PROJECT REPORT

ON

FPGA IMPLEMENTION OF AES CRYPTOGRAPHIC ALGORITHM

Submitted To:
Dr. Gaurav Trivedi

Mentored By:
Miss. Meenali Janveja

Submitted By:
Kautilya Joshi (IIIT NAGPUR)
Utkarsh Bhiogade (IIIT NAGPUR)

ACKNOWLEDGEMENT

We express our deep sense of gratitude to professor, ***Dr.Gaurav Trivedi***, and our Mentor ***Miss.Meenali Janveja*** for there valuable guidance and encouragement. We would also like to express our gratitude to all those who were involved directly or indirectly with the completion of this project.

Contents

1 AES Algorithm

1.1 About The Paper

In this paper[1], using Advanced Encryption Standard (AES) Algorithm, the author has proposed an optimized key expansion module for secure transmission of ECG signals. AES is a symmetric, non-fiestal block cipher cryptographic algorithm that encrypts and decrypts the data block on 128 bits using different key sizes. Based on the block sizes, the number of rounds of encryption and decryption operations and the number of subkeys generated from the main key differs. In the proposed algorithm the subkey generation is altered to speed up the process of generating subkeys from the main key. To decrease the time consumption for obtaining the subkeys required to convert the plain text into non-intelligent cipher text, a pipelined architecture was implemented. By this proposed algorithm, the author found that there was 50% reduction in the time taken to generate all the subkeys.

1.2 AES Algorithm

AES serves as a most important cryptographic algorithm which satisfies the most important security goals (Confidentiality, Integrity, Availability) for the secure communication over an unsecure communication channel. The key lengths which are used for encryption and decryption of data in AES is associated with the number of rounds used in the AES architecture. The size of data block is same for all version of AES algorithm. Each different subkey which is generated from the single main key is used in each round of encryption and decryption.

1.2.1 AES Encryption and Decryption

The steps involved in the algorithm of Encryption and Decryption are as follows:

1. Sub-byte/Inverse Sub-byte
   In this step, each byte of plain text array is replaced with a SubByte using 8-bit Substitution box.It is replaced in hexadecimal byte using substitution table/Inv Subtution Table for Encrytion/Decryption Process respectively. It is the only Non-linear process in the algorithm.

2. The Shift Row Step
   In this step the rows in the state arrays are shifted left cyclically by a certain offset. The first row remains unchanged as the offset in zero. The second, third and fourth row are shifted by offset of 1,2 and 3 respectively. The columns of output which is obtained after this step is composed of bytes from each column from the input state.In case of Decryption, the rows are shifted cyclically right with respective offset value.The importance of this step is to avoid the columns being encrypted independently, in which case AES degenerates into four independent block ciphers.

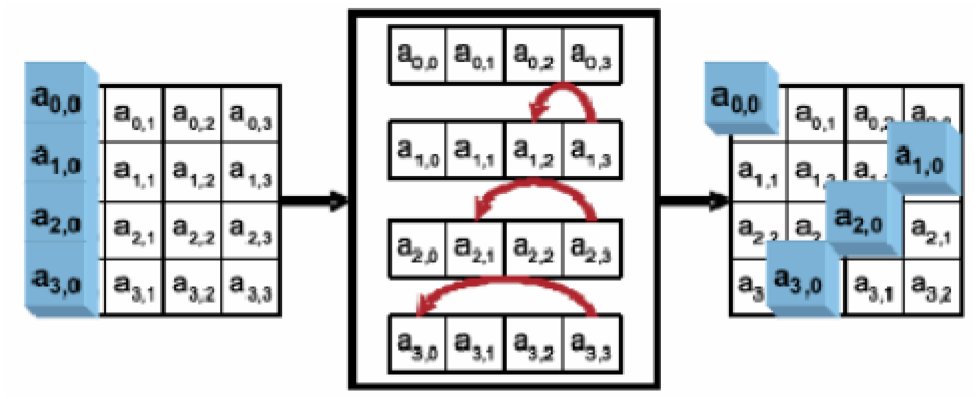


Figure 1: Shift Row

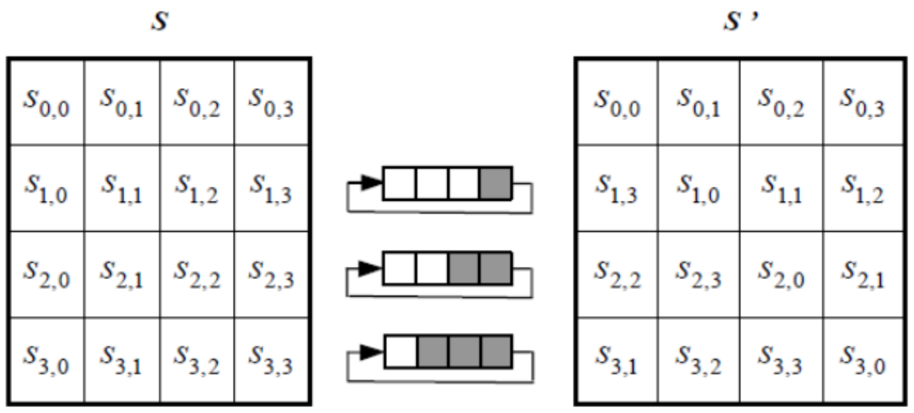


Figure 2: Inverse Shift Row

3. Mix columns/ Inv Mix columns

   In this Step, column level operations take place, a simple way to obtain is Matrix Multiplication

   For Mix Column operation

$$\begin{bmatrix} S'1 \\ S'2 \\ S'3 \\ S'4 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S1 \\ S2 \\ S3 \\ S4 \end{bmatrix}$$

   For Inv Mix Column operation

$$\begin{bmatrix} S'1 \\ S'2 \\ S'3 \\ S'4 \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S1 \\ S2 \\ S3 \\ S4 \end{bmatrix}$$

   Matrix multiplication is composed of multipliaction and addition of entries. Here adding is simply XOR

4. Add Round Key

   In this step,the subkey is combined with each state. Then this subkey (Round Key) is simply added to the State array by a bitwise XOR operation.

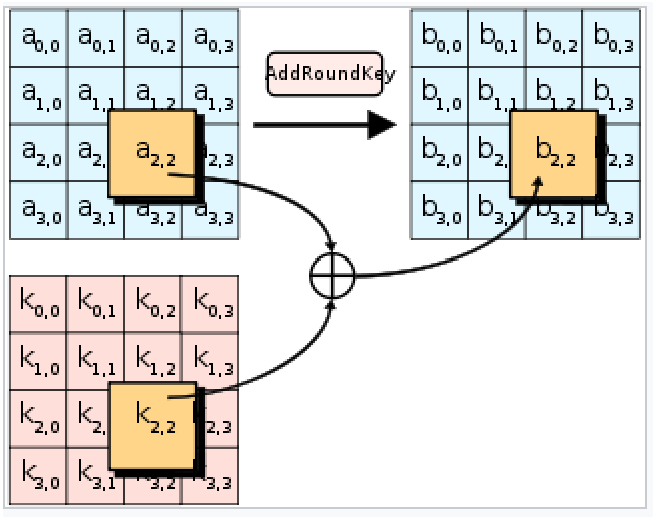


Figure 3: Add Round Key

1.2.2 Existing Architecture for Key-Expansion

In this process round key are generated which are used in each round.If the number of rounds is N_r, then the Key-Expansion routine creates $N_r + 1$ round keys (128 bit) from single cipher key (128 bit). It creates round keys word by word, where a word is an array of four bytes. In the Existing Architecture $4(N_r + 1)$ words are obtained from the intial main key, W_0,W_1,W_2,W_3. From Fig. 4, it is clear that each word created purely depends on the word at the left and the word at the top if it i $\neq 4$.
$Ifi = 4$ditemize , then the current word also depends on a temporary word, which is the result of subword and rotwr- n $W_{i1}$and XORing the result with R con. Where N_r = number of rounds and i = word number in key. Temporary word = (Rotword(W_{i1})) Rcon

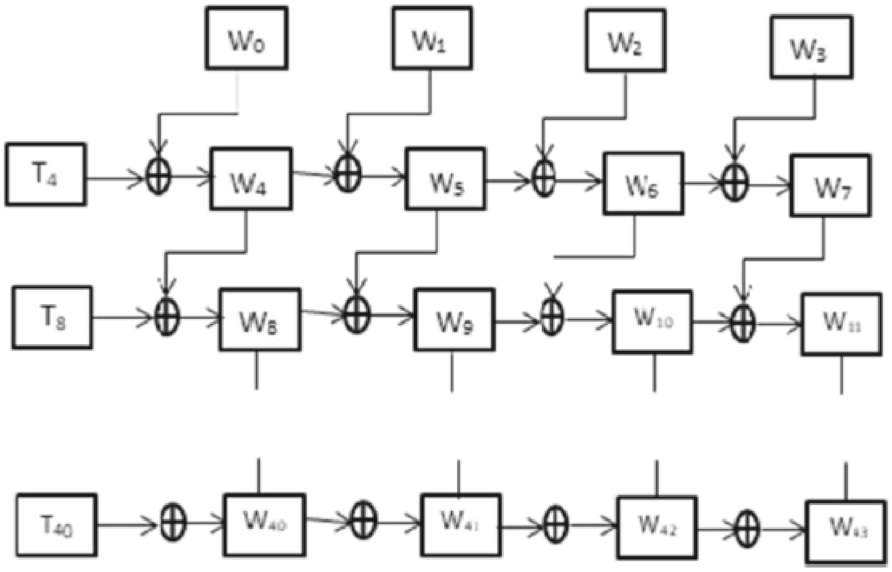


Figure 4: Existing Architecture

The table below shows the R constanst (Rcon) values for diffrent rounds in AES-128

Round	Rcon	Round	Rcon
1	$(01000000)_{16}$	6	$(20000000)_{16}$
2	$(02000000)_{16}$	7	$(40000000)_{16}$
3	$(04000000)_{16}$	8	$(80000000)_{16}$
4	$(08000000)_{16}$	9	$(1B000000)_{16}$
5	$(10000000)_{16}$	10	$(36000000)_{16}$

Table 1: Rcon values for diffrent rounds in AES-128

1.2.3 Proposed architecture

In the existing architecture, each word in the current subkey depends on the previous subkey. Therefore all the subkeys are generated in sequential manner, one after the other. Generation of last subkey takes place after the completion of the generation of all previous subkeys.This process consumes lots of time to generate all the subkeys from main key, since all the subkeys cannot be determined simultaneously. This is the major drawback of the existing architecture of Key expansion in AES algorithm.
In the proposed architecture, the time consumption was significantly reduced. Here, the whole architecture was splitted into two block which were executed in parallel. In the new architecture too, the process of generation of temporary words remains same as that of existing arrchitecture.
In this architecture, sixth subkey is generated from the main key instead of fifth key, and because of this modification, sixth key will be generated at the same time when first subkey is generated. Hence, both blocks can generate subkeys in parallel.

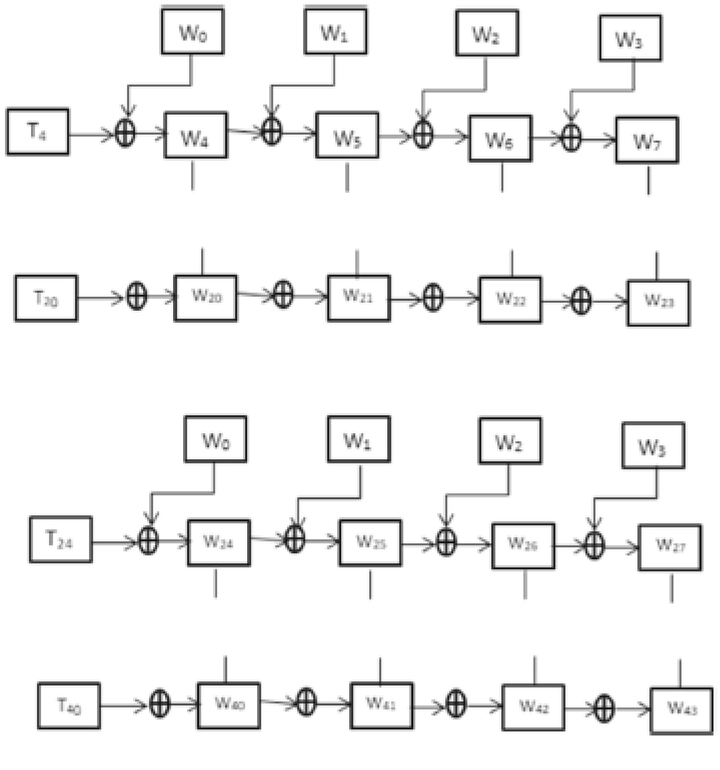


Figure 5: Proposed Architecture

It is clear that in the proposed architecture t_{24} is generated from W_2, where as in existing architecture t_{24} is generated from W_{23}. Also sixth subkey is fully dependent on the main key instead of fifth subkey.

1.3 Code explaination

CODE EXPLANATION:
A) ENCRYPTION
• An sbox module is created which takes the input as an 8 bit value where the upper nibble is the row index and lower nibble is column index. The output of this module is the value to be substituted in the subbyte transformation.
• In Subbytes module each byte of the plain text given is substituted using sbox.
• A Shiftrow module is created to perform the shift row operation as described above.
• A mixcolumn module is created to perform the mix column function as described above. A function is defined in this module in which each bit of the output can be expressed in terms of the input bits.
• A KeyGen module is created to perform the above described key generation process.

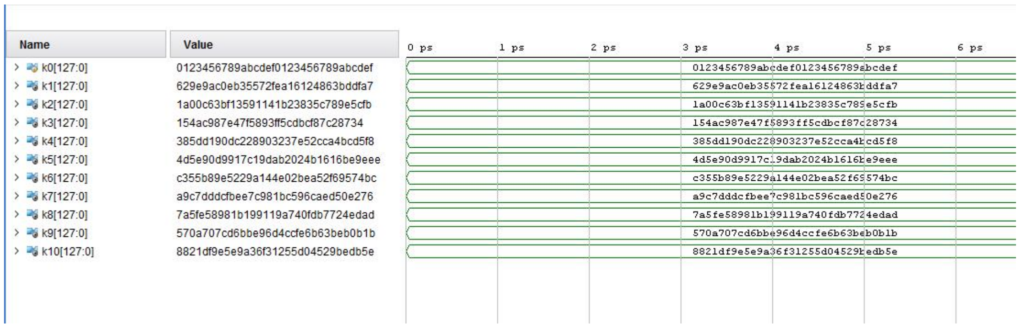
• A rounds module is created in which the following operations are done sequentially:
I) Subbytes transformation to the input data by using SubBytes module.
II) The rows are shifted using ShiftRow module.
III) Mixcolumn operation is done based on mixcolumn module.
IV) The output of rounds module is obtained by XORing the output of (III) with the respective key input.

• A roundlast module is created in which the following operations are done sequentially:
I) Subbyte transformation to the input data by using Subbyte module.
II) The rows are shifted using Shiftrow module.
III) The output of rounds module is obtained by XORing the output of (III) with the respective key input.
• A cipher module is created which performs the following
a) All the subkeys are produced using KeyGen module.
b) The rounds module is used in which the first round has a input which is obtained by XORing the input with the secret key. For the rest of the rounds the input will be outputs of the previous rounds.
c) For the last round roundlast module is used and output of this module is the output of cipher module.

The input image file is converted into a text file (input.txt) using a MATLAB code .
• A read module is created to read input.txt file and store it in an array.
• A aestop module is created which reads the given input file data by the read module and the output of this module is given as the input to the cipher module.
• A Test bench is created (tb_encryption) for this top module where the secret key and the input data are given and the produced output is stored in a text file (encrypted.txt).

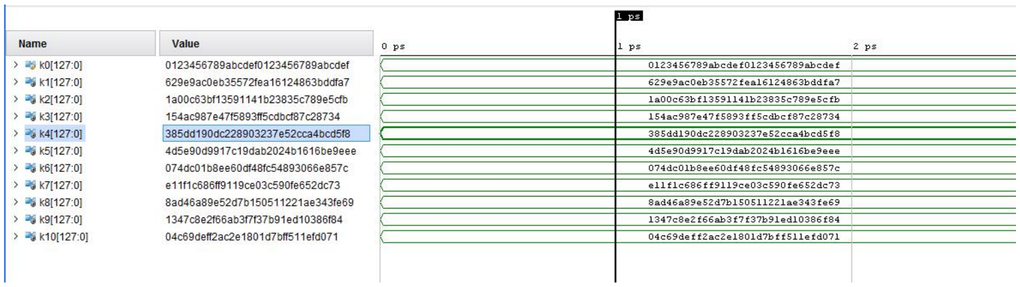
B) MODIFIED AES:
• In the modified AES algorithm , the image encryption and decryption is done in the same way of image encryption and decryption algorithm of standard AES. Here only the key generation process varies where round 6 uses secret key as the parent key.

1.4 Result



Name	Value
k0[127:0]	0123456789abcdef0123456789abcdef
k1[127:0]	629e9ac0eb35572fea16124863bddfa7
k2[127:0]	1a00c63bf13591141b23835c789e5cfb
k3[127:0]	154ac987e47f5893ff5cdbcf87c28734
k4[127:0]	385dd190dc228903237e52cca4bcd5f8
k5[127:0]	4d5e90d9917c19dab2024b1616be9eee
k6[127:0]	c355b89e5229a144e02bea52f69574bc
k7[127:0]	a9c7dddcfbee7c981bc596caed50e276
k8[127:0]	7a5fe58981b199119a740fdb7724edad
k9[127:0]	570a707cd6bbe96d4ccfe6b63beb0b1b
k10[127:0]	8821df9e5e9a36f31255d04529bedb5e

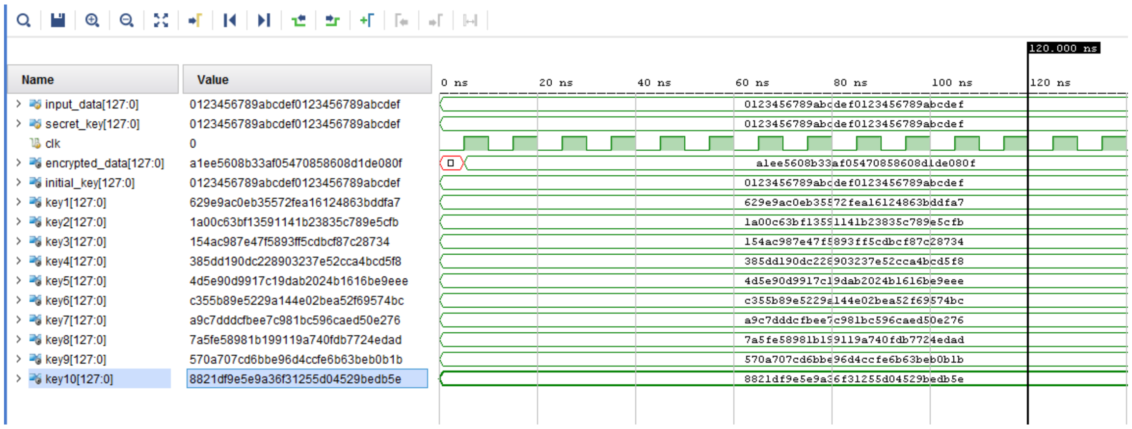
(a) Key Generation with Existing Architecture



Name	Value
k0[127:0]	0123456789abcdef0123456789abcdef
k1[127:0]	629e9ac0eb35572fea16124863bddfa7
k2[127:0]	1a00c63bf13591141b23835c789e5cfb
k3[127:0]	154ac987e47f5893ff5cdbcf87c28734
k4[127:0]	385dd190dc228903237e52cca4bcd5f8
k5[127:0]	4d5e90d9917c19dab2024b1616be9eee
k6[127:0]	074dc01b8ee60df48fc54893066e857c
k7[127:0]	e11f1c686ff9119ce03c590fe652dc73
k8[127:0]	8ad46a89e52d7b150511221ae343fe69
k9[127:0]	1347c8e2f66ab3f7f37b91ed10386f84
k10[127:0]	04c69deff2ac2e1801d7bff511efd071

(b) Key Generation with Modified Architecture

Figure 6: Simulation Results of Key Generation



Name	Value
input_data[127:0]	0123456789abcdef0123456789abcdef
secret_key[127:0]	0123456789abcdef0123456789abcdef
clk	0
encrypted_data[127:0]	a1ee5608b33af05470858608d1de080f
initial_key[127:0]	0123456789abcdef0123456789abcdef
key1[127:0]	629e9ac0eb35572fea16124863bddfa7
key2[127:0]	1a00c63bf13591141b23835c789e5cfb
key3[127:0]	154ac987e47f5893ff5cdbcf87c28734
key4[127:0]	385dd190dc228903237e52cca4bcd5f8
key5[127:0]	4d5e90d9917c19dab2024b1616be9eee
key6[127:0]	c355b89e5229a144e02bea52f69574bc
key7[127:0]	a9c7dddcfbee7c981bc596caed50e276
key8[127:0]	7a5fe58981b199119a740fdb7724edad
key9[127:0]	570a707cd6bbe96d4ccfe6b63beb0b1b
key10[127:0]	8821df9e5e9a36f31255d04529bedb5e

Figure 7: Simulation Results of Encryption

1.5 References

1. FPGA implementation of an optimized key expansion module of AES algorithm for secure transmission of personal ECG signals. *Thanikodi Manoj Kumar, Palanivel Karthigaikumar*

2. `https://en.wikipedia.org/wiki/Advanced_Encryption_Standard`

3. Cryptography and Network Security by Behrouz A. Forouzan

4. `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf`