bits.
- The length of the Cipher Key can be either 128, 192, or 256 bits.
- The number of rounds to be performed during the execution of the algorithm is dependent on the key size.

Length of Key (bits)	Number of Rounds
128	10
192	12
256	14

Table 4.1 AES Relation Between Key Length and Number of Rounds

4.5 CRYPTANALYSIS RESULTS OF AES

In AES, there are 3.4×10^{38} possible 128-bit keys and 1.1×10^{77} possible 256-bit keys. Pure brute forcing would take billions of years to crack the smaller 128-bit key. No one can be sure how long the AES - or any other cryptographic algorithm - will remain secure. However, according to NIST's estimates AES should serve at least 30 years without any changes in the algorithm.

The computational complexity of cracking an AES-128 key using the Biclique attack is $2^{126.18}$ using publicly known attacks while it is $2^{254.4}$ for AES-256. This attack is a variant of the meet-in-the-middle (MITM) method of cryptanalysis. It uses a biclique structure.

4.6 AES ENCRYPTION (ROUND BY ROUND)

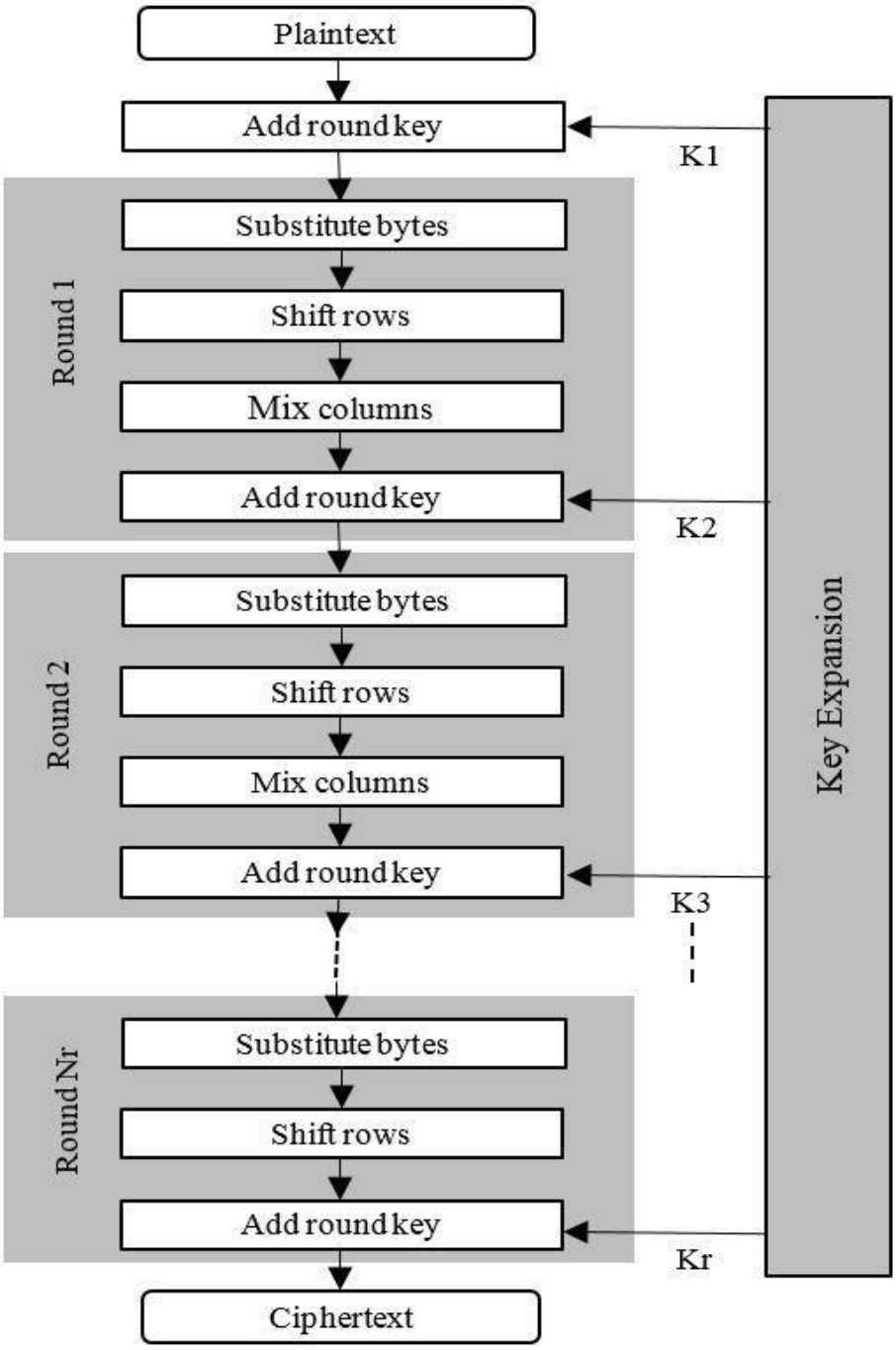
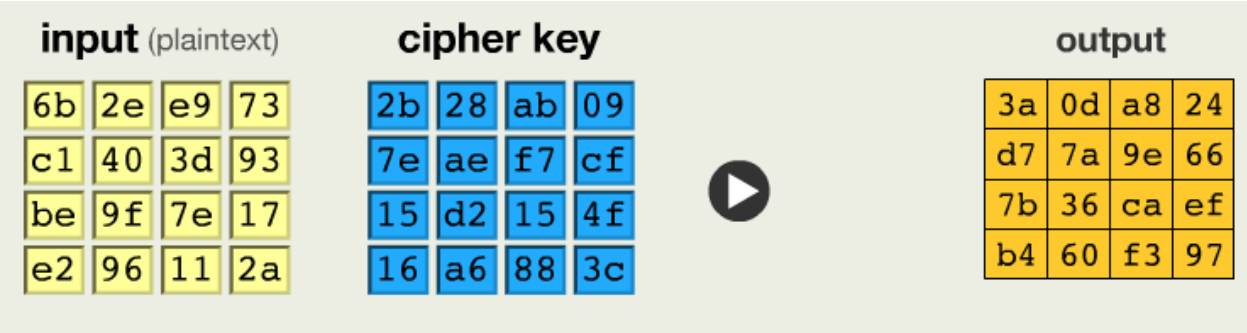


Fig. 4.15 AES Encryption Block Diagram

The following figures show the values in the state matrix as the cipher progresses for a key length of 128 bits. The columns from 1 to 5 show the results of AddRoundKey, SubBytes, ShiftRows, MixColumns and the Round Key itself respectively.



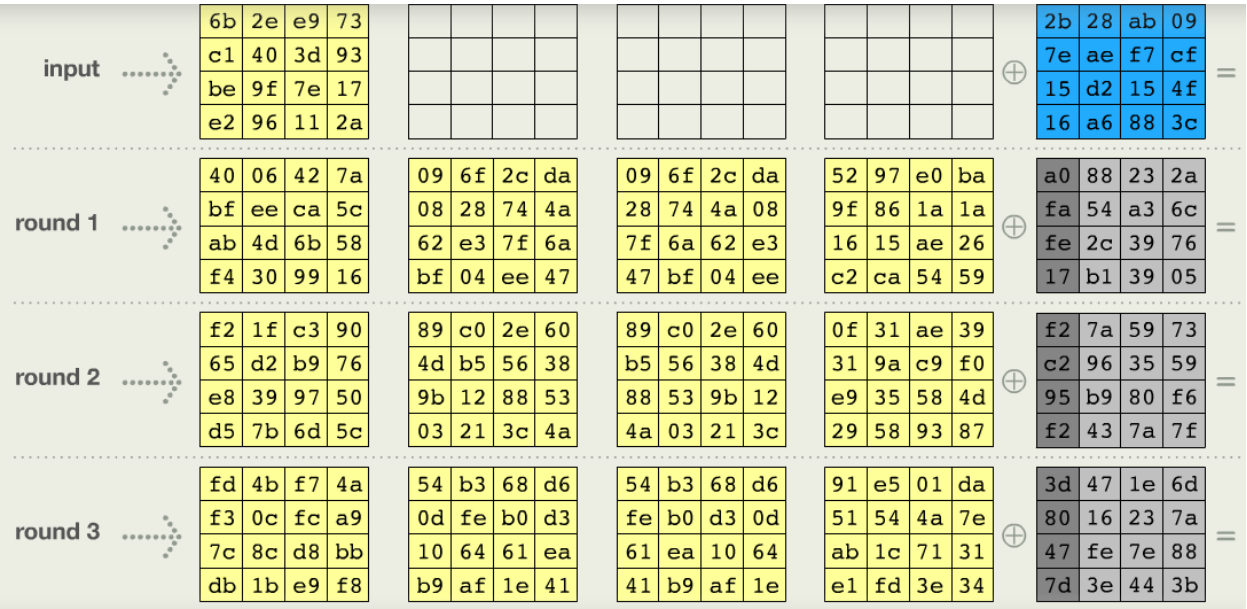


Fig. 4.16 AES-128 Encryption Example

***** The above image was captured from Rijndael-Inspector-v1.1.

4.7 AES DECRYPTION (ROUND BY ROUND)

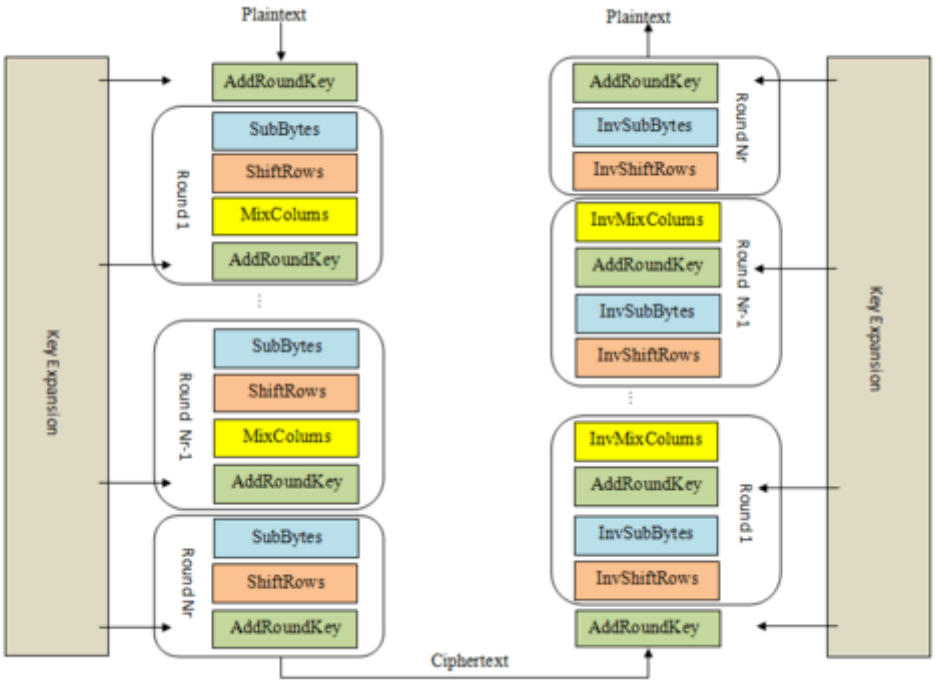
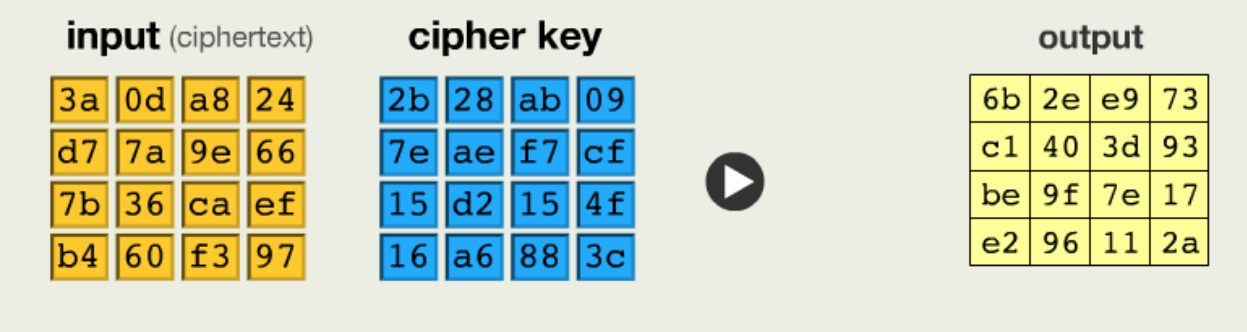


Fig. 4.17 AES Decryption as an inverse of Encryption

The following figures show the values in the state matrix as the decryption progresses for a key length of 128 bits. The columns from 1 to 5 show the results of InvSubBytes, InvShiftRows, InvMixColumns, Round Key itself and the result of AddRoundKey respectively.


input (ciphertext)
3a 0d a8 24
d7 7a 9e 66
7b 36 ca ef
b4 60 f3 97
cipher key
2b 28 ab 09
7e ae f7 cf
15 d2 15 4f
16 a6 88 3c
output
6b 2e e9 73
c1 40 3d 93
be 9f 7e 17
e2 96 11 2a

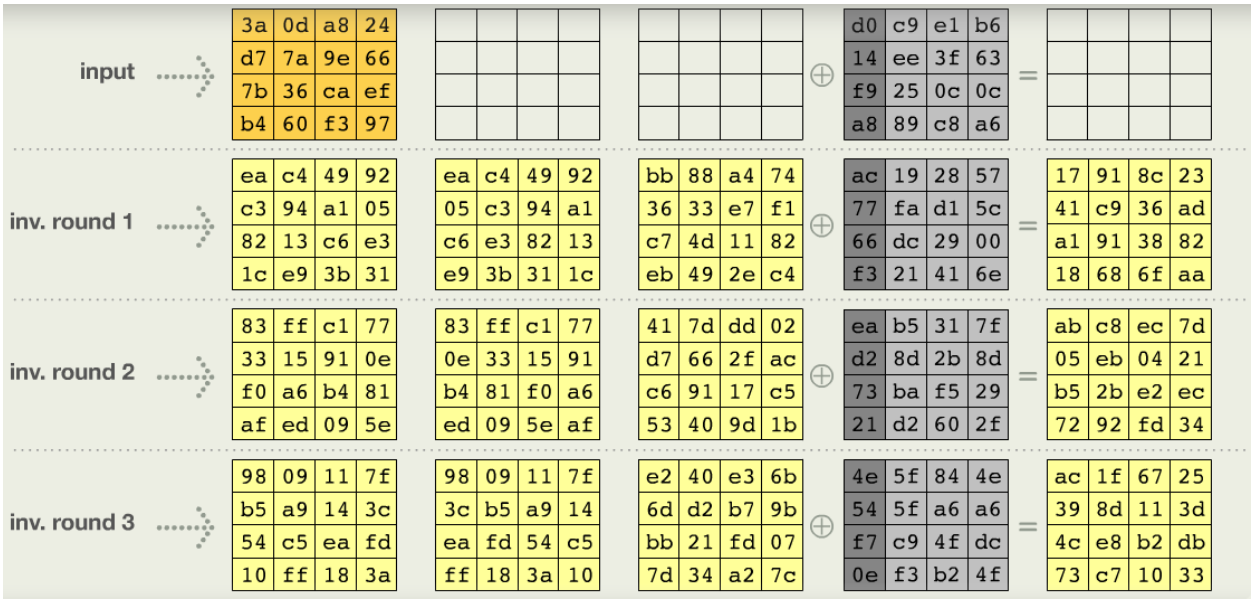

input
3a 0d a8 24
d7 7a 9e 66
7b 36 ca ef
b4 60 f3 97
d0 c9 e1 b6
14 ee 3f 63
f9 25 0c 0c
a8 89 c8 a6
inv. round 1
ea c4 49 92
c3 94 a1 05
82 13 c6 e3
1c e9 3b 31
ea c4 49 92
05 c3 94 a1
c6 e3 82 13
e9 3b 31 1c
bb 88 a4 74
36 33 e7 f1
c7 4d 11 82
eb 49 2e c4
ac 19 28 57
77 fa d1 5c
66 dc 29 00
f3 21 41 6e
17 91 8c 23
41 c9 36 ad
a1 91 38 82
18 68 6f aa
inv. round 2
83 ff c1 77
33 15 91 0e
f0 a6 b4 81
af ed 09 5e
83 ff c1 77
0e 33 15 91
b4 81 f0 a6
ed 09 5e af
41 7d dd 02
d7 66 2f ac
c6 91 17 c5
53 40 9d 1b
ea b5 31 7f
d2 8d 2b 8d
73 ba f5 29
21 d2 60 2f
ab c8 ec 7d
05 eb 04 21
b5 2b e2 ec
72 92 fd 34
inv. round 3
98 09 11 7f
b5 a9 14 3c
54 c5 ea fd
10 ff 18 3a
98 09 11 7f
3c b5 a9 14
ea fd 54 c5
ff 18 3a 10
e2 40 e3 6b
6d d2 b7 9b
bb 21 fd 07
7d 34 a2 7c
4e 5f 84 4e
54 5f a6 a6
f7 c9 4f dc
0e f3 b2 4f
ac 1f 67 25
39 8d 11 3d
4c e8 b2 db
73 c7 10 33

Fig 4.18 AES-128 Decryption Example

***** The above image was captured from Rijndael-Inspector-v1.1.

CHAPTER 5

IMPLEMENTATION

Both the variants 128-bit and the 256-bit were implemented in hardware, Xilinx xc7z020-clg484-1 FPGA using the Xilinx Vivado 2019.1 software. Verilog was used for programming. The design implemented handles both encryption and decryption. It tries to balance resource utilization and performance.

Features of Implementation:

- All keys are generated at the beginning of the encryption process.
- SubBytes and ShiftRows transformations were combined into a single transformation to improve performance (same with InvSubBytes and InvShiftRows).
- Standard NIST testcases were used for testing.
- Loop unrolling optimization was used.

The FPGA was contained in the Zedboard Zynq Evaluation and Development Kit which contains the Zynq-7000 SoC. The SoC has 512 MB DDR3 RAM.

The Specification is the FPGA is given in table 5.1.

Resource	Available
LUT	53200
FF	106400
BRAM	140
IO	200
BUFG	32

Table 5.1 xc7z020-clg484-1 FPGA Specifications

Zedboard, with its IOs marked, is pictured in Fig. 5.1 and its block diagram in Fig. 5.2. The SoC has a clock speed of 100MHz.

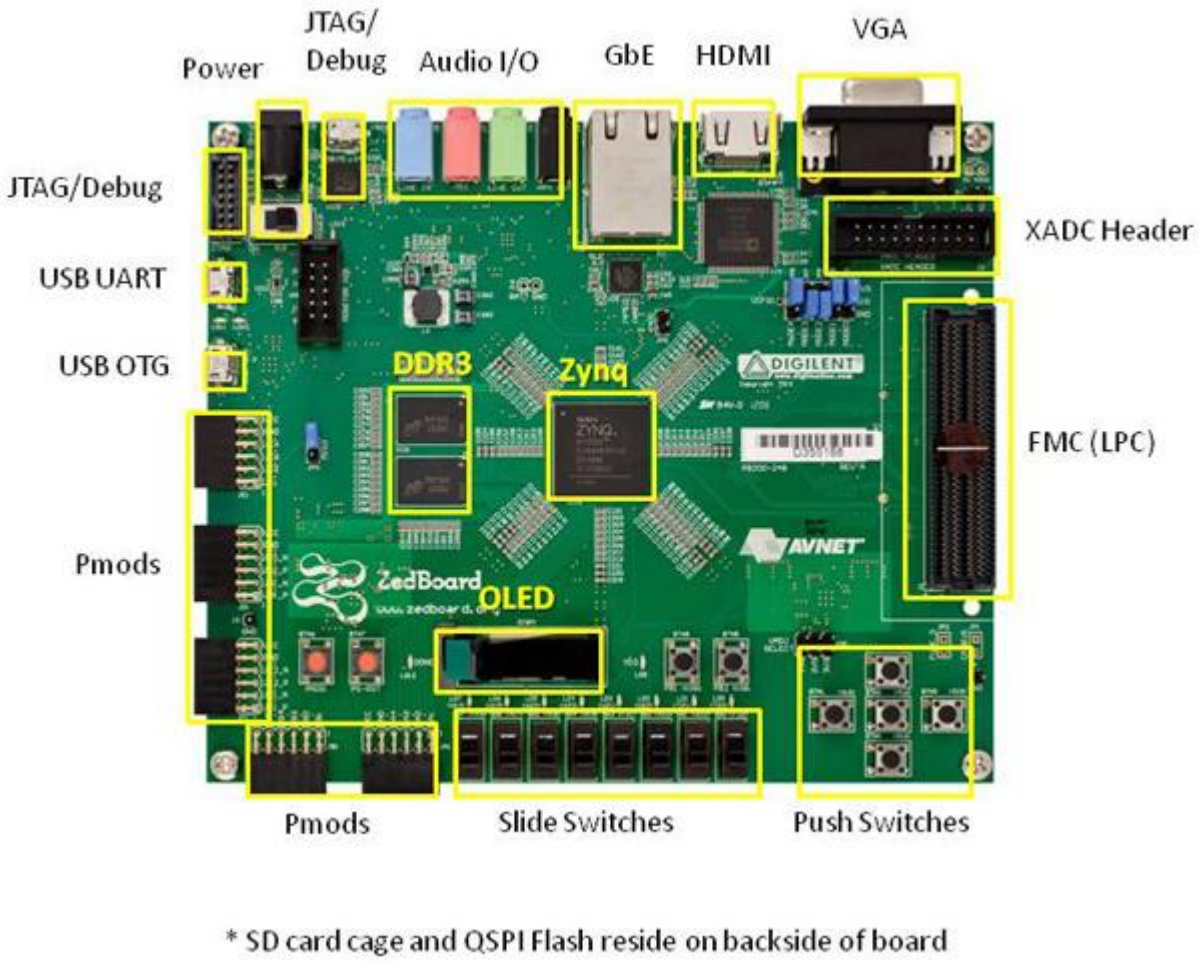


Fig. 5.1 Zedboard

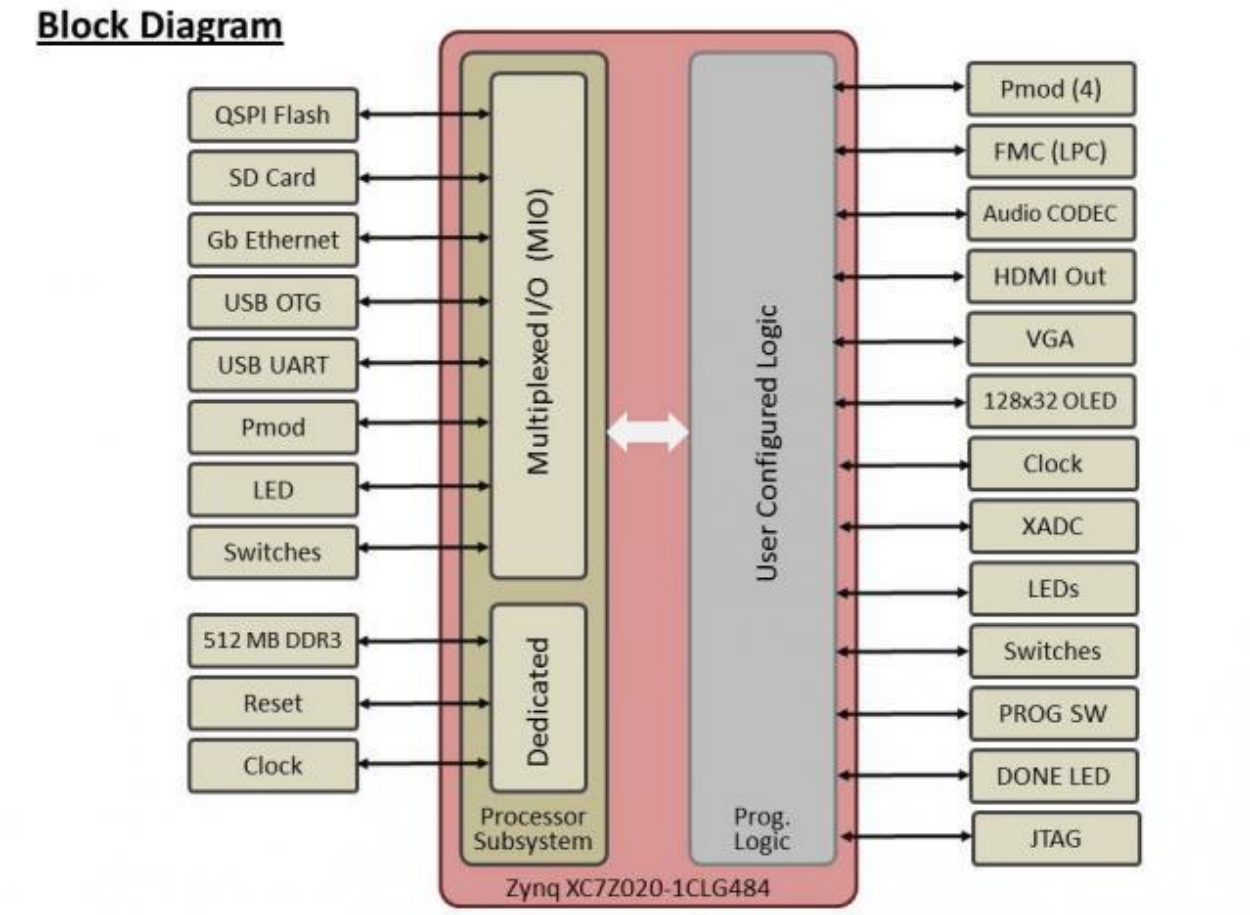


Fig. 5.2 Zedboard Block Diagram

All the testcases used are pictured in Fig. 5.3.

```
128-bit:
key = 2b7e151628aed2a6abf7158809cf4f3c

PT1 = 6bc1bee22e409f96e93d7e117393172a
CT1 = 3ad77bb40d7a3660a89ecaf32466ef97

PT2 = ae2d8a571e03ac9c9eb76fac45af8e51
CT2 = f5d3d58503b9699de785895a96fdbaaf

PT3 = 30c81c46a35ce411e5fbc1191a0a52ef
CT3 = 43b1cd7f598ece23881b00e3ed030688

PT4 = f69f2445df4f9b17ad2b417be66c3710
CT4 = 7b0c785e27e8ad3f8223207104725dd4



256-bit:
key = 603deb1015ca71be2b73aef0857d7781
      1f352c073b6108d72d9810a30914dff4

PT1 = 6bc1bee22e409f96e93d7e117393172a
CT1 = f3eed1bdb5d2a03c064b5a7e3db181f8

PT2 = ae2d8a571e03ac9c9eb76fac45af8e51
CT2 = 591ccb10d410ed26dc5ba74a31362870

PT3 = 30c81c46a35ce411e5fbc1191a0a52ef
CT3 = b6ed21b99ca6f4f9f153e7b1beafed1d

PT4 = f69f2445df4f9b17ad2b417be66c3710
CT4 = 23304b7a39f9f3ff067d8d8f9e24ecc7
```

Fig. 5.3 Testcases

5.1 AES-128 RESULTS

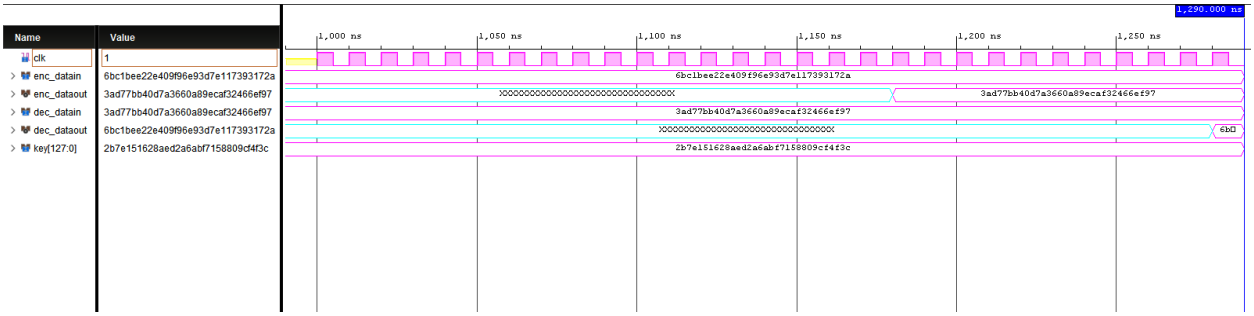


Fig. 5.4 AES-128 Waveform

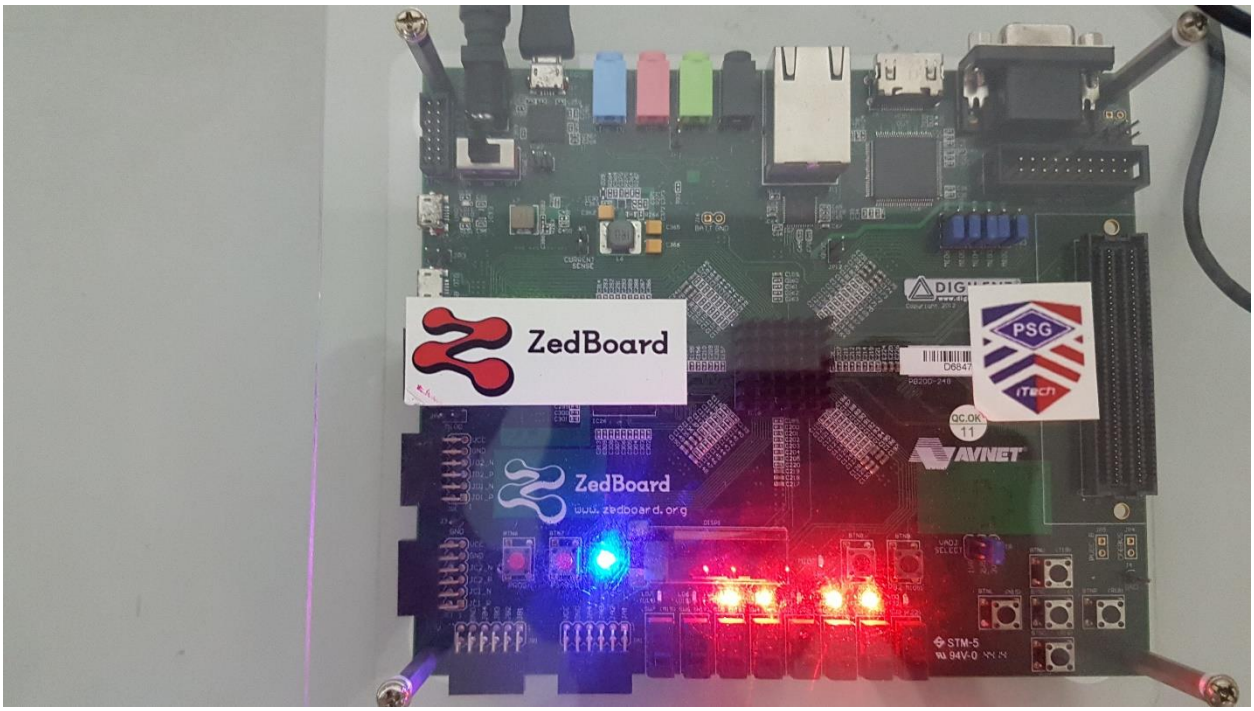


Fig. 5.5 AES-128 implemented in Hardware

The working of AES-128 design can be seen from the figures 5.4 and 5.5. The waveform displays clock, key for cipher, inputs and outputs for both encryption and decryption. The outputs are produced as the clock progresses. The values displayed correspond to the first test case of AES-128, shown in Fig. 5.3. In hardware, the first 4 bits of the ciphertext and decrypted plaintext of the first testcase displayed using the 8 Red LEDs can be seen, i.e. 0x36 or 00110110.

Process Name	Clock Cycles
Key Expansion	10
One Round of Encryption/Decryption	2
Full Encryption Process	19
Full Decryption Process	29

Fig. 5.6 Performance of AES-128 Design

The performance of the AES-128 design in terms of clock cycles is shown in Fig. 5.6. The key expansion stage has to generate 10 keys and each generation took exactly 1 clock cycle to output its corresponding round key. During encryption, the MixColumns transformation takes 1 clock cycle and as the SubBytes and the ShiftRows are combined, they together take 1 clock cycle to complete (same during decryption as InvSubBytes and InvShiftRows are combined). So, encryption and decryption rounds take 2 clock cycles each. The actual encryption process (excluding the key expansion phase) proceeds as soon as the first round key is generated and finishes at 19 clock cycles. The actual decryption process can only start after the last key is generated i.e. after 10 clock cycles and thus, taking 10 more clock cycles than the encryption process and finishing at 29 clock cycles.

Resource	Utilization	Available	Utilization %
LUT	3854	53200	7.24
FF	1691	106400	1.59
BRAM	70.50	140	50.36
IO	9	200	4.50
BUFG	1	32	3.13

Fig. 5.7 Resource Utilization of AES-128 Design in Hardware

The resource utilization of the design is shown in Fig. 5.7 (captured in Vivado 2019.1). One clock source is used. 3854 (7.24%) Look Up Tables, 1691 (1.59%) Flip Flops and 70.5 (50.36%) Block RAM of the FPGA have been consumed.

5.2 AES-256 RESULTS

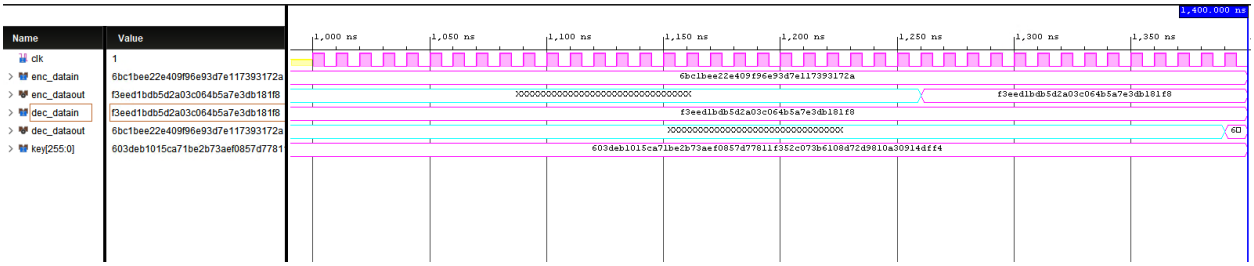


Fig. 5.8 AES-256 Waveform

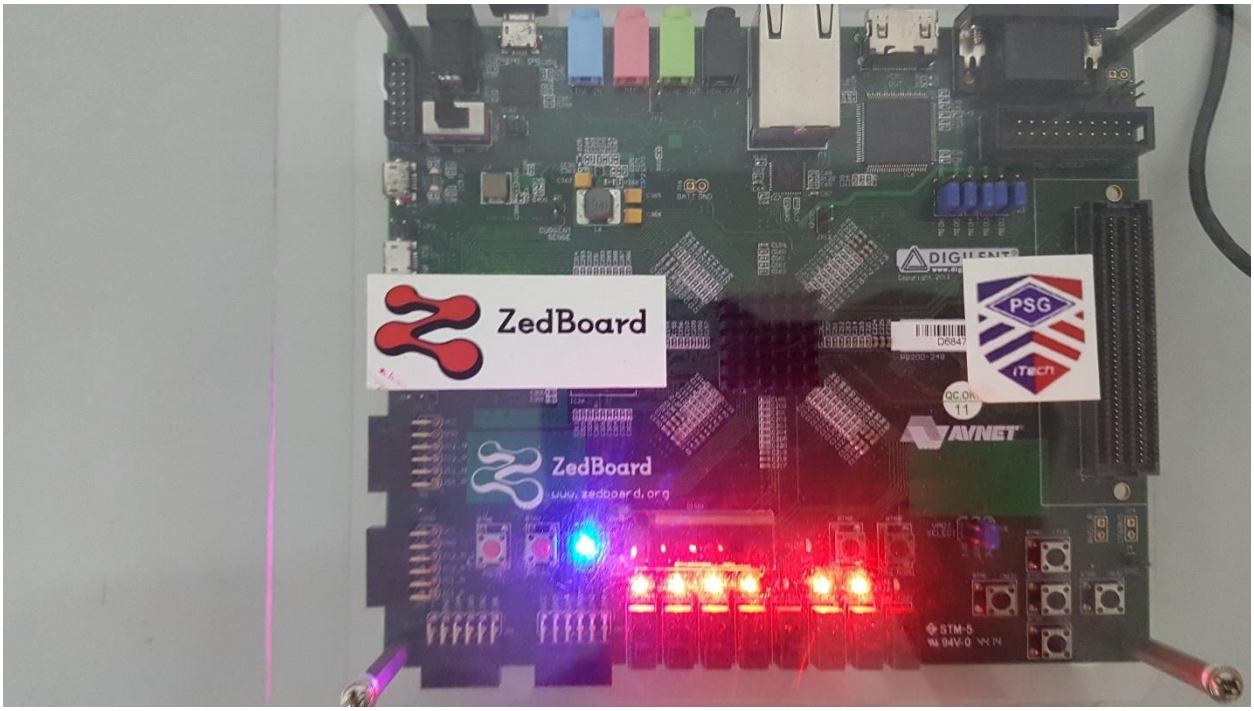


Fig. 5.9 AES-256 implemented in Hardware

The working of AES-256 design can be seen from the figures 5.8 and 5.9. The values displayed correspond to the first test case of AES-256, shown in Fig. 5.3. In hardware, the first 4 bits of the ciphertext and decrypted plaintext of the first testcase displayed using the 8 Red LEDs can be seen, i.e. 0xF6 or 11110110.

Process Name	Clock Cycles
Key Expansion	13
One Round of Encryption/Decryption	2
Full Encryption Process	27
Full Decryption Process	40

Fig. 5.10 Performance of AES-256 Design

The performance of the AES-256 design in terms of clock cycles is shown in Fig. 5.10. The key expansion stage has to generate 13 keys and each generation took exactly 1 clock cycle to output its round key. Encryption and decryption rounds take 2 clock cycles each as described for AES-128 design. The actual encryption process (excluding the key expansion phase) proceeds as soon as the first round key is generated and finishes at 27 clock cycles. The actual decryption process can only start after the last key is generated i.e. after 13 clock cycles and thus, taking 10 more clock cycles than the encryption process and finishing at 40 clock cycles.

Resource	Utilization	Available	Utilization %
LUT	4544	53200	8.54
FF	1595	106400	1.50
BRAM	70.50	140	50.36
IO	9	200	4.50
BUFG	1	32	3.13

Fig. 5.11 Resource Utilization of AES-256 Design in Hardware

The resource utilization of the design is shown in Fig. 5.11 (captured in Vivado 2019.1). One clock source is used. 4544 (8.54%) Look Up Tables, 1595 (1.50%) Flip Flops and 70.5 (50.36%) Block RAM of the FPGA have been consumed.

CHAPTER 6

CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION

As seen from the implementation results, we were successfully able to generate the performance metrics. The following details can be observed from the results.

- Decryption takes extra clock cycles than encryption (to be precise, it is the number of clock cycles taken for key expansion) because the last round key is required for the process to start.
- The number of clock cycles required scaled linearly as only 4 extra rounds were needed for AES-256 which takes 2 clock cycles per round.
- The BRAM usage stayed the same for both versions.
- Nearly 18% more LUTs and 5.6% less Flip Flops were required for the implementation of AES-256 than AES-128.

From the results of the cryptanalysis in section 4.4, it can be said that the cracking of AES-256 is computationally nearly 2^{128} times more complex than AES-128. The former offers exponentially higher encryption security and considering the improvement, the increase in the required resources seems less significant.

6.2 FUTURE SCOPE

How long a cipher will be secure cannot be said certainly, as newer attacks on existing algorithms are being researched every day. It is only possible to give a loose estimate. AES is expected to secure communications at least until 2030 without any modifications in its existing structure. It is being used to secure

communications in various applications such as banking, satellite communication, defense applications, government documents, confidential corporate documents, personal storage devices and wireless networks (Wi-Fi, Zigbee, Wibree). As technology matures, the designs can be made to execute quicker with reduced area and resource consumption.

NIST is taking initiatives to develop newer and better standards of encryption as the importance of privacy is growing day by day. As a result, researchers are developing never algorithms and are tested rigorously. A few examples of such algorithms are Anubis, Grand Cru and Kalyna. None of them have provided any significant advantage over Rijndael algorithm. So, it remains as the standard.

Newer attacks are also researched intensely. It is better to find the vulnerability and patch it before the knowledge reaches the wrong hands. So, all security algorithms have to be tested for, against attacks on all platforms regularly so that the principle of confidentiality is not breached.

REFERENCES

[1] FIPS 197, “Advanced Encryption Standard (AES)”, November 26, 2001

[2] Chodowiec P., Gaj K. (2003) Very Compact FPGA Implementation of the AES Algorithm. In: Walter C.D., Koç Ç.K., Paar C. (eds) Cryptographic Hardware and Embedded Systems - CHES 2003. CHES 2003. Lecture Notes in Computer Science, vol 2779. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-45238-6_26

[3] J. Daemen and V. Rijmen, “AES Proposal: Rijndael”, AES Algorithm Submission, September 3, 1999

[4] Wei Wang, Jie Chen, Fei Xu, “An implementation of AES algorithm Based on FPGA”, 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, May 2012

[5] Swinder Kaur, Prof. Renu Vig, “Efficient Implementation of AES Algorithm in FPGA Device”, ICCIMA, Dec 2007

[6] Hrushikesh S. Deshpande, Kailash J. Karande, Altaaf O. Mulani, “Efficient Implementation of AES Algorithm on FPGA”, International Conference on Communication and Signal Processing, April, 2014

[7] Atul M. Borkar, R. V. Kshirsagar, M. V. Vyawahare, “FPGA Implementation of AES Algorithm”, 2011 3rd International Conference on Electronics Computer Technology

[8] Behrouz A. Forouzan, Debdeep Mukhopadhyay, “Cryptography and Network Security (2nd Edition)”, McGraw-Hill

[9] C K Shyamala, Dr T R Padmanabhan, N Harini, “Cryptography and Security”, Wiley India

[10] William Stallings, “Cryptography and Network Security (6th Edition)”, Pearson

[11] Shuang Chen, Wei Hu, Zhenhao Li, “High Performance Data Encryption with AES Implementation on FPGA”, 2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)

[12] Christoforus Juan Benvenuto, “Galois Field in Cryptography”, 2012

[13] https://www.researchgate.net/

[14] https://www.commonlounge.com/discussion/6747358d828a45c99f61f4c09ff2f371

[15] https://pequalsnp-team.github.io/writeups/Bad-Aes

[16] https://sectigostore.com/blog/types-of-encryption-what-to-know-about-symmetric-vs-asymmetric-encryption/

[17] https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[18] https://www.tutorialspoint.com/cryptography/advanced_encryption_standard.htm