International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 2, Issue 1, January 2015

An alternative method for forming the state matrix required for AES encryption algorithm to enhance security

[1]Mr. Vikrant Shende [2] Dr. Meghana Kulkarni,[3]Mr. Giridhar Sudi
[1]Asst. Prof. GIT, Belagavi. vikrant_shende@git.edugiridharsudi@git.edu
[2]Asso. Prof. VTU, Belagavi. meghanak@vtu.ac.in
[3]Asst. Prof. GIT, Belagavi.

Abstract: **This paper presents an alternative method for forming the state matrix required for AES encryption algorithm. The design uses alternative method for mapping the individual bytes of the block to different columns of the state matrix depending on previously defined encryption vector. This adds an extra layer of security to the process, but at the same time keeps algorithm simple to implement.**

Index Terms **AES, block cipher, modified AES, encryption algorithm**

I. INTRODUCTION

In January 1997, the National Institute of Standards and Technology invited proposals for new algorithms for the Advanced Encryption Standard to replace the old Data Encryption Standard. After two rounds of evaluation on the 15 candidate algorithms, NIST selected the Rijndael as the AES algorithm in October 2000. In cryptography, the AES is also known as Rijndael [1]. The AES algorithm has broad applications, including smart cards and cellular phones, WWW servers and automated teller machines, and digital video recorders. Compared to software implementations, hardware implementations of the AES algorithm provide more physical security as well as higher speed. The AES algorithm specifies 128-bit, 192-bit, and 256-bit modes [2]. The modifications suggested in this paper keeps the AES algorithm as it is but deals with the block to state matrix conversion in a different way such that a higher level of security is achieved without compromising on speed.

II. . Description of AES algorithm

The AES is a symmetric block cipher which can be used for encryption and decryption. AES is iterative in nature which means that the same operations are performed many times [3]. These operations can easily be broken down to the following functions.

Add round key : Every 16 bytes of the state is XORed with every 16 bytes of a portion of the expanded key for the current round.

Byte sub : Each value of the state is replaced with the corresponding SBOX value during encryption.

Shift row : Arrange the state in a matrix and then perform a circular shift for each row. The circular shift just moves each byte one space over.

Mix column : In this step there are two parts. The first part deals with which parts of the state are multiplied against which parts of the matrix. The second part deals with how this multiplication is implemented.

III. . Suggested Modification

In a normal aes algorithm, the block of data to be encrypted is directly converted into the state matrix. The procedure followed is fairly linear. The first byte is made as the first column of the state, secod byte as the second column and so on as shown in the figure 1 on the next page.

a modified approach to this allows for a non-linear method to map the bytes of the block into the state matrix. According to the modified algorithm, the first byte of the block can map into the first or the second column of the state matrix depending on a decision bit say 'a'. Also, if the first

byte takes the first column, the second byte is filled into the second column(similar to the normal algorithm) but if the first byte takes the second column(based on bit 'a'), then the second byte takes the first column. This concept now can be extended to the third and fourth byte of the block being filled into the third and fourth column of the state matrix based on another decision bit say 'b' [4].

based on this approach, we can get 4 permutations in which the the block is modified based on bits 'ab' being '00','01','10','11'. This is described in the figure 2 below. The modified block now formed is encrypted using the normal aes following all the steps and then sent. At the reciever, if we have information about the descision bits, the original block can be formed again by following the same operation as performed before encryption on the newly decrypted data.

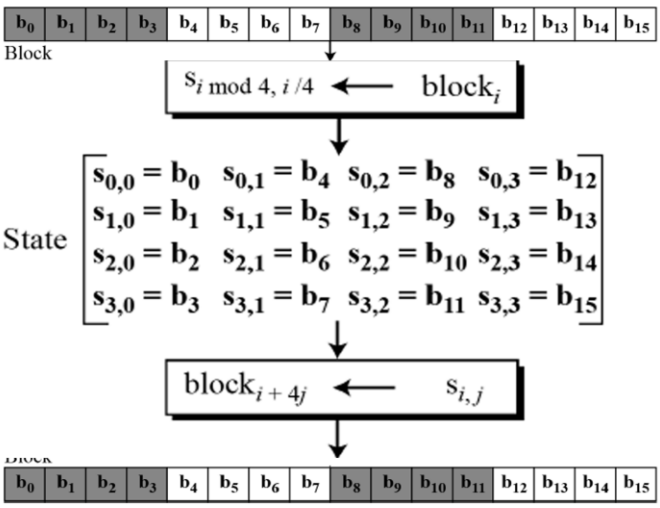


figure 1. Normal block to state conversion

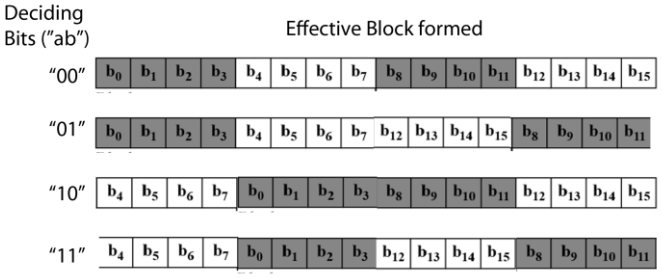


figure 2. Modified "effective" block

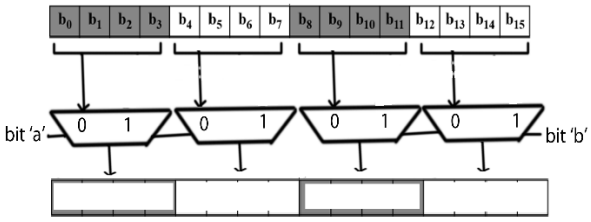
The modified block once formed can be linearly mapped into the state matrix as per the normal algorithm. Here, the decision bits 'a' & 'b' based on which the encryption occurs are of prime importance.

to keep the encryption as secure as possible, the decision bits 'a' and 'b' can be sent as just another block message by the sender and then stored locally by the reciever. As 2 bits are required for the encryption of an entire block, a fully filled "decision bit block"(128 bits) can accomodate for the decision bits for the next 64 blocks that occur. Thus, every multiple of 64, i.e the 0th, 64th, 128th message that is sent can be the matrix that contains the decision bits that would be used for the decryption of the next 64 messages at the reciever and hence must be stored. To summarize the modified algorithm :

- A message state matrix can contain regular data or contain the descision bit data for the next 64 block messages.
- The 0th, 64th, 128th, i.e the blocks which are a multiple of 64 will be the decision bit blocks.
- At the reciever, the matrix that contains decision bits is decrypted the normal way and is stored locally so that its contents are used to decrypt the later messages.
- To decrypt the regular messages, the bits from the saved matrix are retrieved sequentially, two at a time and are used as decision bits to decrypt the block.

IV. . Implementation at hardware level

the hardware implementation of this modified algorithm is fairly simple and can be implemented by using just four multiplexers which accepts two four bit buses and outputs one four bit bus out of those based on the select line 'a' or 'b'. This implementation is shown in the figure 3 that follows.



5. Implementation using vhdl

the hdl implementation of this modification is also very basic. Hdl (hardware description language) like vhdl and verilog implement logic in the form of look up tables. Where in the logic is stored in the form of table with the binary value for each variable as input and a particular output for each of the permutation that exists. The code for vhdl implementation of the modified logic is as shown in the figure below.

a function has been written which takes the original block as an argument and then transforms it based on the decision bits to convert it into the modified block. The "transform" function performs this task of conversion of the block to modified block. It takes in the arguments block, and the bits a and b and returns the modified block in the form of temp. It must be equated to block where the function is being called so that block gets the modified value.

similar to the hardware implementation, the indexes of the stored array from where the 'a' and 'b' bits are being extracted must be incremented so that they point to the next locations. By the time the index reaches 126^{th} and 127^{th} positions, the new block containing decision bits reaches the receiver and the indexes are set to the 0^{th} and 1^{st} bits of this newly received block.

6. Conclusion

Given the number of attempts that have been made during recent times to crack the aes encrypted data, it can be said that this algorithm which adds an extra layer of encryption to the process, makes the task ever more difficult. The modified algorithm boasts of several strengths :

- "diffusion" in the block to state encryption process provides more security.
- For each "decision bit block" that needs to be sent as data, we can encrypt the next 64 blocks of data, which is not that bad of a trade-off.
- The brute force attacks that are performed to extract data will be more difficult to execute.
- Given the complexity of the aes encryption algorithm hardware, the fairly simple hardware implementation of the modified algoritm would not be a hinderance on implementation.
- The use of exactly the same hardware at both, the sender as well as the reciever also aides to ease the implementation.
- The throughput of the modified algorithm should not be a concern as the time taken by the modified unit will always be lesser than that taken by the encryption algorithm that follows.

7. References

[1] fips 197, “advanced encryption standard (aes)”, november 26, 2001.

[2] hoang trang and nguyen van loi, “an efficient fpga implementation of the advanced encryption standard algorithm” ieee conference 2012.

[3] daemen, joan; rijmen, vincent (9/04/2003),"aes proposal: rijndael", national institute of standards and technology.

[4] announcing the “advanced encryption standard(aes)",federal information processing standards publication 197, united states national institute of standards and technology (nist), november 26, 2001.