IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-6, Issue-8, Aug 2018 E-ISSN: 2347-2693

A Review on Advanced Encryption Standard – (AES)

Ajit Karki

School of Information Technology, ICFAI University, Sikkim

*Corresponding Author: ajitkarki@iusikkim.edu.in

Available online at: www.ijcseonline.org

Accepted: 13/Aug/2018, Published: 31/Aug/2018

Abstract- Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication. In data and telecommunications, cryptography is necessary when communicating over any unreliable medium, which includes any network particularly the internet. AES encryption is the solution for Data Encryption Standard (DES) aging problem. Rijndanel Symmetric block cipher standard version which can encrypt and decrypt plaintext 128 bits blocks using key with 128-bit, 192-bit, or 256-bit size. The Rijndael cipher has simple structure and suitable to 8-bit and 32-bit processing. The cipher has numbers round of plaintext transformation. Key length determines how many rounds to be executed. The key with 128-bit use 10 rounds, 192-bit use 12 rounds, and 256-bit use 14 rounds. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm uses a round function that is composed of four different byte-oriented transformations: Sub Bytes, Shift Rows, Mix columns and Add Round Key.

Keywords: Advanced Encryption Standard, Cryptography, Decryption, Encryption.

I. Introduction

Effective May 26, 2002 the National Institute of Science and Technology (NIST) has selected a block cipher called RIJNDAEL (named after its creators Vincent Rijmen and Joan Daemen) as the symmetric key encryption algorithm to be used to encrypt sensitive but unclassified American federal information. The Advanced Encryption Standard (AES) is formal encryption method adopted by the National Institute of Standards and Technology of the US Government, and is accepted worldwide. In 1997 the National Institute of Standards and Technology (NIST), a branch of the US government, started a process to identify a replacement for the Data Encryption Standard (DES)[1]. It was generally recognized that DES was not secure because of advances in computer processing power. The goal of NIST was to define a replacement for DES that could be used for non-military information security applications by US government agencies. Of course, it was recognized that commercial and other non-government users would benefit from the work of NIST and that the work would be generally adopted as a commercial standard [2].

The NIST invited cryptography and data security specialists from around the world to participate in the discussion and selection process. Five encryption algorithms were adopted for study. Through a process of consensus the encryption algorithm proposed by the Belgium cryptographers Joan Daeman and Vincent Rijmen was selected. Prior to selection Daeman and Rijmen used the name Rijndael (derived from their names) for the algorithm. After adoption the encryption algorithm was given the name Advanced Encryption Standard (AES) which is in common use today. In 2000 the NIST formally adopted the AES encryption algorithm and published it as a federal standard under the designation FIPS-197. The full FIPS-197 standard is available on the NIST website. As expected, many providers of encryption software and hardware have incorporated AES encryption into their products.

A. AES Algorithm:

AES is an iterated symmetric block cipher, which means that:

- AES works by repeating the same defined steps multiple times.
- AES is a secret key encryption algorithm.
- AES operates on a fixed number of bytes.

AES as well as most encryption algorithms is reversible. This means that almost the same steps are performed to complete both encryption and decryption in reverse order. The AES algorithm operates on bytes, which makes it simpler to implement and explain.
This key is expanded into individual sub keys, a sub keys for each operation round. This process is called KEY EXPANSION.
As mentioned before AES is an iterated block cipher. All that means is that the same operations are performed many times on a fixed number of bytes. These operations can easily be broken down to the following functions:

- Add Round Key
- Byte Sub
- Shift Row
- Mix Column

AES is a symmetric encryption algorithm processing data in block of 128 bits. AES is symmetric since the same key is used for encryption and the reverse transformation, decryption [2]. The only secret necessary to keep for security is the key. AES may configured to use different key-lengths, the standard defines 3 lengths and the resulting algorithms are named AES-128, AES-192 and AES-256 respectively to indicate the length in bits of the key.

B. AES Algorithm Specification:

For the AES algorithm, the length of the input block, the output block and the State is 128 bits. This is represented by Nb = 4, which reflects the number of 32-bit words (number of columns) in the state[3].

Table 1. Key-Block-Round Combinations.

Bit pattern	Key Length (NK Words)	Block Size (NB Words)	No Of Rounds (NR Words)
AES- 128	4	4	10
AES- 192	6	4	12
AES- 256	8	4	14

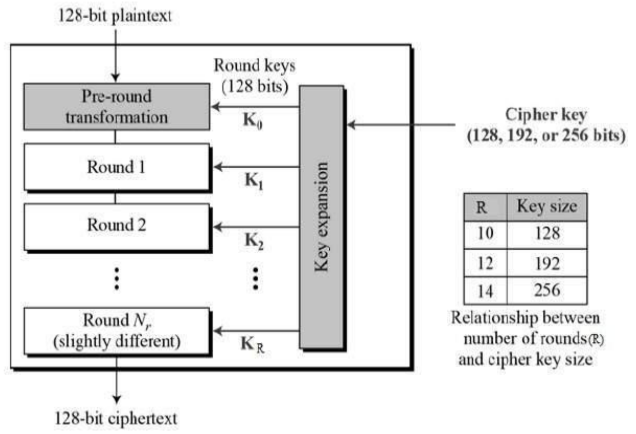

Figure 1: General Structure of AES Algorithm. [1]

An implementation of the AES algorithm shall support at least one of the three key lengths: 128, 192, or 256 bits (i.e., Nk = 4, 6, or 8, respectively). Implementations may optionally support two or three key lengths, which may promote the interoperability of algorithm implementations. For the AES algorithm, the length of the Cipher Key, K, is 128, 192 or 256 bits. The key length is represented by Nk = 4, 6, or 8which reflects the number of 32-bit words (number of columns) in the Cipher Key. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by Nr, where Nr = 10 when Nk = 4, Nr = 12 when Nk = 6, and Nr = 14 when Nk = 8. The only Key-Block-Round combinations that conform to this standard are given in Table 1.

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations:

1) Byte substitution using a substitution table (S-box),
2) Shifting rows of the State array by different offsets,
3) Mixing the data within each column of the State array, and
4) Adding a Round Key to the State.

Table.2

63	7C	77	7B
CA	82	C9	7D
B7	FD	93	26
04	C7	23	C3

II. ENCRYPTION

In encryption mode, the initial key is added to the input value at the very beginning, which is called an initial round. This is followed by 9 iterations of a normal round and ends with a slightly modified final round, as one can see in Figure 2. During one normal round the following operations are performed in the following order: Sub Bytes, Shift Rows, Mix Columns, and Add Round key. The final round is a normal round without the Mix Columns stage[4].

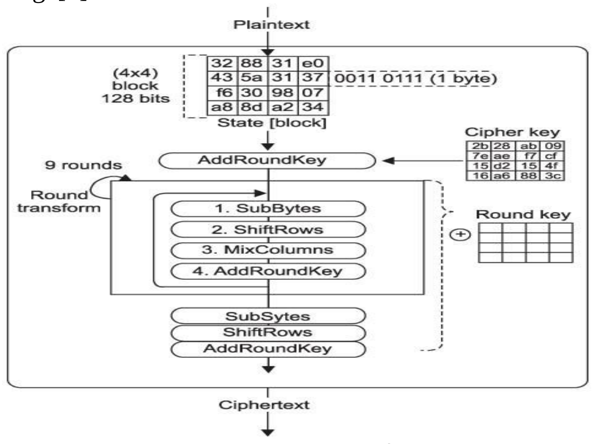

Fig 2: General structure of Encryption.[2]

A. Steps in AES Encryption:

- Sub Bytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
- Shift Rows—a transposition step where each row of the state is shifted cyclically a certain number of steps.
- Mix Columns—a mixing operation which operates on the columns of the state, combining the four bytes in each column
- 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

B. Sub bytes Transformation:

The Sub Bytes transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). This S-box which is invertible is constructed by composing two transformations:
1. Take the multiplicative inverse in the finite field GF (28), the element {00} is mapped to itself.
2. Apply the following affine transformation (over GF (2)):
For 0<i<8, where b_i is the ith bit of the byte, and c_i is the ith bit of a byte c with the Value {63} or {01100011}. Here and elsewhere, a prime on a variable (e.g., b)
Indicates that the variable is to be updated with the value on the right. In matrix form, the affine transformation element of the S-box can be expressed as:

$$\begin{bmatrix} a7 \\ a6 \\ a5 \\ a4 \\ a3 \\ a2 \\ a1 \\ a0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} b7 \\ b6 \\ b5 \\ b4 \\ b3 \\ b2 \\ b1 \\ b0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Fig 3: Affine transformation [2]

Fig 4: S-Box[2]

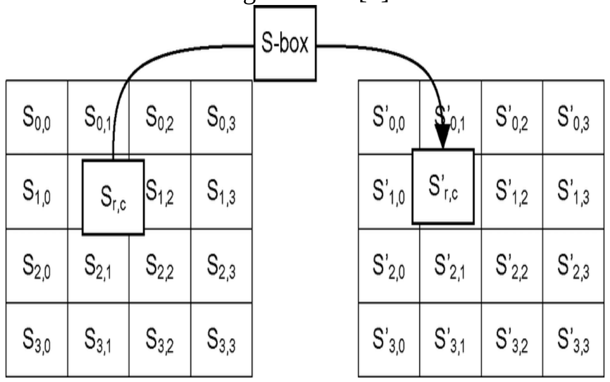

Fig.5: Effect of the Sub Bytes () transformation on the State.[2]

C. Shift Rows Transformation:

In the Shift Rows transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row is not shifted at all, the second row is shifted by one the third row by two, and the fourth row by three bytes to the left. Specifically, the Shift Rows transformation proceeds as follows: The shift value shift(r, Nb) depends on the row number, r, as follows (recall that Nb = 4): shift(1,4) 1; shift(2,4) 2 ; shift(3,4) 3.This has the effect of moving bytes to ―lower‖ positions in the row (i.e., lower values of c in a given row), while the ―lowest‖ bytes wrap around into the ―top‖ of the row (i.e., higher values of c in a given row)

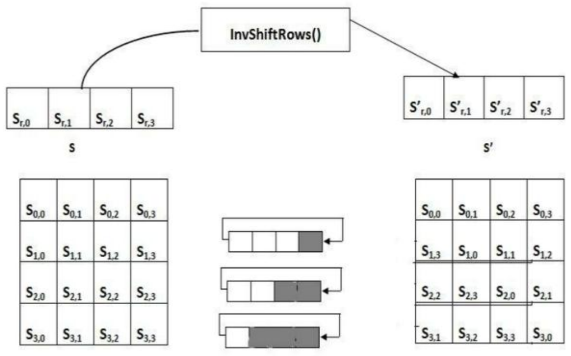

Fig.6: Shift Rows cyclically shifts the last three rows in the State.[2]

D. MixColumns Transformation:

The Mix Columns transformation operates on the State column-by-column, treating each column as a four-term polynomial.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < Nb.$$

As a result of this multiplication, the four bytes in a column are replaced by the following:

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

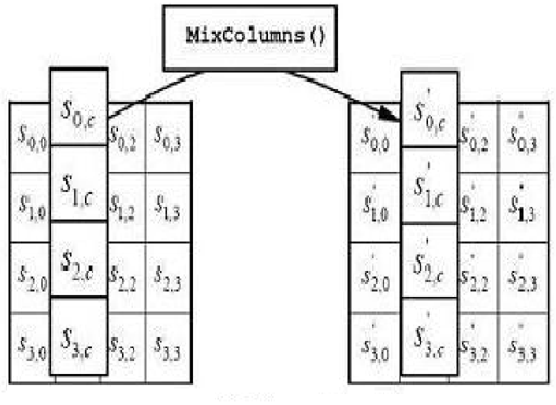

Fig.7: Mix Columns operates on the State column-by-column.[3]

E. Add round Key Transformation:

In the Add Round Key transformation, a Round Key is added to the State by a simple bitwise XOR operation. Each Round Key consists of Nb words from the key schedule. Those Nb words are each added into the columns of the State, such that [wi] are the key schedule words, and round is a value in the range 0 round Nr. In the Cipher, the initial Round Key addition occurs when round = 0, prior to the first application of the round function. The application of the Add round key transformation to the Nr rounds of the Cipher occurs when 1<round <Nr. The action of this transformation is illustrated in fig. 8, where 1 = round *Nb.

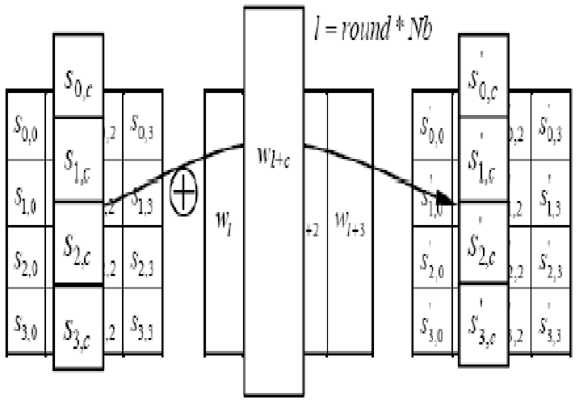

Fig.8: AddRoundKey XORs each column of the State with a word from the key schedule.[3]

F. Key Expansion:

The AES algorithm takes the Cipher Key, K, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of Nb (Nr + 1) words: the algorithm requires an initial set of Nb words, and each of the Nr rounds requires Nb words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [wi], with i in the range 0 < i < Nb(Nr + 1). The expansion of the input key into the key schedule proceeds according to the pseudo code. SubWord is a function that takes a four-byte input word and applies the S-box to each of the four bytes to produce an output word. The function Rot Word takes a word [a0, a1, a2, a3] as input, performs a cyclic permutation, and returns the word [a1, a2,a3,a0]. The round constant word array, Rcon[i], contains the values given by [xi-1,{00},{00},{00}], with x i-1 being powers of x (x is denoted as {02}) in the field GF(28). The first Nk words of the expanded key are filled with the Cipher Key. Every following word, w[i], is equal to the XOR of the previous word, w[i-1], and the word Nk positions earlier, w[i-Nk]. For words in positions that are a multiple of Nk, a transformation is applied to w[i-1] prior to the XOR, followed by an XOR with a round constant, Rcon[i]. This transformation consists of a cyclic shift of the bytes in a word (RotWord), followed by the application of a table lookup to all four bytes of the word (SubWord). It is important to note that the Key Expansion routine for 256-bit Cipher Keys (Nk = 8) is slightly different than for 128- and 192-bit Cipher Keys. If Nk = 8 and i-4 is a multiple of Nk, then SubWord () is applied to w [i-1] prior to the XOR.

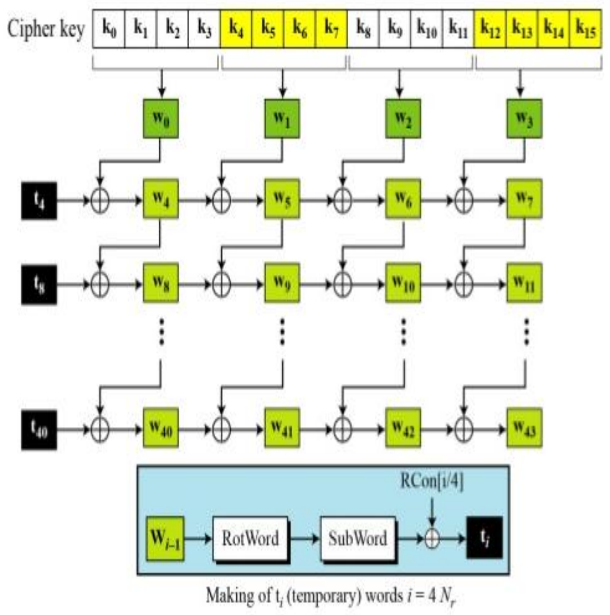
Cipher key
k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10} k_{11} k_{12} k_{13} k_{14} k_{15}
w_0 w_1 w_2 w_3
t_4 w_4 w_5 w_6 w_7
t_8 w_8 w_9 w_{10} w_{11}
t_{40} w_{40} w_{41} w_{42} w_{43}
RCon[i/4]
w_{i-1}
RotWord
SubWord
t_i
Making of t_i (temporary) words $i = 4\,N_r$

Fig.9. key expansion [4].

III. DECRYPTION

In decryption mode, the operations are in reverse order compared to their order in encryption mode. Thus it starts with an initial round, followed by 9 iterations of an inverse normal round and ends with an AddRoundKey. An inverse normal round consists of the following operations in this order: AddRoundKey, InvMixColumns, InvShiftRows, and InvSubBytes. An initial round is an inverse normal round without the InvMixColumns.

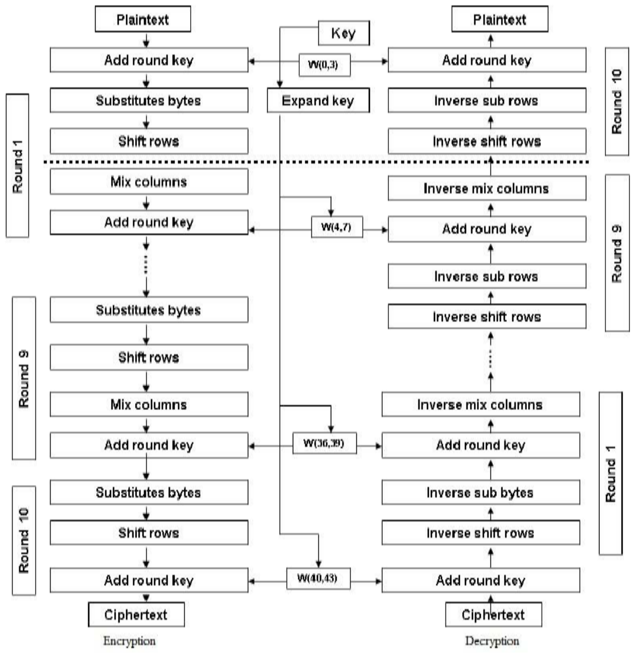

Fig 10: General structure of Decryption. [5]

Table.3

52	09	6A	D5
7C	E3	39	82
54	7B	94	32
08	2E	A1	66

A. Inv Shift rows Transformation:
InvShiftRows is the inverse of the ShiftRows transformation. The bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, r = 0, is not shifted. The bottom three rows are cyclically shifted by Nb - shift(r, Nb) bytes, where the shift value shift(r,Nb) depends on the row number.

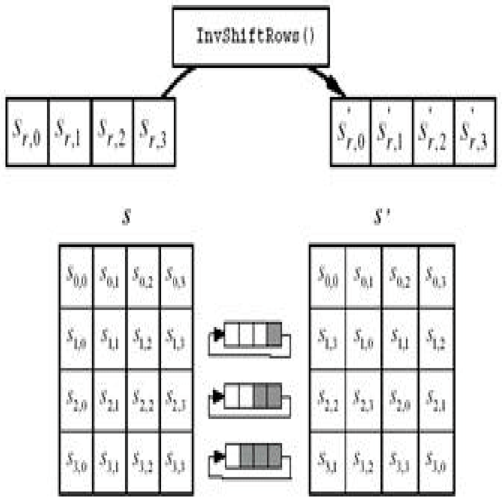

Fig.11.InvShiftRows transformation [4].

B. Inv Subbytes Transformation:
InvSubBytes is the inverse of the byte substitution transformation, in which the inverse Sbox is applied to each byte of the State. This is obtained by applying the inverse of the affine transformation followed by taking the multiplicative inverse in GF (28).The inverse S-box used in the InvSubBytes () transformation is presented in Fig 12.

C. Inv MixColumns Transformation:
InvMixColumns is the inverse of the MixColumns transformation. InvMixColumns operates on the State column-by-column, treating each column as a four term polynomial. The columns are considered as polynomials over GF (28) and multiplied modulo x4 + 1 with a fixed polynomial a-1(x), given by a-1(x) = {0b} x3 + {0d} x2 + {09} x + {0e}, this can be written as a matrix multiplication. Let As a result of this multiplication, the four bytes in a column are replaced by the following:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < Nb.$$

As a result of this multiplication, the four bytes in a column are replaced by the following:

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

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

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

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

D. Inverse of the Addroundkey Transformation:
AddRoundKey is its own inverse, since it only involves an application of the XOR operation. Equivalent Inverse Cipher transformations differ from that of the Cipher, while the form of the key schedules for encryption and decryption remains the same. However, several properties of the AES algorithm allow for an Equivalent Inverse Cipher that has the same sequence of transformations as the Cipher (with the transformations replaced by their inverses). This is accomplished with a change in the key schedule. The two properties that allow for this Equivalent Inverse Cipher are as follows: The Sub Bytes and Shift Rows transformations commute; that is, a Sub Bytes transformation immediately followed by a Shift Rows transformation is equivalent to a Shift Rows transformation immediately followed by a Sub Bytes transformation.

The same is true for their inverses, InvSubBytes and InvShiftRows. The column mixing operations - MixColumns and InvMixColumns – are linear with respect to the column input, which means Inv MixColumns(state XOR Round Key) =InvMixColumns(state)XORInvMixColumns(RoundKey).T hese properties allow the order of InvSubBytes and InvShiftRows transformations to be reversed. The order of the AddRoundKey and InvMixColumns transformations can also be reversed, provided that the columns (words) of the decryption key schedule are modified using the InvMixColumns transformation. The equivalent inverse cipher is defined by reversing the order of the InvSubBytes and InvShiftRows transformations and by reversing the order of the AddRoundKey and InvMixColumns transformations used in the ―round loop‖ after first modifying the decryption key schedule for round = 1 to Nr-1 using the InvMixColumns transformation. The first and last Nb words of the decryption key schedule shall not be modified in this manner [5].

IV. AES APPLICATIONS

AES Encryption and Decryption has many applications. It is used in cases where data is too sensitive that only the authorized people are supposed to know and not to the rest. The following are the various applications:

- Secure Communication
- Smart Cards
- RFID.
- ATM networks.
- Image encryption
- Secure Storage
- Confidential Cooperate Documents
- Government Documents
- FBI Files
- Personal Storage Devices
- Person Information Protection

V. CONCLUSION

The Rijndael algorithm was chosen as the new Advanced Encryption Standard (AES) for several reasons. The purpose was to create an algorithm that was resistant against known attacks, simple, and quick to code. Choosing to use field GF(2)8 was a very good decision. The inverse of the addition operation was itself, making much of the algorithm easy to do. In fact, every operation is invertible by design. In addition, the block size and key size can vary making the algorithm versatile. AES was originally designed for non-classified U.S. government information, but, due to its success, AES-256 is usable for top secret government information [6]. As of July 2009, no practical attacks have been successful on AES [7]. Advanced encryption standard algorithm is the efficient algorithm and analyzes the result in terms of speed, area, power consumption with the Standard algorithm. This is based on the software as well as hardware system so that security is high as compared to the traditional algorithm. Also the key system is sufficient to encrypt the data and this is suitable for any application where the requirement is strong encryption technology.

REFERENCES

[1]. Selent, D. Advanced encryption standard. Rivier Academic Journal 2010.

[2]. Sarker, M.Z.H., Parvez, M.S. A cost effective symmetric key cryptographic algorithm for small amount of data. In: 9th International Multitopic Conference, IEEE INMIC 2005. IEEE; 2005, p. 1–6.

[3]. Stallings, W., Tahiliani, M.P.. Cryptography and network security: principles and practice; vol. 6. Pearson London; 2014.

[4]. Satoh, A., Morioka, S., Takano, K., Munetoh, S. Compact rijndael hardware architecture with s-box optimization. In: Asiacrypt; vol. 2248. Springer; 2001, p. 239–254.

[5]. Ahmad, N., Hasan, R., Jubadi, W.M.. Design of aes s-box using combinational logic optimization. In: Industrial Electronics & Applications (ISIEA), 2010 IEEE Symposium on. IEEE; 2010, p. 696–699.

[6]. Z. Yuan, Y. Wang, J. Li, R. Li and W. Zhao, “FPGA based optimization for masked AES implementation”, Proc. IEEE 54th

[7]. International Midwest Symposium on Circuits and Systems (MWSCAS), pp.1-4 August 2011.

[8]. B.Schneier. Applied Cryptography. John Wiley and Sons, second edition, 2012.

[9]. Cryptography and Elliptic Curves, koblitz, second edition, 2011.

[10]. Julio Lopez and Ricardo Dahab, “An overview of elliptic curve cryptography”, May 2011.

[11]. V. Miller, “Uses of elliptic curves in cryptography”, Advances in Cryptology -CRYPTO'85, LNCS 218, pp.417-426, 2011.

[12]. Botes, J.J., Penzhorn, W.T., 1994. An implementation of an elliptic curve cryptosystem. Communications and Signal Processing. COMSIG-94. In Proceedings of the 1994 IEEE South African Symposium, 85 -90.

Authors Profile

Mr. Ajit Karki pursued Master of Technology (CSE) from Sikkim Manipal Institute of Technology. He is currently working as Assistant Professor in Department of Information Technology since 2006 at ICFAI University, Sikkim. He has published more than 10 research papers in reputed international journals including Thomson Reuters (SCI & Web of Science) and conferences including IEEE and it’s also available online. His main research work focuses on Cryptography Algorithms, Network Security, Cloud Security and Privacy, Data Mining, Computational Intelligence based education. He has 14 years of teaching experience and 5 years of Research Experience.