International Journal of Electrical and Electronics Research ISSN 2348-6988 (online)
Vol. 3, Issue 1, pp: (133-148), Month: January - March 2015, Available at: www.researchpublish.com

Design and Implementation of AES Algorithm for Complex Encryption and Decryption

CH.SALA ANKARAJU[1], M.A.VIJAY KAMALNATH[2]

[1] P.G Student/VLSI Design, AVR & SVR College of Technology, Nandyal, Andhra Pradesh, India
[2]M.TECH, Associate Professor, Dept. of ECE, AVR & SVR College of Technology, Nandyal, Andhra Pradesh, India

Abstract: **In this paper we present a high-performance, high throughput, and area efficient architecture for AES algorithm. The sub keys, required for each round of the Rijndael algorithm, are generated in real-time by the key scheduler module by expanding the initial secret key, thus reducing the amount of storage for buffering. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. AES has a fixed block size of 128 bits and a key size of 128, 192 or 256 bits, whereas Rijndael can be specified with key and block sizes in any multiple of 32 bits, with a minimum of 128 bits and a maximum of 256 bits. AES operates on a 4×4 array of bytes, termed the state. For encryption, each round of AES (except the last round) consists of four stages. a) Sub Bytes - a non-linear substitution step where each byte is replaced with another according to a lookup table (known as S Box). b) Shift Rows - a transposition step where each row of the state is shifted cyclically a certain number of steps. c) Mix Columns - a mixing operation which operates on the columns of the state, combining the four bytes in each column using a linear transformation. d) Add Round Key - each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule.**

Keywords: **AES algorithm.**

1. INTRODUCTION

1.1 Fundamentals Of Cryptography:

Cryptography is the exploration of utilizing math to scramble and unscramble information. Cryptography empowers you to store delicate data or transmit it crosswise over unstable systems (like the Internet) with the goal that it can't be perused by anybody aside from the expected beneficiary. While cryptography is the exploration of securing information, cryptanalysis is the science of investigating and breaking secure correspondence. Established cryptanalysis includes an intriguing blend of explanatory thinking, application of numerical devices, design discovering, persistence, determination, and good fortune. Cryptanalysts are likewise called aggressors. Cryptology grasps both cryptography and cryptanalysis.

1.1.1 Symmetrical Cryptography:

In Symmetrical cryptography, likewise called mystery key or customary key encryption, one key is utilized both for encryption and unscrambling. The Data Encryption Standard (DES) is an illustration of a traditional cryptosystem that is broadly utilized by the Federal Government. Figure 1-1 is a representation of the customary encryption process.

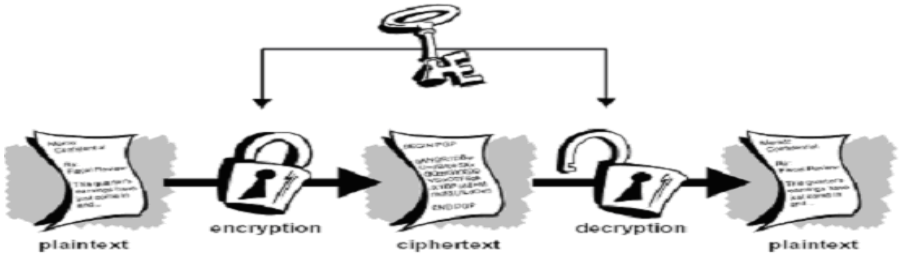


Fig.1.1 Symmetric Key cryptography

Traditional encryption has profits. It is quick. It is particularly valuable for scrambling information that is not going anyplace. Be that as it may, traditional encryption alone as methods for transmitting secure information can be truly costly just because of the trouble of secure key dissemination. Review a character from your most loved spy motion picture: the individual with a bolted attaché cuffed to his or her wrist. What is in the portfolio, at any rate? It's presumably not the rocket dispatch code/biotoxin equation/intrusion plan itself. It's the key that will decode the mystery information. For a sender and beneficiary to impart safely utilizing routine encryption, they must concur upon a key and keep it mystery between themselves. On the off chance that they are in distinctive physical areas, they must trust a dispatch, the Bat Phone, or some other secure correspondence medium to keep the divulgence of the mystery key amid transmission. Any individual who catches or blocks the key in travel can later read, change, and fashion all data scrambled or validated with that key.

The issues of key appropriation are comprehended by open key cryptography, the idea of which was presented by Whitfield Diffie and Martin Hellman in 1975.

1.1.2 Asymmetric or Public Key cryptography:

Open key cryptography is an unbalanced plan that uses a couple of keys for encryption:

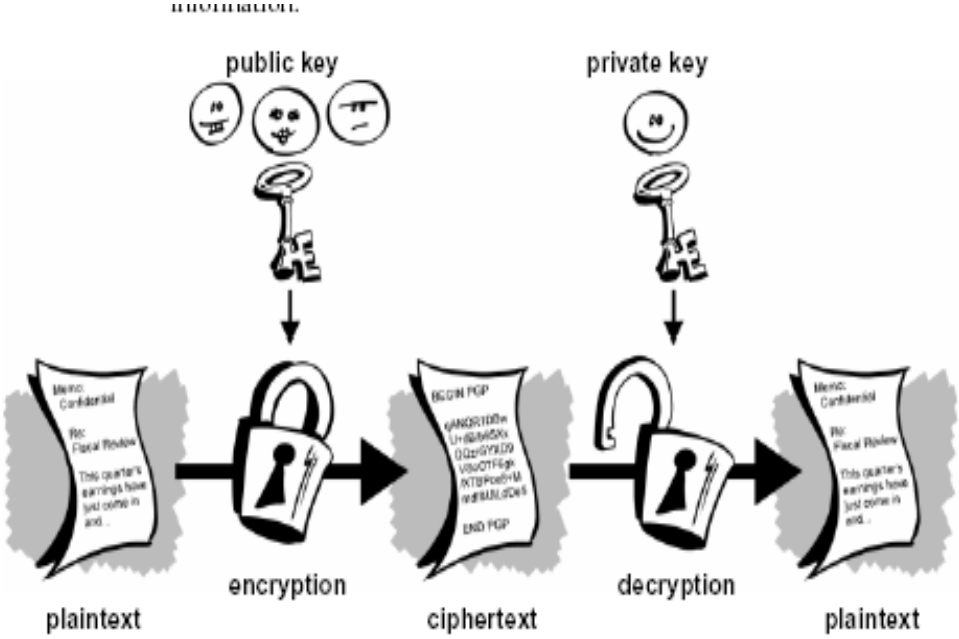


Fig.1.2. Public key cryptography

As demonstrated in Figure 1-2 open key, which encodes information, and a comparing private, or mystery key for unscrambling. You distribute your open key to the world while keeping your private key mystery. Anybody with a duplicate of your open key can then encode data that no one but you can read. Indeed individuals you have never met. It is computationally infeasible to conclude the private key from general society key. Any individual who has an open key can scramble data yet can't decode it. Just the individual who has the relating private key can decode the data.

2. FUNDAMENTALS OF MULTIPROTOCO LABEL SWITCHING (MPLS) HEADER

2.2.1. Multi-Protocol Label Switching (MPLS) Header:

In machine systems administration and information transfers, Multi-convention Label Switching (MPLS) is information convey component, which imitates a few properties of a circuit-exchanged system over a parcel exchanged system. MPLS works at an OSI Model layer that is for the most part considered to lie between customary meanings of Layer 2 (information connection layer) and Layer 3 (system layer), and hence is frequently alluded to as a "Layer 2.5" convention. It was intended to give a bound together information convey administration for both circuit-based customers and bundle exchanging customers, which give a datagram administration model. It can be utilized to convey numerous various types of movement, including IP parcels, and in addition local ATM, SONET, and Ethernet outlines. Fig 1.3 demonstrates the MPLS Header in ISP system model.

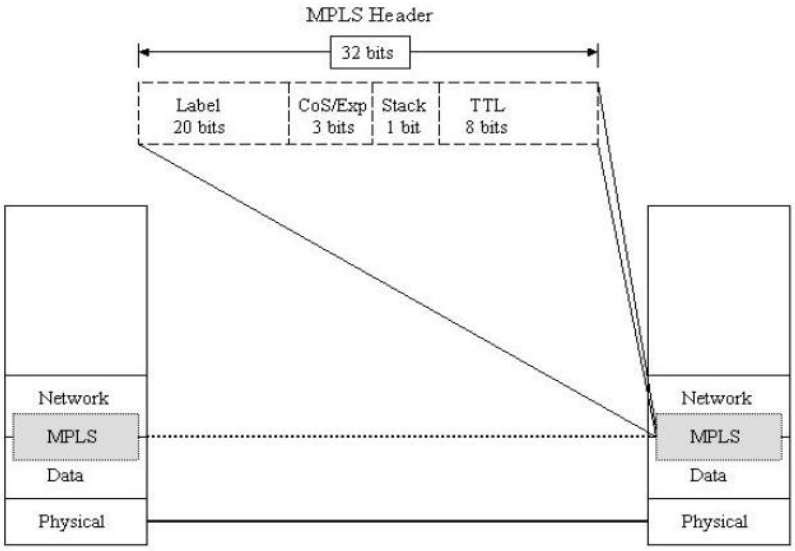


Fig.1.3 MPLS Header in ISP Network

Various distinctive innovations were formerly sent with basically indistinguishable objectives, for example, edge hand-off and ATM. MPLS is currently supplanting these advances in the commercial center, basically on the grounds that it is better adjusted to present and future engineering and needs.

Specifically, MPLS sheds the phone exchanging and flagging convention stuff of ATM. MPLS perceives that little ATM cells are not required in the center of cutting edge systems, since advanced optical systems (starting 2001) are so quick (at 10 gbit/s and well past) that even full-length 1500 byte parcels don't acquire huge constant lining defers (the need to decrease such postpones, to help voice activity, having been the inspiration for the cell nature of ATM). In the meantime, it endeavors to safeguard the activity building and out-of-band control that made casing transfer and ATM alluring for conveying substantial scale systems.

MPLS was initially proposed by a gathering of specialists from Cisco frameworks, inc.; it was called "label exchanging" when it was a Cisco restrictive proposal, and was renamed "name exchanging" when it was given over to the IETF for open institutionalization.

One unique inspiration was to permit the formation of straightforward high velocity switches, since it was at one point thought to be difficult to forward IP bundles completely in fittings. Nonetheless, propels in VLSI have made such gadgets conceivable. The systemic points of interest of MPLS, for example, the capacity to help various administration models, do movement administration, and so on, remain.

2.2.2 Structure of MPLS Header:

MPLS works by pre-pending packets with an MPLS header, containing one or more 'labels'. This is called a label stack.

Each label stack entry contains four fields:

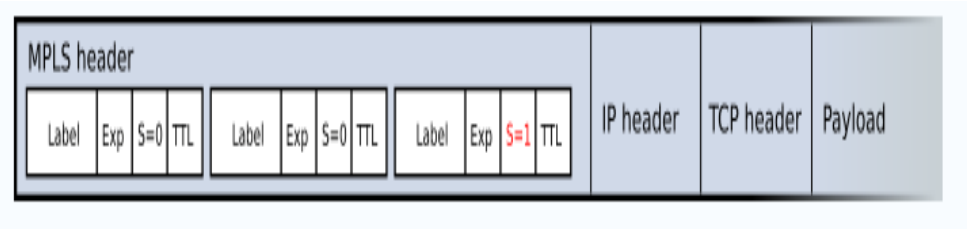


Fig.1.4 MPLS HeaderA 20-bit label value

A 3-bit experimental field reserved for future use.

A 1-bit bottom of stack flag. If this is set, it signifies the current label is the last in the stack.

An 8-bit TTL (time to live) field.

2.2.3 Functioning of MPLS Header:

The MPLS named parcels are sent (exchanged is the right term) after a mark lookup/switch rather than a lookup into the IP table. Name lookup and mark exchanging may be quicker than regular rib lookup in light of the fact that it can happen straightforwardly into fabric and not CPU.

The passageway purposes of a MPLS system are called mark edge switches (LER). Switches that are performing steering built just in light of name exchanging are called mark switch switches (LSR). Keep in mind that a LER is not typically the particular case that is popping the name. For more data see penultimate jump popping.

Gadgets that capacity as entrance and/or departure switches are regularly called PE (supplier edge) switches. Gadgets that capacity just as travel switches are also called p (supplier) switches. The occupation of a p switch is essentially simpler than that of a PE switch, so they can be less intricate and may be more reliable in light of this.

At the point when an unlabeled parcel enters the entrance switch and needs to be passed on to a MPLS burrow, the switch first decides the sending proportionality class the bundle ought to be in, and afterward embeds one (or more) names in the bundle's recently made MPLS header. The bundle is then passed on to the following bounce switch for this passage.

At the point when a MPLS switch gets a marked parcel, the highest name is analyzed. In light of the substance of the name a swap, push or pop operation can be performed on the parcel's mark stack. Switches can have prebuilt lookup tables that let them know which sort of operation to do focused around the highest name of the approaching bundle so they can prepare the parcel rapidly. In a swap operation the name is swapped with another mark, and the bundle is sent along the way connected with the new mark.

In a push operation another mark is pushed on top of the current name, successfully "typifying" the bundle in an alternate layer of MPLS. This permits the progressive steering of MPLS bundles. Prominently, this is utilized by MPLS Vpns.

In a pop operation the name is expelled from the bundle, which may uncover an inward mark underneath. This methodology is called "decapsulation". In the event that the popped mark was the keep going on the name stack, the parcel "leaves" the MPLS burrow. The departure switch typically does this, however see PHP underneath.

Amid these operations, the substance of the bundle underneath the MPLS mark stack are not inspected. For sure travel switches commonly require just to inspect the highest mark on the stack. The sending of the parcel is carried out focused around the substance of the names, which permits "convention free bundle sending" that does not have to take a gander at a convention ward directing table and evades the lavish IP longest prefix match at each one bounce.

At the departure switch, when the last mark has been popped, just the payload remains. This can be an ip parcel, or any of various different sorts of payload bundle. The departure switch should along these lines have directing data for the bundle's payload, since it must forward it without the assistance of mark lookup tables. A MPLS travel switch has no such necessity.

In some exceptional cases, the last mark can likewise be popped off at the penultimate bounce (the jump before the departure switch). This is called penultimate bounce popping (PHP). This may be intriguing in situations where the departure switch has bunches of parcels leaving MPLS shafts, and consequently invests over the top measures of CPU time on this. By utilizing PHP, travel switches associated specifically to this departure switch viably offload it, by popping the last name themselves. Since the departure switch will do a higher-layer steering table lookup in any case, the measure of higher-layer work required for an awhile ago popped bundle continues as before, and the real name popping need not be carried out.

MPLS can make utilization of existing ATM system foundation, as its marked streams can be mapped to ATM virtual circuit identifiers, and the other way around.

2.4. Objective of the Project:

The objectives for this undertaking have been the accompanying.

To plan a cryptographic convention to ensure the Multi-convention Label Switching (MPLS) header utilized within an Internet Service Provider (ISP) system. This convention ought to secure the MPLS header basically against altering for purposes of commandeering ISP assets. Auxiliary objectives are insurance against replay assault and activity examination of ISP movement. The convention ought to be quick to minimize postponement brought into the fast MPLS switches.

One objective has been to aggregate a prologue to the subject of cryptography. There exist various investigations of different parts of the cryptographic gauges, yet finish medications on a specialized level are not as normal. Material from papers, diaries, and meeting processes are utilized that best depict the different parts.

An alternate objective has been to hunt down calculations that can be utilized to actualize the suitable cryptography for MPLS name exchanging.

A third objective is to assess their execution of different cryptographic conventions and to choose a best convention that can be actualized best for MPLS exchanging. These properties were picked in light of the fact that they have the best effect on the usage exertion.

A last objective has been to plan and reenact a cryptographic convention. This ought to be carried out in C or MATLAB. The source code ought to be straightforward with the goal that it can serve as a kind of perspective on the standard for architects that need to actualize a framework.

3. IMPLEMENTATION OF THE PROPOSED SYSTEM

3.1: Introduction:

The AES algorithm is a round-based symmetric block cipher that processes data block of 128 bits using a cipher key of 128, 192, or 256 bits. A sequence of four primitive functions, SubByte, ShifRow, MixColumn and AddRoundKey execute Nr-1 times forms a loop called a round. The number of iteration loop Nr can be 10, 12, or 14 depending on the size of key. SubByte operation is a nonlinear byte substitution that operates independently on each byte of the state using a substitution table (Sbox). ShiftRow operation is a circular shifting on the rows of the state with different numbers of bytes (offsets). MixColumn operation mixes the bytes in each column by the multiplication of the state with a fixed polynomial modulo x4 + 1. AddRoundKey operation is an XOR process that adds a round key to the state in each iteration [1].

Fig. 2 summarizes the AES algorithm in two flow diagrams, MixColumn is not performed at the last round, the sequence of SubByte and ShiftRow can be switched without affecting the final cipher output.

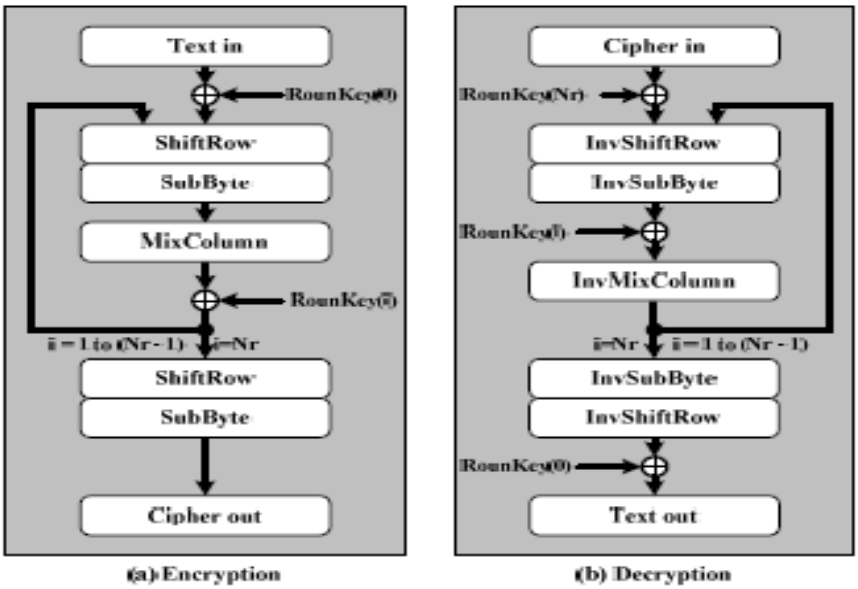


Fig.2 128-b AES Encryption/Decryption flow diagram

Rijndael is a block cipher developed by Joan Daemen and Vincent Rijmen. The algorithm is flexible in supporting any combination of data and key size of 128, 192, and 256 bits. However, AES simply permits a 128 bit information length that can be isolated into four essential operation pieces. These squares work on exhibit of bytes and composed as a 4×4 lattice that is known as the state. For full encryption, the information is passed through Nr rounds (Nr = 10, 12, 14) [4, 6]. These rounds are represented by the accompanying changes:

i. bytesub change: Is a non straight byte Substitution, utilizing a substation table (s-box), which is developed by multiplicative opposite and relative change. The Fig.1 demonstrates the venture of the Bytesub change.

ii. shiftrows change: Is a basic byte transposition, the bytes in the last three columns of the state are cyclically moved; the counterbalance of the left move fluctuates from one to three bytes.

iii. mixcolumns change: Is proportionate to a lattice duplication of sections of the states. Every section vector is increased by a settled network. It ought to be noted that the bytes are dealt with as polynomials instead of numbers.

iv. Addroundkey change: Is a basic XOR between the working state and the round key. This change is its own particular converse.

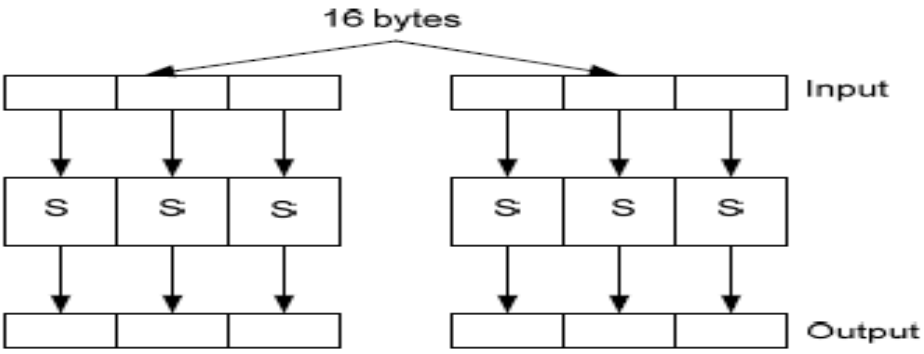


Figure: block diagrams for subsitition

The encryption system comprises of a few steps as demonstrated by Fig. 2. After a beginning addroundkey, a round capacity is connected to the information piece (comprising of byte sub, shift rows, mix columns and addroundkey change, individually). It is performed iteratively (Nr times) contingent upon the key length. The unscrambling structure has precisely the same arrangement of changes as the one in the encryption structure. The changes Inv-Bytesub, the Inv-Shiftrows, the Inv-Mixcolumns, and the Addroundkey permit the type of the key timetables to be indistinguishable for encryption and unscrambling.

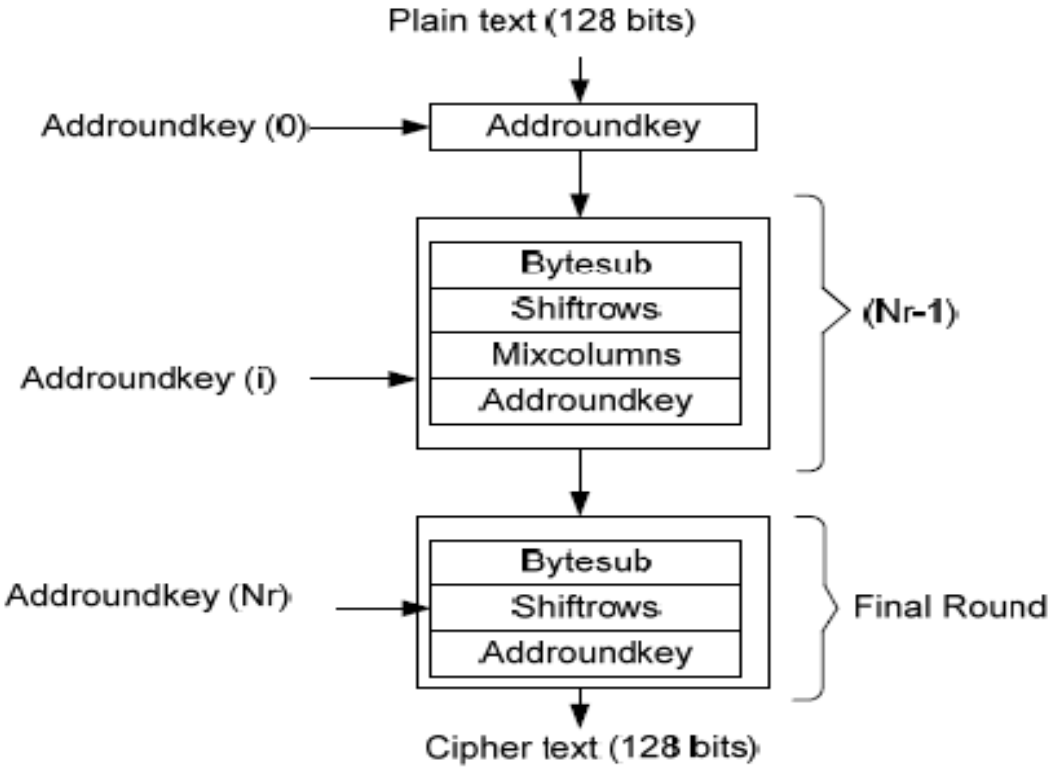
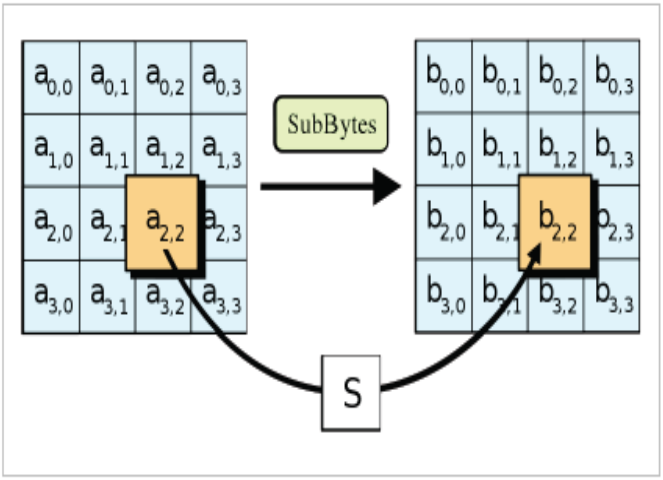


Figure:. AES algorithm Encryption structure

3.2 Rijndael Advanced Encryption Standard:



The SubBytes step, one of four stages in a round of AES

In cryptography, the Advanced Encryption Standard (AES) is a symmetric-key encryption standard received by the U.s. government. The standard includes three square figures, AES-128, AES-192 and AES-256, embraced from a bigger gathering initially distributed as Rijndael. Each of these figures has a 128-bit piece size, with key sizes of 128, 192 and 256 bits, individually. The AES figures have been dissected broadly and are presently utilized around the world, as was the situation with its forerunner, the Data Encryption Standard (DES).

AES was published by National Institute of Standards and Technology (NIST) as U.s. FIPS PUB 197 (FIPS 197) on November 26, 2001 after a 5-year institutionalization handle in which fifteen contending outlines were introduced and assessed before Rijndael was chosen as the most suitable (see Advanced Encryption Standard methodology for more points of interest). It got to be compelling as a Federal government standard on May 26, 2002 after regard by the Secretary of Trade. It is accessible in numerous distinctive encryption bundles. AES is the first openly open and open figure affirmed by the NSA for top mystery data (see Security of AES, underneath).

The Rijndael figure was created by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, and put together by them to the AES determination process. Rijndael (proclaimed "Rhine dall") is a wit with the names of the two creators.

3.3: Description of the cipher:

AES is focused around an outline standard known as a Substitution change system. It is quick in both programming and hardware.[6] Unlike its antecedent, DES, AES does not utilize a Feistel system.

AES has an altered square size of 128 bits and a key size of 128, 192, or 256 bits, while Rijndael can be determined with piece and key sizes in any different of 32 bits, with at least 128 bits. The blocksize has a most extreme of 256 bits, yet the keysize has no hypothetical greatest.

AES works on a 4×4 exhibit of bytes, termed the state (forms of Rijndael with a bigger square size have extra sections in the state). Most AES estimations are carried out in an unique limited field.

The AES figure is indicated as various redundancies of change adjusts that change over the data plaintext into the last yield of ciphertext. Each round comprises of a few transforming steps, including one that relies on upon the encryption key. A set of converse rounds are connected to change ciphertext go into the first plaintext utilizing the same encryption key.

3.4: High-level description of the algorithm:

1. Key expansion—round keys are determined from the figure key utilizing Rijndael's key calendar

2. Beginning Round

1. Addroundkey—every byte of the state is joined with the round key utilizing bitwise or

3. Rounds

1. Sub bytes—a non-straight substitution step where every byte is supplanted with an alternate as per a lookup table.

2. Shift rows—a transposition step where each one line of the state is moved cyclically a specific number of steps.

3. Mix columns—a blending operation which works on the sections of the state, joining the four bytes in every section.

4. Add Round Key

4. Final Round (no Mix Columns)

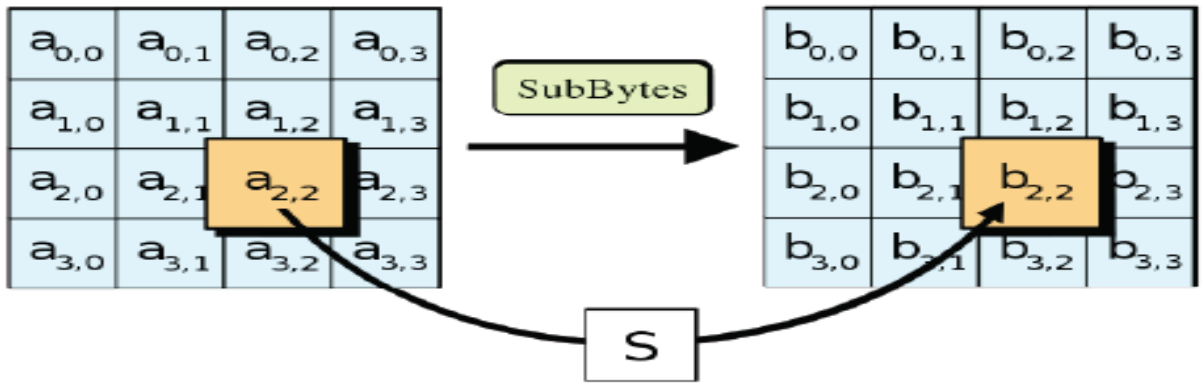
1. Sub Bytes

2. Shift Rows

3. Add Round Key

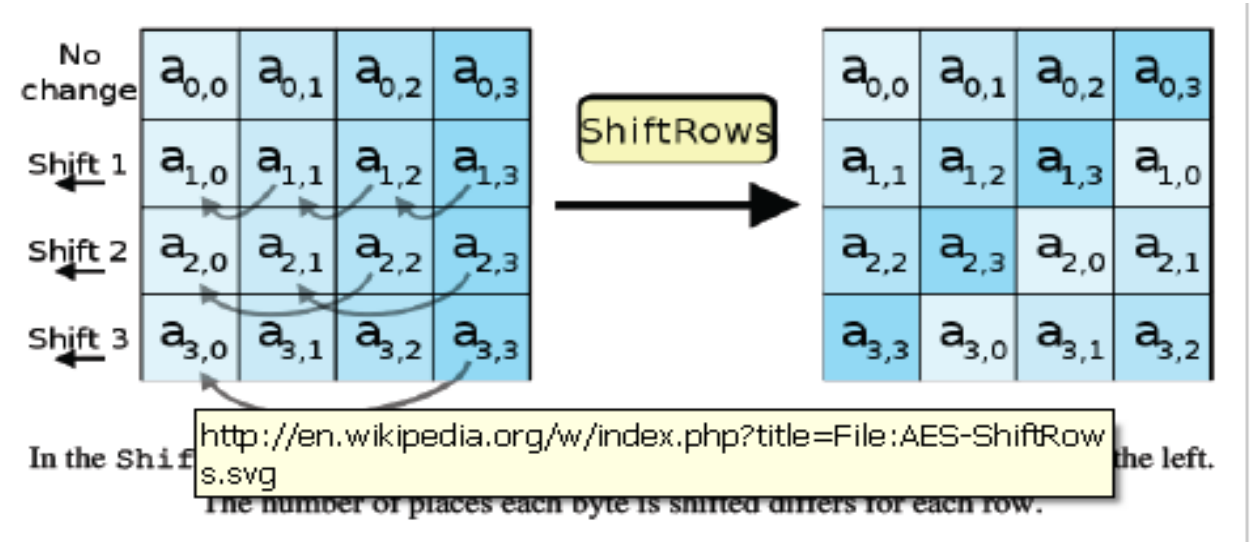
3.5 The Sub Bytes step:

In the Sub bytes step, every byte in the exhibit is upgraded utilizing a 8-bit substitution box, the Rijndael S-box. This operation gives the non-linearity in the figure. The S-box utilized is inferred from the multiplicative reverse over Gf(28), known to have great non-linearity properties. To dodge assaults focused around basic logarithmic properties, the S-box is developed by consolidating the backwards work with an invertible relative change. The S-box is additionally decided to keep away from any settled focuses (along these lines is an unsettling), furthermore any inverse altered focuses.



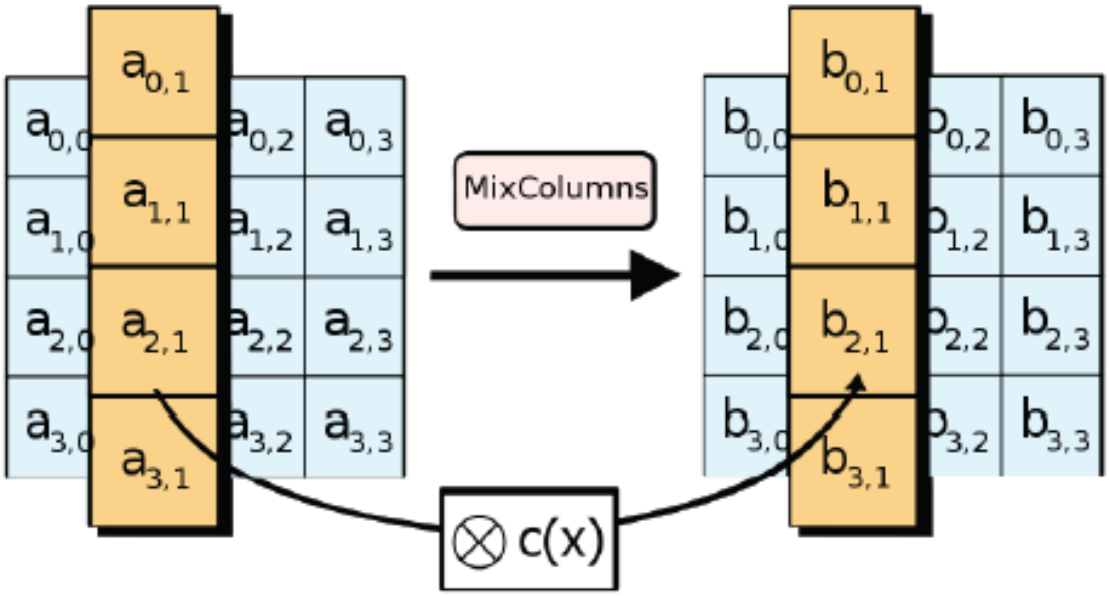
In the SubBytes step, each byte in the state is replaced with its entry in a fixed 8-bit lookup table, S; $b_{ij} = S(a_{ij})$.

3.6 The Shift Rows step:



In the Shif the left.
The number of places each byte is shifted differs for each row.

The Shift rows step works on the lines of the state; it cyclically moves the bytes in each one line by a certain counterbalance. For AES, the first column is left unaltered. Every byte of the second column is moved one to the left. Additionally, the third and fourth columns are moved by counterbalances of two and three individually. For the piece of size 128 bits and 192 bits the moving example is the same. Along these lines, every section of the yield state of the Shift rows step is made out of bytes from every segment of the information state. (Rijndael variations with a bigger piece size have marginally distinctive counterbalances). On account of the 256-bit hinder, the first column is unaltered and the moving for second, third and fourth column is 1 byte, 3 bytes and 4 bytes separately - this change applies for the Rijndael figure when utilized with a 256-bit hinder, as AES does not utilize 256-bit squares.3.6 **The Mix Columns step**



In the `MixColumns` step, each column of the state is multiplied with a fixed polynomial *c(x)*.

In the Mix columns step, the four bytes of every section of the state are consolidated utilizing an invertible straight change. The Mix columns capacity takes four bytes as data and yields four bytes, where each one information byte influences every one of the four yield bytes. Together with Shift rows, Mix columns give dissemination in the figure.

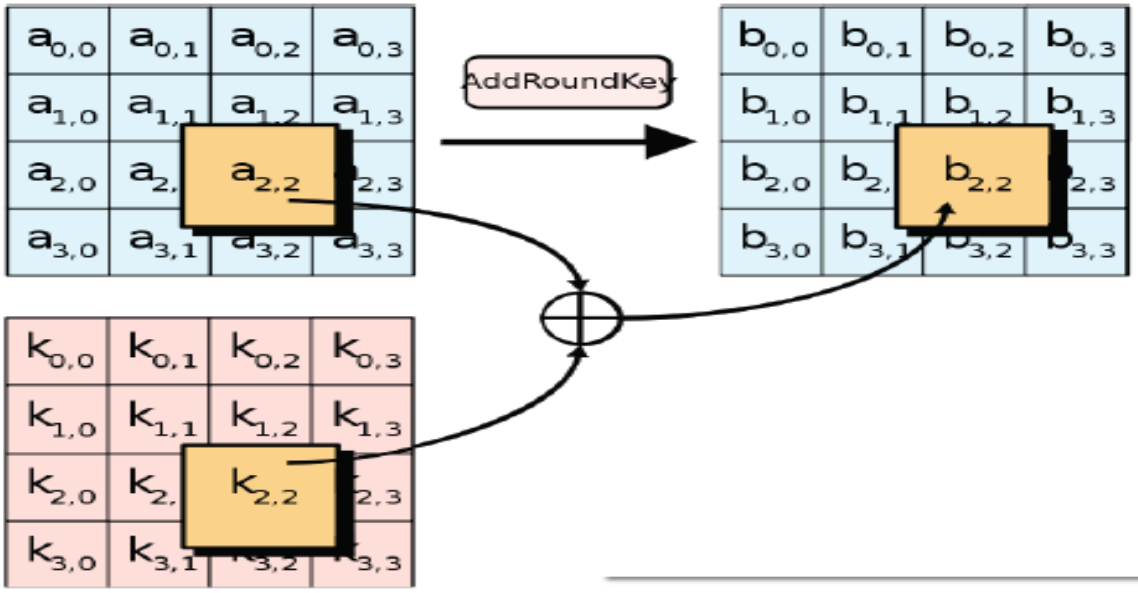
Amid this operation, every section is reproduced by the known network that for the 128 bit key is

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

The increase operation is characterized as: augmentation by 1 methods leaving unaltered, duplication by 2 methods moving byte to the left and increase by 3 methods moving to the left and afterward performing xor with the introductory upshifted quality. In the wake of moving, a restrictive xor with 0x1b ought to be performed if the moved worth is bigger than 0xff.

In more general sense, every segment is dealt with as a polynomial over Gf(28) and is then duplicated modulo x4+1 with an altered polynomial c(x) = 0x03 • x3 + x2 + x + 0x02. The coefficients are shown in their hexadecimal likeness the paired representation of bit polynomials from Gf(2)[x]. The Mix columns step can likewise be seen as an increase by a specific MDS lattice in a limited field. This methodology is depicted further in the article Rijndael blend segment.

3.7 The AddRoundKey step:



In the AddRoundKey step, each byte of the state is combined with a byte of the round subkey using the XOR operation (⊕).

In the Addroundkey step, the sub key is joined with the state. For each round, a sub key is determined from the fundamental key utilizing Rijndael's key calendar; every sub key is the same size as the state. The sub key is included by consolidating every byte of the state with the relating byte of the sub key utilizing bitwise XOR.

3.8 Optimization of the cipher:

On frameworks with 32-bit or bigger words, it is conceivable to accelerate execution of this figure by consolidating Sub bytes and Shift rows with Mix columns, and changing them into a succession of table lookups. This obliges four 256-section 32-bit tables, which uses an aggregate of four kilobytes (4096 bytes) of memory—one kilobyte for each one table. A round can now be finished with 16 table lookups and 12 32-bit restrictive or operations, emulated by four 32-bit select or operations in the Addroundkey step 17.

In the event that the ensuing four kilobyte table size is excessively extensive for a given target latform, the table lookup operation can be performed with a solitary 256-passage 32-bit (i.e. 1 kilobyte) table by the utilization of roundabout turns. Utilizing a byte-arranged methodology, it is conceivable to join the Sub bytes, Shift rows, and Mix columns steps into a solitary round operation.

3.9 Shift Row/InvShiftRow realization:

Fig. 4 shows the actual circuit realization of Fig. 3. Three 32-bit Shift registers (a2, a1, a0), four 32-bit rotate registers (b3, b2, b1, b0) which contain 16 bytes briefed by (0, 1, 2, 3, …, 14, 15), and two 8-bit 2×1 multiplexers are organized to perform ShiftRow/InvShiftRow operation. First, the (b3, b2, b1, b0) is shifting 'tin' in 4 consecutive clocks to form an 128- bit input. Then the first 32-bit ShiftRow output is obtained in the 32-bit output 'T' (t0, t1, t2, t3) and ready to proceed the operation of SubByte&MixColumn. The output of SubByte&MixColumn MX/MX' is looped back to 'ak'. An 128-bit new state is then ready after next 3 consecutive clocks and is shifted in parallel to (b3, b2, b1, b0) when counter e = 0 for initiating the operation of next round.

The dashed box in Fig. 4 shows the detailed ShiftRow processing. ShiftRow is realized by left rotating registers (b0, b1, b2, b3) after they are loaded parallely a new state value from input (a0, a1, a2, ak) shown in Fig. 4. In the dashed box, above (b0, b1, b2, b3), shows the 3 consecutive rotates to accomplish a complete 128-bit ShiftRow or InvShiftRow.

For encryption (enc=1), the ShiftRow outputs (t0, t1, t2, t3) are connected from (0, 5, 10, 15). The next 3 consecutive left rotates obtain three values, namely, (4, 9, 14, 3), (8, 13, 2, 7), and (12, 1, 6, 11) in three rows on top of (b0, b1, b2, b3), as those small circles shown in Fig. 4, which are the same as the 4 columns after ShiftRow showing at Table 2 -b. For decryption (enc=0), it can be observed that the 4 columns in table2-c are the same as the numbers with gray-dotted in the four rows above (b0, b1, b2, b3).

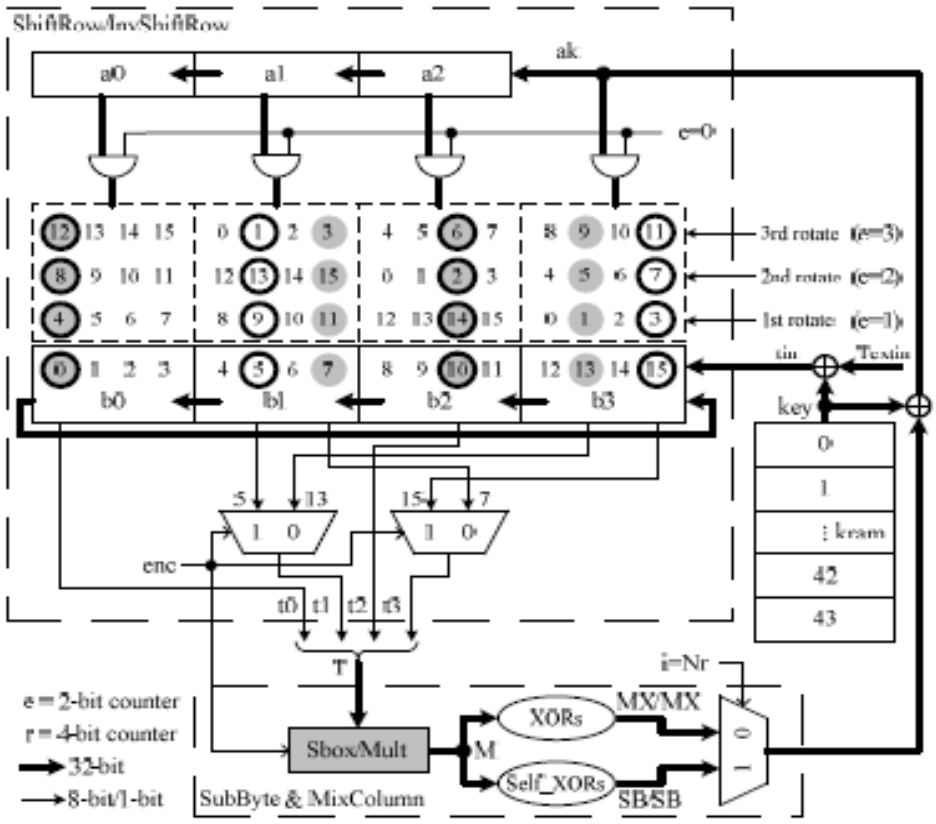


Figure 4.Circuit realization of 32-bit AES

Another approach, using by [7] and [8], which configures two 16×8 emories with shifting ability. To read from output by sending addresses (0, 5, 10, 15), (4, 9, 14, 3), (8, 13, 2, 7), (12, 1, 6, 11), starts doing SubByte, MixColumn, and AddRoundKey, and then shifts the result to input memory consecutively from 0 to 15. Output memory now becomes input for the next round. This approach needs address generating schemes to send proper addresses to output memory. For example, an 4-bit countup-5 counter for addressing ShiftRow, while countdown-3 counter for addressing InvShiftRow. To avoid the address control and address decoding, previous 2 shift registers are used for our approach.

TABLE II RESULT OF SHIFTROW/INVSHIFTROW

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

(a) original input

0	4	8	12
5	9	13	1
10	14	2	6
15	3	7	11

(b) after ShiftRow

0	4	8	12
13	1	5	9
10	14	2	6
7	11	15	3

(c) after InvShiftRow

3.10: *SubByte and MixColumn realization:*

SubByte and Mixcolumn are organized by Sbox/Mult lookup table followed by a combinations XORs circuit.

1) Sbox/Mult lookup table:

The organization of Sbox/Multiplication lookup table may affect XORs circuit. In Fig. 3, signal T(t0, t1, t2, t3) is the state input, signal M(m0, m1, m2, m3) represents the SubByte output from 'T', according to the definition of MixColumn/InvMixColumn, the output MX(mx0, mx1, mx2, mx3) or MX'(mx0', mx1', mx2', mx3') is expressed as follows [1]:

$$\begin{bmatrix} mx0 \\ mx1 \\ mx2 \\ mx3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} Sub(t0) \\ Sub(t1) \\ Sub(t2) \\ Sub(t3) \end{bmatrix} = \begin{bmatrix} 2 \cdot Sub(t0) + 3 \cdot Sub(t1) + 1 \cdot Sub(t2) + 1 \cdot Sub(t3) \\ 1 \cdot Sub(t0) + 2 \cdot Sub(t1) + 3 \cdot Sub(t2) + 1 \cdot Sub(t3) \\ 1 \cdot Sub(t0) + 1 \cdot Sub(t1) + 2 \cdot Sub(t2) + 3 \cdot Sub(t3) \\ 3 \cdot Sub(t0) + 1 \cdot Sub(t1) + 1 \cdot Sub(t2) + 2 \cdot Sub(t3) \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} mx0' \\ mx1' \\ mx2' \\ mx3' \end{bmatrix} = \begin{bmatrix} e & b & d & 9 \\ 9 & e & b & d \\ d & 9 & e & b \\ b & d & 9 & e \end{bmatrix} \begin{bmatrix} Sub(t0)' \\ Sub(t1)' \\ Sub(t2)' \\ Sub(t3)' \end{bmatrix} = \begin{bmatrix} e \cdot Sub(t0)' + b \cdot Sub(t1)' + d \cdot Sub(t2)' + 9 \cdot Sub(t3)' \\ 9 \cdot Sub(t0)' + e \cdot Sub(t1)' + b \cdot Sub(t2)' + d \cdot Sub(t3)' \\ d \cdot Sub(t0)' + 9 \cdot Sub(t1)' + e \cdot Sub(t2)' + b \cdot Sub(t3)' \\ b \cdot Sub(t0)' + d \cdot Sub(t1)' + 9 \cdot Sub(t2)' + e \cdot Sub(t3)' \end{bmatrix} \quad (2)$$

The (512×32) lookup table Sbox/Mult which stores the 4 SubByte multiplication results of (2, 3, 1, 1) and (e, b, d, 9) is built according to the following format:

2 • Sub(tn)	3 • Sub(tn)	1 • Sub(tn)	1 • Sub(tn)

(3)

e • Sub(tn)'	b • Sub(tn)'	d • Sub(tn)'	9 • Sub(tn)'

(4)

If the lower 256 words of Sbox/Mult is used for storing MixColumn multiplications and the higher 256 words for InvMixColumn, then it needs 9-bit address, denoted by 'TN' which concatenates 'enc' and 'tn' as shown in (5). (enc=1 for encryption, enc=0 for decryption)

$$T_N = not(enc) \quad \& \quad tn \qquad (5)$$

2) XORs circuit.

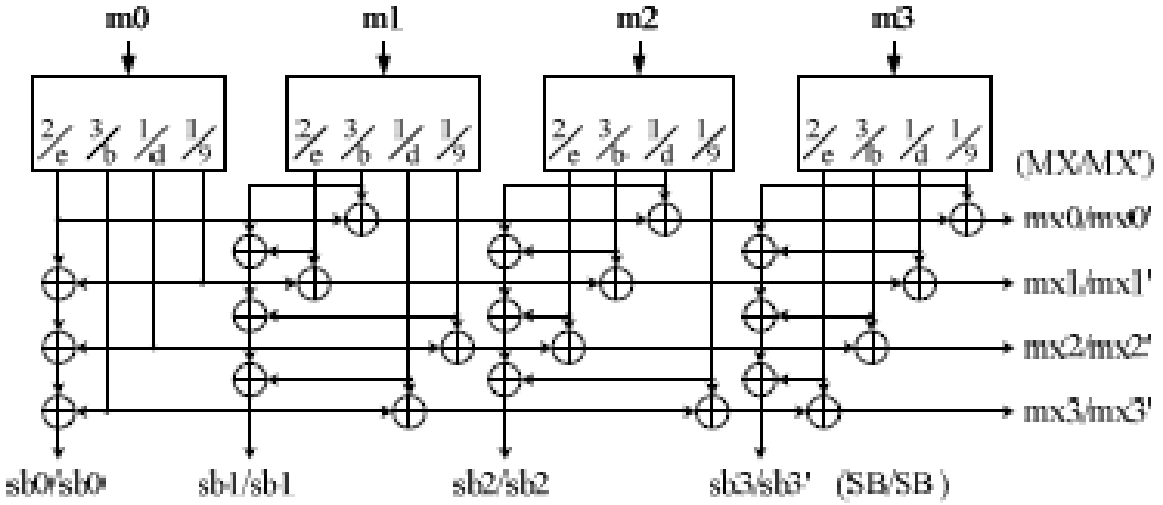


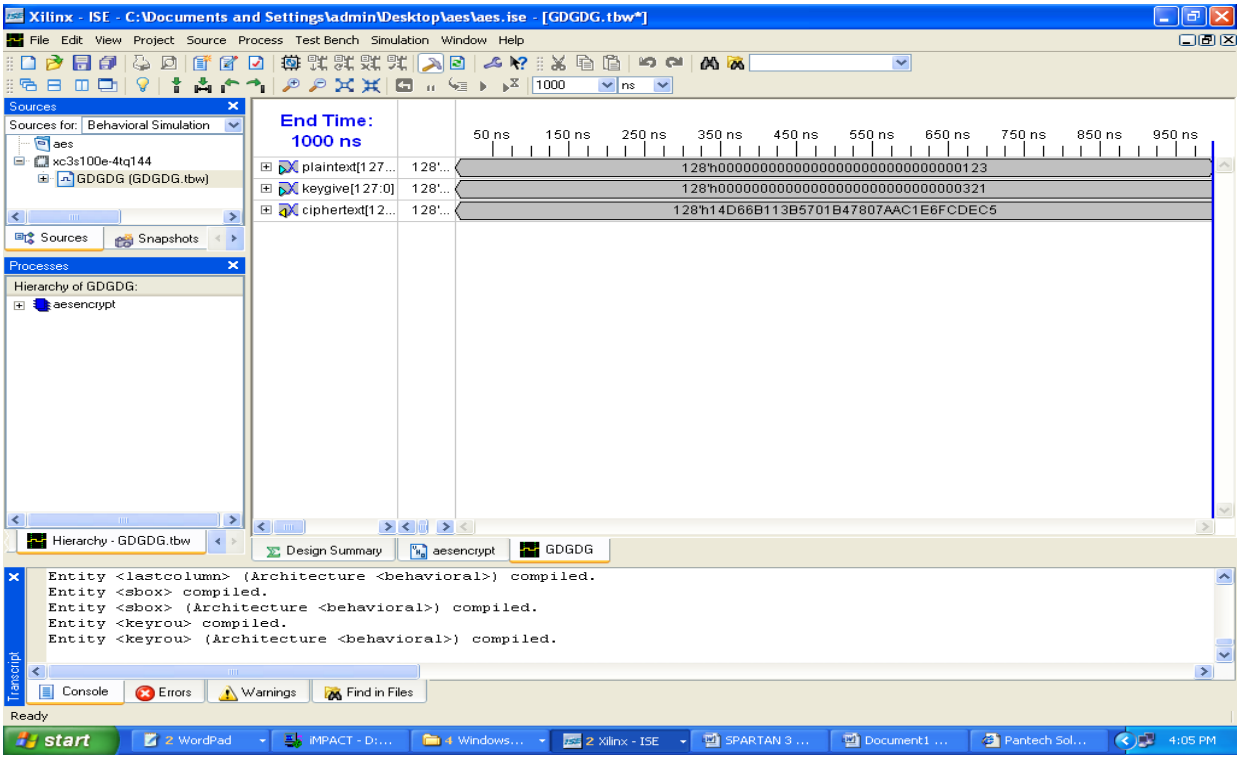
Fig. XORs and self_ XOR based on format (3), (4)

According to the memory word format (3), (4) and expression (1), (2), there are 3 additions (XORs) are needed to obtain each element of MX/MX' as shown in Fig. 5.

2) Self_ XORs:

Form Fig. 2, the last round is not needed for MixColumn or InvMixColumn, SubByte can be taken form Sbox/Mult table directly from either one of the last two values which equal to the multiplication by 1, However, the table for InvSubByte are multiplied by (e, b, d, 9). Fortunately, e+b+ d+9=1 exists, so including another 3 additions for (e, b, d, 9), called Self_XORs, can obtain aInvSubByte value. Since 2+3 +1+1=1 also exists, Self_XORs circuit can be shared for obtaining both SubByte and InvSubByte. By using XORs and Self_XORs the last round computation is able to share with the inner round, which further increases the reconfigurable area efficiency. SB/SB' also is shown in Fig.5.

4. RESULT & OUTPUT



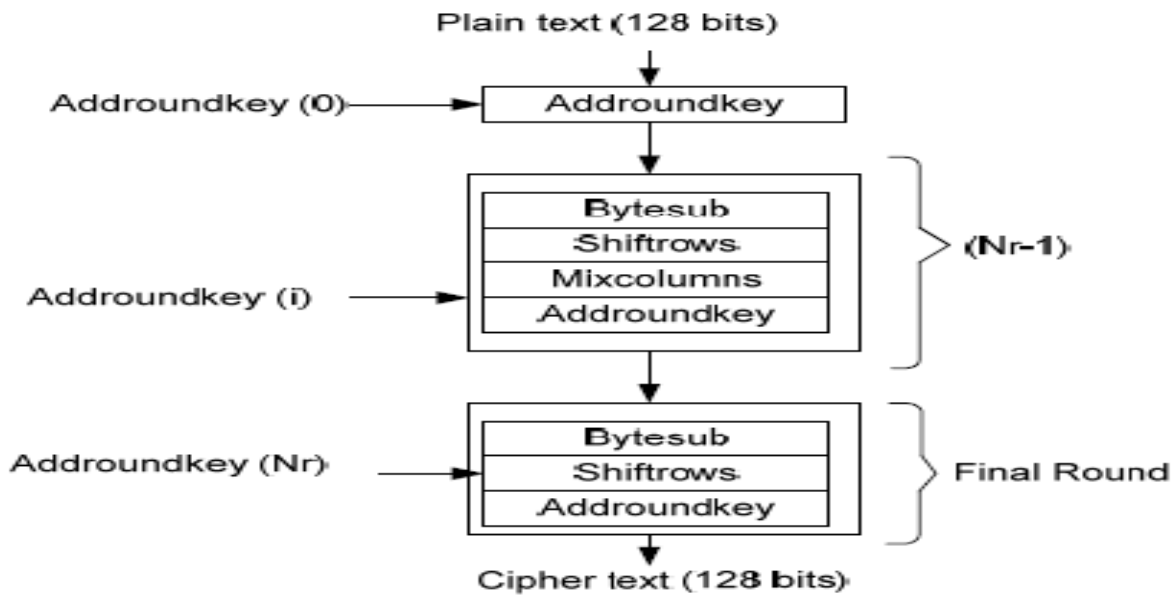
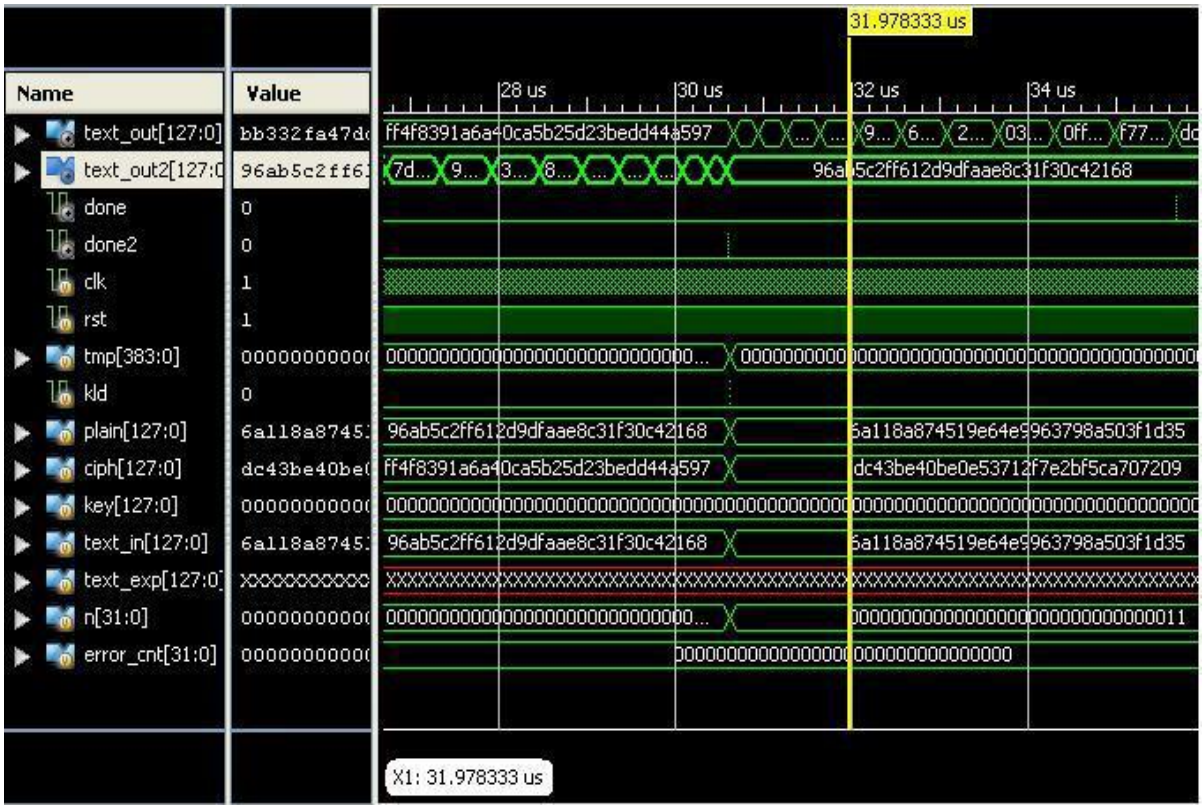
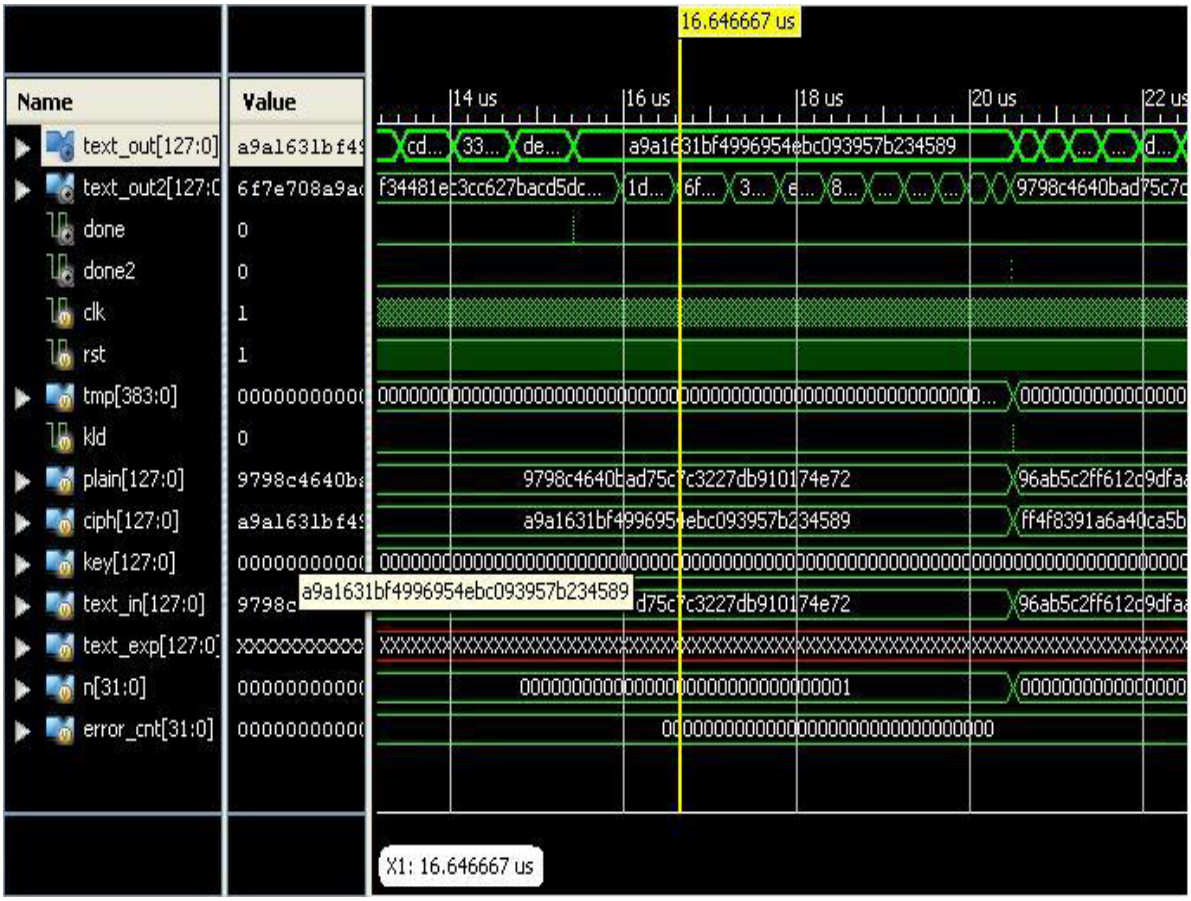
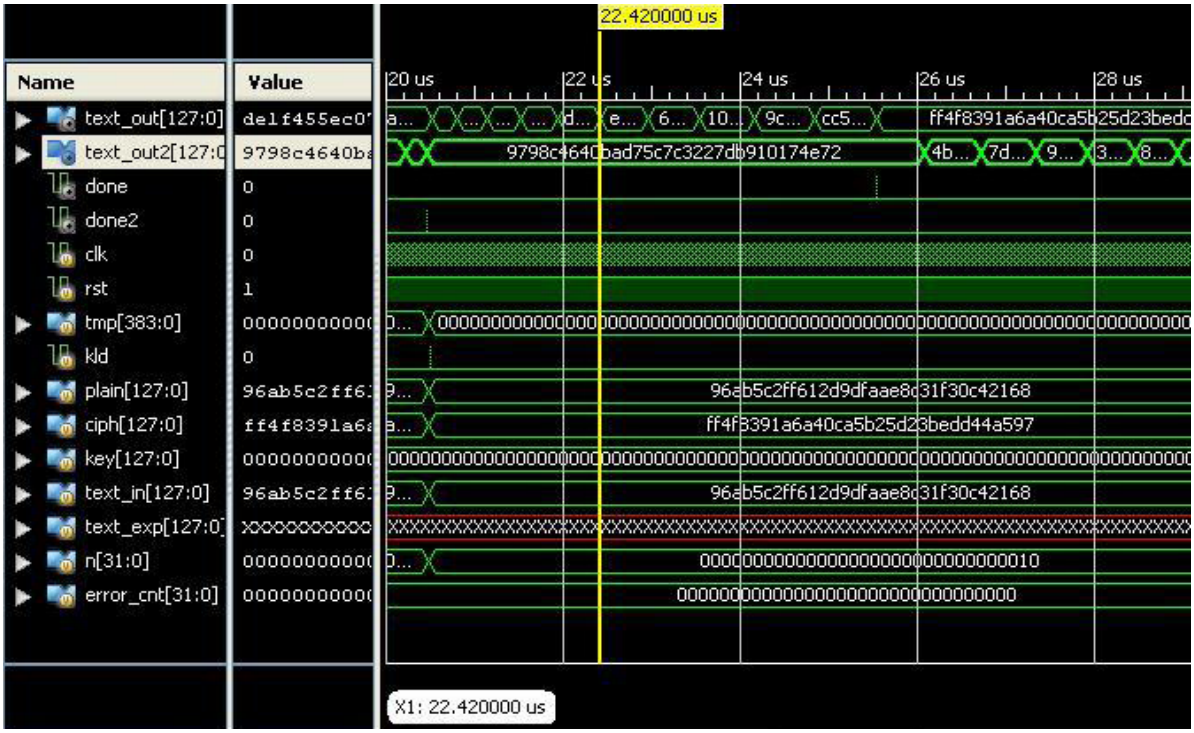


Figure: AES algorithm Encryption structure


31.978333 us
Name
Value
28 us
30 us
32 us
34 us
text_out[127:0]
text_out2[127:0]
done
done2
clk
rst
tmp[383:0]
kld
plain[127:0]
ciph[127:0]
key[127:0]
text_in[127:0]
text_exp[127:0]
n[31:0]
error_cnt[31:0]
X1: 31.978333 us


16.646667 us
Name
Value
14 us
16 us
18 us
20 us
22 us
text_out[127:0]
text_out2[127:0]
done
done2
clk
rst
tmp[383:0]
kld
plain[127:0]
ciph[127:0]
key[127:0]
text_in[127:0]
text_exp[127:0]
n[31:0]
error_cnt[31:0]
a9a1631bf4996954ebc093957b234589
X1: 16.646667 us



5. COMPARISON TABLE

Speed Comparisons of Block Ciphers on a Pentium

Algorithm	Clock cycles per round	# of rounds	# of clock cycles per byte encrypted	Notes
Blowfish	9	16	18	Free, unpatented
Khufu/Khafre	5	32	20	Patented by Xerox
RC5	12	16	23	Patented by RSA Data Security
DES	18	16	45	56-bit key
IDEA	50	8	50	patented by Ascom-Systec
Triple-DES	18	48	108	

From the designed methods the various parameters aretabulated and compared. With this comparison results the Proposed methods.

6. CONCLUSION

The Advanced Encryption Standard (AES) is the current encryption standard intended to be used by U.S. Government organizations to protect sensitive (and even secret and top secret) information. It is also becoming a (de facto) global standard for commercial software and hardware that use encryption or other security features. Rijndael is an encryption algorithm that has been designed with the state of art in the cryptographic research and is still believed very secure by most of the people.

It has been designed to have very strong resistance against the classical approximation attacks, such as linear cryptanalysis, differential cryptanalysis etc. However since Rijndael is very algebraic, new algebraic attacks appeared. Rijndael is an encryption algorithm that has been designed with the state of art in the cryptographic research and is still believed very secure by most of the people. It has been designed to have very strong resistance against the classical approximation attacks, such as linear cryptanalysis, differential cryptanalysietc. However since Rijndael is very algebraic, new algebraic attacks appeared.

REFERENCES

[1] Software Performance Results from the eSTREAM Project, e STREAM, the ECRYPT Stream Cipher Project, http://www.ecrypt.eu.org/stream/perf/#results, 2012.

[2] The Current eSTREAM Portfolio, e STREAM, the ECRYPT Stream Cipher Project, http://www.ecrypt.eu.org/stream/index.html, 2012.

[3] S.R. Fluhrer and D.A. McGrew, "Statistical Analysis of the Alleged RC4 Key stream Generator," Proc. Seventh Int'l Workshop Fast Software Encryption (FSE '00), vol. 1978, pp. 19-30, 2000.

[4] S.R. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," Proc. Eighth Ann. Int'l Workshop Selected Areas in Cryptography (SAC '01), vol. 2259, pp. 1-24, 2001.

[5] M.D. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos, and C.E. Goutis, "Comparison of the Hardware Implementation of Stream Ciphers," Int'l Arab J. Information Technology, vol. 2, no. 4, pp. 267-274, 2005.