Design and synthesis of Dual Key based AES Encryption

Abhiram.L.S, Gowrav.L, Punith Kumar.H.L
Department of Electronics and Communication
M.S.Ramaiah Institute of Technology
Bangalore, India

Sriroop.B.K , Manjunath.C.Lakkannavar
Department of Electronics and Communication
M.S.Ramaiah Institute of Technology
Bangalore, India

***Abstract*—** Key aspect of communication is security. Encryption of information to make it inaccessible to unauthorized recipients is a main area in network security. AES encryption is one of the most secure encryption algorithms. But with improving technology, attacks which can be used for easy cryptanalysis of encrypted information have been developed. One of the main reasons for vulnerability of AES encryption algorithm is the use of static S-Boxes. Also with attempts made to tap the key, AES has become more vulnerable to cryptanalysis attacks. Static S-Boxes make it very easy for reverse engineering which form the basis of Super S-Box attack. Hence there have been a lot of proposals for generation of S-Boxes which are key dependent. Majority of those algorithms involve probabilistic methods which are very complicated for hardware implementation. In this paper, we present a synthesizable algorithm which involves the use of conventional bitwise operations for generation of key dependent S-Boxes. Also a dual key based AES is presented in the paper which is FPGA implementable. The algorithm is reliable in terms of security and also suitable for hardware implementation. Mathematical analyses have been carried out on the algorithm to compute the reliability.

Keywords— AES, S-Box, Dual Key

I. INTRODUCTION

Encryption has an important role in data protection with increase in global wide data communication. Encryption makes sense when data packets are transferred using open channels, which can be reached by other devices or people, leading to failure in integrity security and anonymity of the communicating entities.

This paper begins with a brief introduction to the Advanced Encryption Standard, the Sub-Byte and Inverse Sub-Byte transformation, a short discussion on the previous implementations of the SBOX and its vulnerability to cryptanalysis, stipulating the need for dynamic SBOX as an alternative solution followed by the limitations in the existing Dynamic SBOX [2] generation techniques and introducing the new methodology of a key dependent pseudo dynamic SBOX and Dual key based AES.

The core component of any Block Cipher is the non-linearity hidden in the SBOX. Hence building a stronger and a key dependent dynamic SBOX basically strengthens the very base of encryption standards. Also the use of dual key and additional Shift column transformation enhances the security.

The implementation of Pseudo dynamic S-Box is highlighted in this section [1]. The implementation of Dual Key SBOX on a FPGA platform is sketched in Section-3 along with design algorithm and modifications made to accommodate the conventional static SBOX in the design alongside the proposed approach.

The simulation results are briefly covered in the Section-4 titled “System Level and Design Validation”.

II. SYSTEM AND IC ARCHITECTURE

An Encryption system contains set of transformations that convert plain text into cipher text. In the block cipher system, plain text is divided into blocks on which the encryption algorithm is applied to create cipher text.

The block cipher systems are based on two general principles: Diffusion and Confusion. In Diffusion principle, each character of plain text converts into one of many other characters. However, in Confusion principle, the characters are reordered. Usually in both principles, round repetition is used to create a cipher text. Repeating a single round contributes to cipher’s complexity.

i. ADVANCED ENCRYPTION STANDARD (AES)

This standard specifies the Rijndael algorithm [2]. It has a fixed block size of 128 bits and a key length of 128, 192 or 256 bits. It generates its round key from an input key using the Key Expansion function. The AES operates on a 4x4 array of bytes which is called a state. The state undergoes 4 transformations which are namely the Shift Row, Mix Column, Add Round Key and Sub Byte transformation.

a) **Shift Row Transformation**: This is done by cyclically shifting the rows in the array with different offsets. It is carried out by linear diffusion process operating on individual rows. Depending on the row location, offset of right shift varies from zero to three bytes.
b) **Mix Column Transformation**: This transform is a column mixing operation, where the bytes in the new column are a function of the 4 bytes of a column in the state array. Matrix multiplication is done over Galois Field. Column vector is multiplied with a fixed matrix where the bytes are treated as a polynomials rather than numbers.

 978-1-4799-6546-5/14/$31.00

c) **Add Round Key:** This transformation involves a bitwise XOR operation between the state array and the resulting Round Key that is output of the Key Expansion algorithm.

d) **Sub Byte Transformation**: Sub Byte transformation is a non-linear byte substitution process where each byte in the state array is replaced with another from a lookup table called an S-Box which is obtained by carrying out a process involving multiplicative inverse and affine transformation.

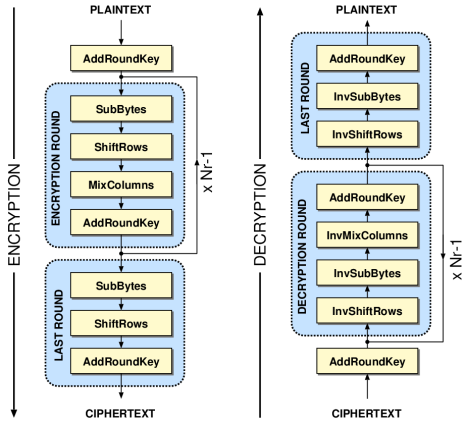

Fig. 2.1 AES encryption & decryption flowchart.

The above operations are repeated 10 to 14 times depending on the block size. The decryption scheme retraces the steps backward to obtain the plain text.

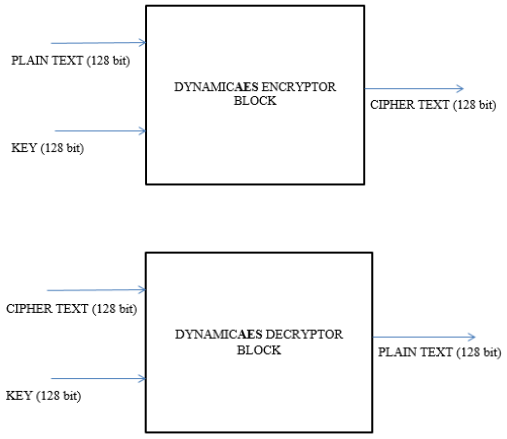

Fig 2.2 Black Box representation of AES Encryption and Decryption

III. PHYSICAL DESIGN METHODOLOGY

i. VULNERABILITIES IN THE EXISTING S-BOX TECHNOLOGIES:

As mentioned earlier, Substitution is a nonlinear-transformation which performs diffusion of characters. A nonlinear transformation is essential for every modern encryption algorithm and is proved to be a strong cryptographic primitive against linear and differential cryptanalysis. Nonlinear transformations are implemented as lookup tables (S-Boxes).

There are two ways of generation of S-Bytes.

1. Look up tables
2. Dynamic generation of S-Bytes

The implementation of S-Box as a look up table is shown in Fig 3.1. The upper nibble of the byte forms the input to the column and the lower nibble forms the input to the row. The corresponding S-box generated above is given below.

63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fig. 3.1 Tables of values for a Static S-BOX

The other method for generation of S-Box is dynamic generation. The multiplicative inverse of the byte in GF (2^8) is computed and affine transform is applied on it to obtain the S-Byte. The architecture for computation of S-Byte is shown in Fig 3.2 and 3.3.

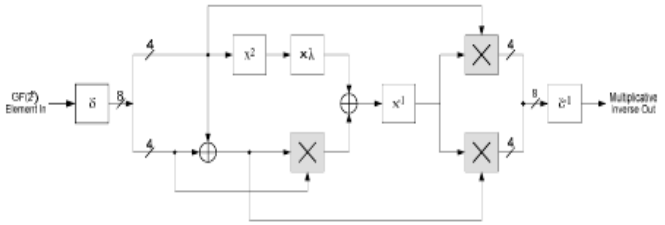

Fig 3.2 Multiplicative inverse in a Galois Field

In matrix form, the affine transformation element of the S-box can be expressed as:

$$\begin{pmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Fig. 3.3 affine transformation element of the S-box

This method of generating substitution bytes would keep the output byte constant always for a given input byte. That is, the

only parameter influencing the output byte would be the input byte. This makes reverse engineering easy for crypt-analyzers. Some doubts persist about the level of security provided by static S-Boxes due to its simple algebraic structure. Hence it is advisable to make the Substitution bytes dependent on both the input byte and also the key.

Super SBOX attack uses divide and conquer policy to allow the hacker to enter deep into the algorithm and access parts of it. This would reveal the cipher key slowly. This is an algorithm which is very simple to implement and provides high efficiency. This major hit on an AES Encryption scheme was the driving force behind the Key based SBOXs in Dynamic AES algorithms.

ii. PSEUDO DYNAMIC SBOX:

Developments on the algorithm have suggested probabilistic techniques for key based S-Box generation are highly efficient but extremely complex to understand and implement. Hence the algorithm proposed in this paper is equally efficient in terms of complexity of reverse engineering but very simple to understand and implement.

The proposed Pseudo Dynamic SBOX [6] generation algorithms used in generation of add round keys as well as the Non linearity in mainstream data substitution are as follows:

The procedure involves computation of *mod_val* which is the remainder obtained when first 8 bits of the key are divided by 8. The number 8 is selected because the remainder obtained should be within 0 to 7.

```
mod_val=(key[127:120])%8

        for: i=0 to 15 loop
                temp[i]=key[(8i)+mod_val

        for: j=0 to 7 loop
                offset[j]=temp[2i]+temp[2i+1]

dyn_subbyte = static_subbyte ^ offset
```

Fig. 3.4 Algorithm for generating Pseudo-Dynamic SBOX

This $(mod\_val)^{th}$ bit of every byte is selected and are added with each other. The resultant offset of this XOR operation is added with the sub-byte obtained by conventional method. This result gives the key based sub-byte.
This efficiency of the algorithm is comparable with the probabilistic method of generating S-Box but easier to implement on hardware. The limitation of the algorithm is that the output is again dependent on input and key. It is not a variable in time.

iii. PSEUDO DYNAMIC DUAL KEY BASED AES:

The proposed algorithm involves the use of two keys. One key is generated within the system (called as System Key) and the other key is provided by the user (called as User Key). Also the new algorithm involves the use of a new transformation called SHIFT COLUMNS. The transformations SHIFT ROWS and SHIFT COLUMNS are also made key based. The randomly generated key is used for SUB-BYTES and ADD ROUND KEY transformation whereas the USER DEFINED KEY is used for SHIFT ROWS AND SHIFT COLUMNS. The algorithm used for AES is as shown below:

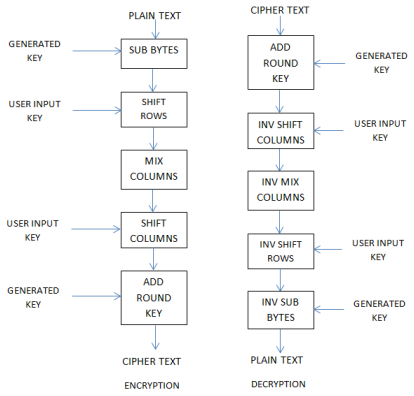

Fig 3.5: Block Diagram of dual key based AES

The SUB-BYTES transformation involves the transposition of S-Box based on the round key. The algorithm used for transposition is as shown in fig 3.4

The procedure involves computation of *mod_val* which is the remainder obtained when first 8 bits of the User key are divided by 8. The number 8 is selected because the remainder obtained should be within 0 to 7. This $(mod\_val)^{th}$ bit of every byte is selected and are added with each other. The resultant offset of this XOR operation is added with the sub-byte obtained by conventional method. This result gives the system key based sub-byte. This efficiency of the algorithm is comparable with the probabilistic method of generating S-Box but easier to implement on hardware. The limitation of the algorithm is that the output is again dependent on input and key. It is not a variable in time.

The Shift Rows and Shift columns involves shifting the elements of the state matrix by a 2 bit offset value. The 2 bit offset is generated from the System key as stated below:

Step 1: $R = (System\_Key)\ \%8$
Resultant is a value between 0 and 7

Step 2: Concatenate the R^{th} bit of every byte to obtain 16 bit offset K

Step 3: Generate the shifting offset as

$$Off = \{K_{15},K_7\}\hat{}\{K_{14},K_6\}\hat{}\{K_{13},K_5\}\hat{}\{K_{12},K_4\}\hat{}\{K_{11},K_3\}\hat{}\{K_{10},K_2\}\hat{}\{K_9,K_1\}\hat{}\{K_8,K_0\}$$

The Rows are shifted as follows:

Shift the i^{th} row of the state matrix $Off\hat{}i$ times. Similarly shift the columns during the next transformation.

For decryption it is essential to know the System key. Hence it has to be transmitted. But since it consumes a lot of bandwidth, the offset generated is transmitted by merging it with the encrypted text.

R^{th} bit position and the S^{th} bit position, where S=!R, consists of the higher and lower *Off* bits.

IV. SYSTEM-LEVEL DESIGN VALIDATION

The Following design was first implemented using XILINX ISE and Vivado EDA tools for the FPGA fabric with a good performance on Spartan 3E and Virtex-5 chips.

The Current design is under carefully scrutiny for any design changes to obtain an optimised code for the ASIC implementation which is being carried out in Cadence Incisive as a digital EDA.

The Pseudo Dynamic AES has an input signal to enable or disable the use of static SBOX in the application, which was an added bonus for quick verification of the performance changes in the two methods of encryption thus at the same time allowing the legacy mode in AES which doesn't support the new approach. The following Figures represent the AES chip design under various divisions of code verification and synthesis.

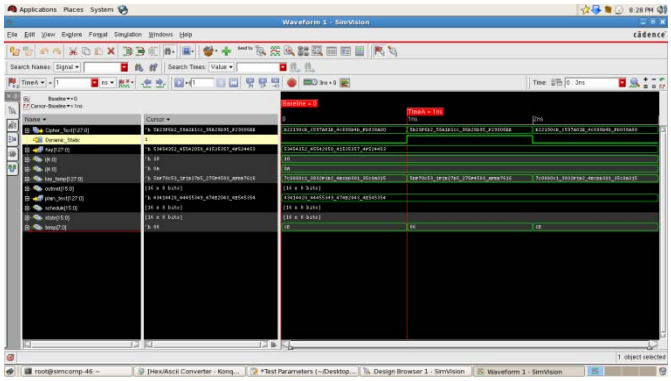

Fig 4.1 Simulation on Cadence SimVision

V. DESIGN RESULTS

The level of security provided by AES is usually measured in terms of time taken for cryptanalysis. The higher the time taken for cryptanalysis, the higher is the level of security offered.

The time taken for brute forcing increases by the order of 5 as justified below:

- It is calculated that time taken for brute forcing using conventional AES=2.9e+21 years, on a 3.7GHz processor that completes one combination in 1 clock cycle;
- Number of rounds of Encryption=10;
- 2^{128} combinations of key have to be tried since it is randomly generated.
- Also 4 different offsets have to be tried every round. That is 40 combinations of offsets for shift columns have to be tried.
- For every round 2^8 different offsets for Sub-bytes have to be tried. Thus a total of 2^8*10 combinations will be tired.
- Totally brute forcing has to be done 2^{128}*40*2^8*10 times.
- That is 3.4e+43 combinations will be tried.
- If each combination works out in 1 clock cycle on a 3.7GHz processor, total time taken for brute forcing would be 3e+26 years.
- To complete the processing in 1 year 3e+26 computers would be required.

The performance parameters are mentioned in the table below:

Table 5.1: Comparison of synthesis results between conventional AES encryption and Dual-Key based AES encryption

	Conventional	Dual-Key based
Combinational Delay(ns)	24.664	38.219
Logic slices used	6%	12%

Table 5.2: Comparison of synthesis results between conventional AES decryption and Dual-Key based AES decryption

	Conventional	Dual-Key based
Combinational Delay(ns)	58.305	94.079
Logic slices used	8%	15%

In the present paper, performance and usage of logic slices of 128 bit AES have been measured in hardware implementation. Results as shown in the above table, indicate that the combinational delay increases by a factor of 2 and the number of logic slices used also increase by a factor of 1.6. The increase in the number of logic slices can be tolerated because the number of logic slices available is too large. But the trade-off between delay and security continues to exist. The trade-off decision between level of security, performance and area is left for designers or application engineers implementing the AES algorithm

Further implementation of this project is being carried out for few realistic embedded applications such as AES secured Bluetooth headsets in 2G cell phone communication.

VI. REFERENCES

[1] [1] Abhiram.L.S, Gowrav.L, Punith Kumar.H.L, Sriroop.B.K, Manjunath C Lakkannavar, "Design and synthesis of Pseudo dynamic S-BOX based AES encryption", National Conference on Electrical and Electronics Engineering, HKBK college of Engineering, 2014

[2] [2] "Advanced Encryption Standard, Federal Information Processing Standards 197", National Institute of Standards and Technology, November 2001

[3] [3] A.F Webster and S.E Travares, "On The Design of S-boxes," Queen's university Kingston, Springer-verlag, Canada1998 .

[4] [4] Muhammad Asim, "Efficient and Simple Method for Designing Chaotic S-boxes, "Electronic and Telecommunications Research Journal, University of Technology Petronas, Malaysia February 2008.

[5] [5] Xinmiao Zhang and Keshab K. Parhi, "On the Optimum Constructions of Composite Field for the AES Algorithm", IEEE, VOL. 53, NO. 10, OCTOBER 2006

[6] [6] F. Fahmy and G. Salama, "A proposal for Key-dependant AES, " 3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications (SETIT), TUNISIA March 2005.

[7] [7] Eltayeb Salih, "An Optimized Implementation of S-box using Residues of Prime Numbers," International Journal of Computer Science and Network Security (Vol.8), April 2008.Science, 1989.

[8] [8] Joan Daemen and Vincent Rijmen."Two-Round AES Differential". Cryptology ePrint Archive, Report 2006/039, 2006.