International Journal of Research in Engineering, Science and Management
Volume 5, Issue 8, August 2022
https://www.ijresm.com | ISSN (Online): 2581-5792

AES-128 Algorithm Design with Verilog

Mahalsa Sai Dontha*
Department of Electronics and Communication Engineering, Sri Indu College of Engineering & Technology, Sheriguda, India

***Abstract*: The issue of multimedia data security has emerged as a matter of essential concern due to the increasing growth of digital data exchanges through unsecure networks. For this reason, multimedia files are encrypted using cryptographic protocols to prevent unauthorized access. The US government uses the symmetric block cipher Advanced Encryption Standard (AES) to protect sensitive information. The Advanced Encryption Standard (AES) is used to encrypt private information all over the world. This standard is built into both software and hardware. It is crucial for cybersecurity and the protection of digital information and government networks. Maintaining one's right to privacy is becoming more and more important, not just for regular citizens but also for governments that want to stop signal and data interception. Maintaining one's right to privacy is becoming more and more important, not just for regular citizens but also for governments that want to stop signal and data interception. With the rise of personal communications devices, people are asking for more privacy in conversations that used to be unprotected. The length of the key is the main determining factor between DES and AES. This is because they are both examples of symmetric key block ciphers (56 bit for DES). When encrypting messages that are bigger than the size of a block, the right mode of operation must be chosen. After AES has been implemented, I will explain how this mode of operation functions. The original Rijndael allowed for key and block sizes of any multiple of 32 bits, from a minimum of 128 bits up to a maximum of 256 bits. This was in contrast to AES, which supported blocks with a fixed-size support of 128 bits and a range of 128, 192, and 256-bit key sizes. However, AES has a limitation in that its maximum block size is 128 bits.**

***Keywords*: AES, cryptography, DES, encrypt, security.**

1. Introduction

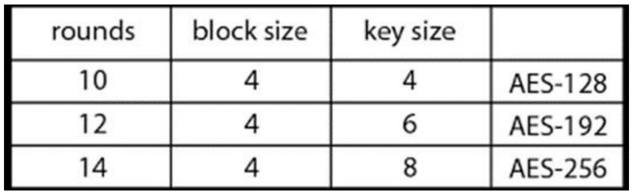
It takes a block of a specific size as input (usually 128) and produces a comparable block of the same size as output. A second input, the hidden key, is required for the transformation to occur. AES uses three different key lengths: 128 bits, 192 bits, and 256 bits. It is important to understand that the length of the secret key may vary depending on the cipher used. Digital video recorders use AES for both picture transmission and reception.

In today's financial operations, AES is used to exchange data in a way that is very safe. In today's financial operations, AES is used to exchange data in a way that is very safe.

Cryptography is the study of secret codes that make it possible to send secret information over a medium that is not secure. It safeguards against unauthorized persons by prohibiting illicit alterations to usage. The process of converting plaintext to ciphertext employs a cryptographic procedure, typically involving the usage of a key.

A. Number of Rounds

The round function, which consists of the four distinct byte-oriented transformations Sub Bytes, Shift Rows, Mix Columns, and Add Round Key, regulates the number of rounds the AES algorithm performs. A number of AES parameters are impacted by key length, as the table illustrates.



rounds	block size	key size	
10	4	4	AES-128
12	4	6	AES-192
14	4	8	AES-256

Fig. 1. Number of rounds

B. Background of the AES Algorithm

Cryptography is the study of secret codes that make it possible to send secret messages over a shaky channel. It protects against unauthorized parties by preventing unauthorized changes to utilization. In general, it uses a cryptographic framework to turn plaintext into ciphertext, and it often needs a key to do this.

A block cipher is a type of encryption that converts a fixed-length block of plaintext data into a fixed-length block of ciphertext data using a symmetric key. This transition occurs as a result of the user's use of a secret key. As illustrated in the diagram below.

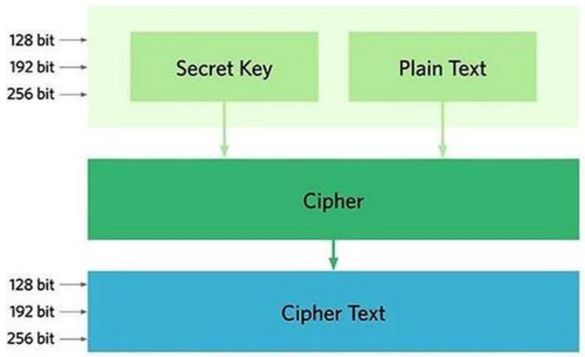


Fig. 2. AES model

2. History

The Advanced Encryption Standard (AES) was first suggested in 1997. With assistance from businesses and the

*Corresponding author: mahalsasai333@gmail.com cryptography community, NIST created it. The primary objective was to create a Federal Information Processing Standard (FIPS) that specified an encryption algorithm capable of safeguarding sensitive government data well into the twenty-first century. The US government and the private sector were both supposed to use the algorithm on a voluntary basis. On January 2, 1997, NIST said that work on the AES would begin, and they got a lot of feedback. On September 12, 1997, NIST issued a formal call for algorithms.

The call stated that the AES would specify an unclassified, widely accepted encryption algorithm or algorithms that would be made available without any restrictions anywhere in the world. Additionally, the algorithm(s) must support 128-bit block sizes, 128-, 192-, and 256-bit key sizes, as well as symmetric key cryptography as a block cypher.

At the First AES Candidate Conference (AES1), which was held on August 20, 1998, NIST announced a group of fifteen AES candidate algorithms. These algorithms were supplied by cryptography community members from all across the world. At the meeting and in an announcement in the Federal Register ("Round 1"), NIST asked the public what they thought about the candidates.

The Second AES Candidate Conference (AES2) was held in March 1999 to discuss the findings of the global cryptography community's analysis of the candidate algorithms. On April 15, 1999, the window for public comments on the initial review of the algorithms closed. Based on the analyses and Round 1 comment, NIST chose five of the fifteen algorithms.

The AES finalist candidate algorithms were MARS, RC6, Rijndael, Serpent, and Two Finish, and NIST created a Round1 Report outlining the selection process.

3. AES Algorithm Encryption Model

The recommended approach employs AES-128 encryption. A 4-4 state and key matrix controls the handling of each 128-bit data block and encryption key. The key matrix is initialized with the user input key, whereas the state matrix is initialized with the original plain text at the beginning of the process. The user-entered key is the first element of the key matrix, and the original plain text is the first element of the state matrix. The four following fundamental operations are included in each cycle of the AES algorithm:

1. Sub-byte
2. Shift row
3. Mixed column
4. Add Round Key.

The mixed column operation is not carried out in the final round of encryption, which makes it different.

A. *Sub Byte*

The SubBytes() transformation is a non-linear byte replacement that uses a substitution table to operate on each byte of the State separately (S-box).

B. *Shift Row*

By repeatedly shifting the bytes in each row by an AES has exclusive use of the front row, the Shift Rows step operates on the rows of the state. The second row's bytes are shifted one position to the left. offset. The first row is left alone for AES. Each byte in the second row is moved to the left by one. Similarly, the third and fourth rows are displaced by two and three offsets, respectively.

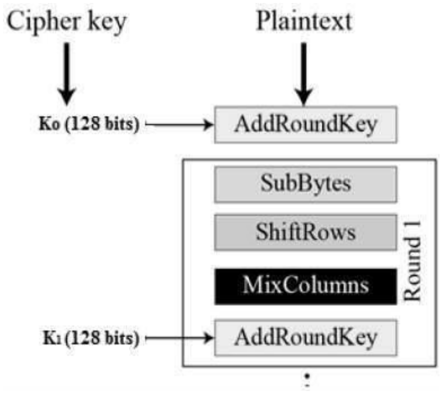


Fig. 3. Illustrates the 10 rounds of AES-128 encryption

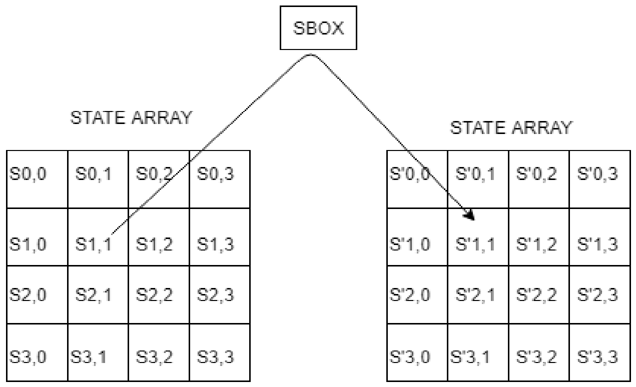


Fig. 4. Sub byte

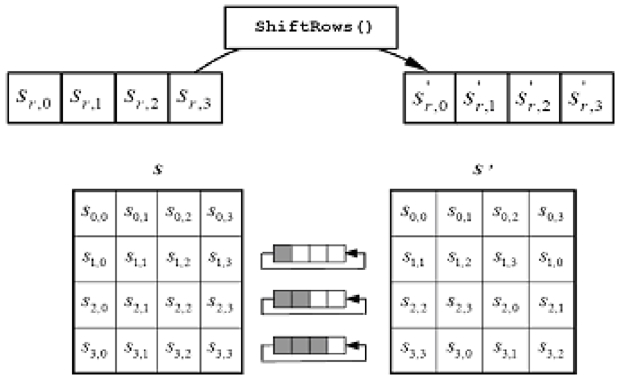


Fig. 5. Shift row

C. *Mixed Column*

Mix Column Transformation, which involves multiplying the input matrix (Over) by the MDS matrix, is the most expensive thing that AES can do. This shift is significant in terms of the cipher's wide trail strategy.

D. *Add Round Key*

AddRoundKey works one column at a time. The fact that it includes the cipher key is the most significant aspect of this transformation. A round key is added to the state array using a bitwise operation called XOR. as illustrated in figure 7.

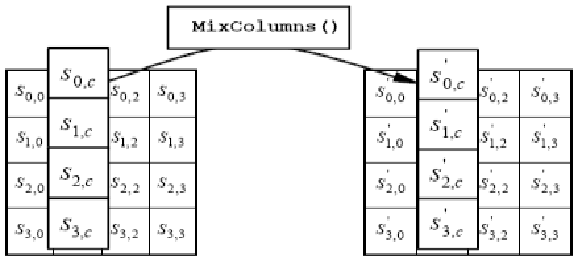
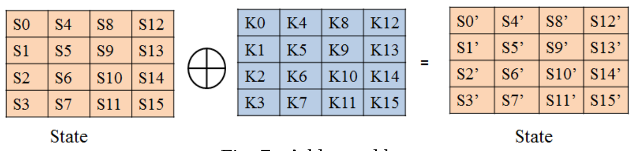


Fig. 6. Mix column



S0	S4	S8	S12
S1	S5	S9	S13
S2	S6	S10	S14
S3	S7	S11	S15

⊕

K0	K4	K8	K12
K1	K5	K9	K13
K2	K6	K10	K14
K3	K7	K11	K15

=

S0’	S4’	S8’	S12’
S1’	S5’	S9’	S13’
S2’	S6’	S10’	S14’
S3’	S7’	S11’	S15’

Fig. 7. Add round key

4. AES Description

The changes in the rounds are easy to undo because each step has an opposite that, when done, makes the changes go backwards. Each 128-block goes through 10, 12, or 14 rounds depending on the key size.

Following are the stages of each round of decryption:

1. Add round key
2. Inverse Mix Columns
3. Shift Rows
4. Inverse Sub Byte

Since the process of decryption is the opposite of the process of encryption, I will explain how the two are different in important ways. Because decryption is the inverse of encryption, the processes are carried out in the inverse order. The final round of encryption is decrypted as the first round.

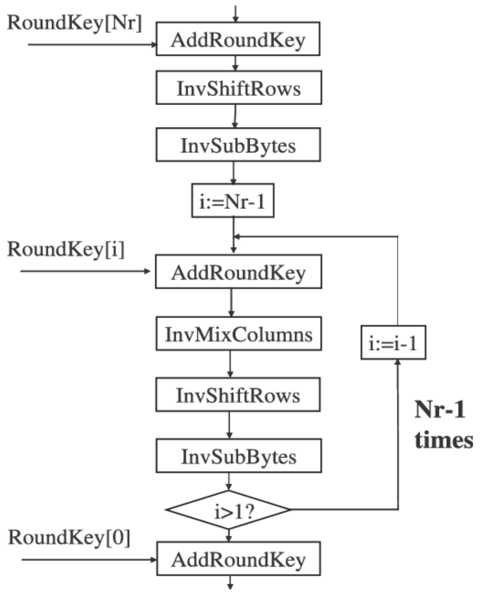


Fig. 8. AES decryption block diagram

A. *Inverse of Add Round Key*

Because it just comprises an application of the XOR operation AES, Add Round Key(), which was explained in the preceding round with the same matrix, is its own inverse.

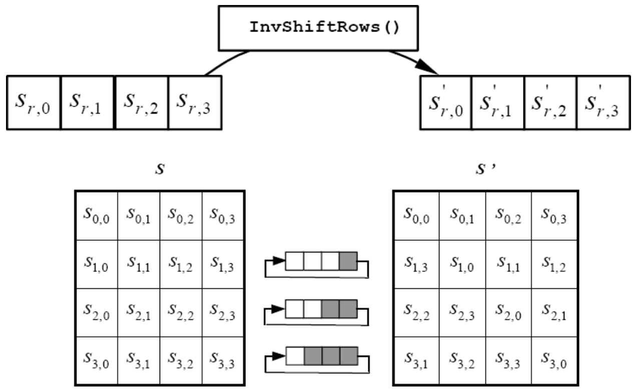
B. *Inverse Mix Columns*

InvSubBytes and InvShiftRows have the identical inverses. The inverses of InvSubBytes and InvShiftRows are the same. InvMixColumns(state XOR Round Key) XOR InvMixColumns(RoundKey) = InvMixColumns(state) XOR InvMixColumns(RoundKey). These properties may be used to reverse the order of InvSubBytes and InvShiftRows conversions.

As long as the columns (words) of the decryption key schedule are changed using the InvMixColumns transformation, the order of the AddRoundKey and InvMixColumns transformations may also be reversed.

C. *Inverse Shift Rows*

ShiftRows' inverse transformation is InvShiftRows() (). The bytes in the final three rows of the State are moved cyclically across varied numbers of bytes (offsets). The last three rows of the State's bytes cycle through different offsets or groups of bytes. No shift is applied to the first row (r = 0). The row number determines the shift value shift(r, Nb), which is displayed in fig. The bottom three rows are cycled by Nb shift(r, Nb) bytes.


InvShiftRows ()
$s_{r,0}$ $s_{r,1}$ $s_{r,2}$ $s_{r,3}$
$s'_{r,0}$ $s'_{r,1}$ $s'_{r,2}$ $s'_{r,3}$
s
s'

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

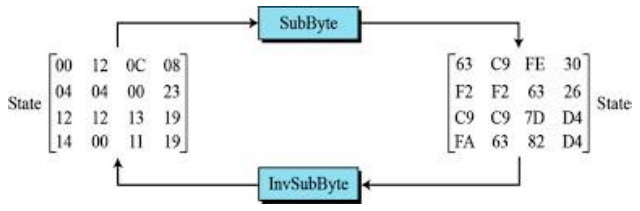
$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,3}$	$s_{1,0}$	$s_{1,1}$	$s_{1,2}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,1}$	$s_{3,2}$	$s_{3,3}$	$s_{3,0}$

Fig. 9. InvShiftRows () transformation

D. *Inverse Sub Bytes*

The byte substitution transformation's opposite, InvSubBytes(), applies the inverse Sbox to each of the State's bytes.S-boxes and inverse tables are additional names for SubBytes tables.

The SubBytes table is shaped like an inverse S-box. The affine transformation has two parts. In the multiplication part, the input is multiplied by the inverse constant matrix. In the addition part, a constant vector is added to the result of the multiplication. As illustrated in fig. 10.



$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix} \quad \begin{bmatrix} 63 & C9 & FE & 30 \\ F2 & F2 & 63 & 26 \\ C9 & C9 & 7D & D4 \\ FA & 63 & 82 & D4 \end{bmatrix}$$

Fig. 10. Inverse sub byte

5. Simulations and Result

Simulation results are based on test performed of Questa 10.0. The verification is done using Verilog HDL.

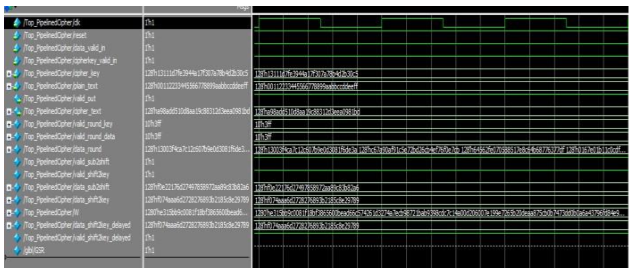
Test Case:

Plane Text: 00112233445566778899aabbccddeeff

Key: 13111d7fe3944917f307a78b4d2b30c5

Output/Cipher Text: 98add51098aa19c88312d3eea0981bd

The waveforms generated by the 128-bit full encryption process are displayed in Figure 11. Clocks 1 and 2 as well as Active High Reset, a 4-bit round, a 128-bit state and key, and a 4-bit round are the inputs. The data that has been 128-bit encrypted is the output.



```
                                                  Msgs
/Top_PipelinedCipher/clk                          1'h1
/Top_PipelinedCipher/reset                        1'h1
/Top_PipelinedCipher/data_valid_in                1'h1
/Top_PipelinedCipher/cipherkey_valid_in           1'h1
/Top_PipelinedCipher/cipher_key                   128'h13111d7fe3944a17f307a78b4d2b30c5
/Top_PipelinedCipher/plain_text                   128'h00112233445566778899aabbccddeeff
/Top_PipelinedCipher/valid_out                    1'h1
/Top_PipelinedCipher/cipher_text                  128'ha98add510d8aa19c88312d3eea0981bd
/Top_PipelinedCipher/valid_round_key              10'h3ff
/Top_PipelinedCipher/valid_round_data             10'h3ff
/Top_PipelinedCipher/data_round                   128'h13003f4ca7c12c607b9e0d3081f6de3...
/Top_PipelinedCipher/valid_sub2shift              1'h1
/Top_PipelinedCipher/valid_shift2key              1'h1
/Top_PipelinedCipher/data_sub2shift               128'hf0e22176d27497858972aa89c83b82a6
/Top_PipelinedCipher/data_shift2key               128'hf074aaa6d2728276893b2185c8e29789
/Top_PipelinedCipher/W                            1280'he315bb9c0081f18bf3865600bead6...
/Top_PipelinedCipher/data_shift2key_delayed       128'hf074aaa6d2728276893b2185c8e29789
/Top_PipelinedCipher/valid_shift2key_delayed      1'h1
/glbl/GSR                                         1'h1
```

Fig. 11. Result

6. Applications of AES

In hardware and software, sensitive data is encrypted using AES all over the world.

A password management service protects its users' credentials from hackers by using AES encryption.

Because it may be utilized in a wide range of software, firmware, hardware, or any combination, it is most commonly employed in Wi-Fi.

7. Conclusion

During the course of this research project, a pipelined architectural implementation of 128-bit AES encryption was demonstrated. Security and efficiency are the two aspects that are prioritized when developing new cyphers. The architecture was designed using Verilog, and RTL Compiler was used for the synthesizing process. Both the data length and key length of the AES (Advanced Encryption Standard) were 128 bits. The document that came before had only the most basic information that was needed to make the AES encryption algorithm. This AES solution will keep a high level of security while giving good throughput and being able to handle mistakes.

References

[1] Xinmiao Zhang and Keshab K. Parhi, "Implementation Approaches for the Advanced Encryption Standard Algorithm" IEEE 2002.

[2] X. Zhang and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," IEEE Transactions on Very Large Scale Integration Systems, vol. 12, no. 9, pp. 95 967, Sep. 2004.

[3] Archana garg et al, "Implementation of Advanced Encryption Standard Algorithm using VHDL", International Journal of Engineering Trends and Technology, Volume 4 Issue 9, pp. 3956- 3961, September 2013.

[4] Ashwini M. Deshpande, Mangesh S. Deshpande and Devendra N. Kayatanavar, "FPGA Implementation of AES Encryption and Decryptionm" International Conference on Control, Automation, Communication and Energy conservation -2009.

[5] Chih-Peng Fanand and Jun-Kui Hwang "FPGA Implementations of High Throughput Sequential and Fully Pipelined AES Algorithm," in International journal of Electrical Engineering, vol. 15, no. 6, pp. 447-455, 2008.

[6] Pachamuthu Rajalakshmi, "Hardware-software codesign of AES on FPGA," in International Conference on Advances in Computing, Communications and Informatics, pp. 1118-1122, 2010.

[7] Richa Sharma, Purnima Gehlot, S. R. Biradar, "VHDL Implementation of AES-128," UACEE International Journal of Advances in Electronics Engineering, Volume 3, Issue 2, pp. 17-20, 2013.

[8] Mehran Mozaffari-Kermani and Arash Reyhani-Masoleh "Efficient and High-Performance Parallel Hardware Architecture for the AES-GCM," in IEEE Transactions on Computers, vol. 61, no. 8, August 2012.

[9] Samir Palnitkar, "Verilog HDL-A Guide to Digital Design and Synthesis", Prentice Hall, pp. 3-10, 2003.

[10] Pallavi Atha et al, "Design & Implementation of AES Algorithm Over FPGA Using VHDL," in International Journal of Engineering, Business and Enterprise Applications, pp. 58-62, 2013.