AES 128 Bit Decryption

Le Ngoc Ha*, Le Duc Manh†, Senura Hansaja Wanasekara‡
Ha Noi University of Sciences and Technology
Email: *ha.ln182916@sis.hust.edu.vn, †manh.ld182927@sis.hust.edu.vn, ‡wanasekara.190185@sis.hust.edu.vn

***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. In this paper, a 128 bit AES Decryption by using Rijndael algorithm (Advanced Encryption Standard algorithm)[1] is been made into a synthesizable using Verilog code which can be easily implemented on to FPGA. The algorithm is composed of three main parts: cipher, inverse cipher[2] and Key Expansion. Cipher converts data to an unintelligible form called plaintext. Key Expansion generates a Key schedule that is used in cipher and inverse cipher procedure. Cipher and inverse cipher are composed of special number of 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[3] and Add Round Key.**

***Index Terms*—AES, AES 128 bit, AES descryption, AES algorithms**

I. Introduction

Advanced Encryption Standard (AES), also known as its original name Rijndael is an encryption standard for encryption, electronic data transmission currently used Internet or otherwise. it is a country established by the united nations institute of standards and technology in 2001, now used worldwide. It inherits from its predecessor, Data Encryption Standard (DES) [4] established in 1977 becoming the first publicly top secret data algorithm and approved by the National Security Agency (NSA) in 2002.

AES is a symmetric key algorithm, which means the same key is used for encryption and decryption. three Select an algorithm from the Rijndael family for encryption. The block size for all three algorithms is 128 bits, but within the key length. AES-128 has a 128-bit key. Similarly, AES-192 AES-256 has a key length of 192 and 256 bits, respectively. This article is organized as follows. This article is organized as follows. Sections III and IV provide a general overview of the algorithmic background and other similar algorithms used. Parts V and VI give the simulation results of AES 128 bit descryption algorithm.

II. Security

The AES-128 security[5], though very algebraically simple, would take 1.02 x 1018 years to break, against a brute-force attack which would render it secure against all such attacks since the output is a non-linear mathematical function of the input. Although practical side-channel attacks exist for AES, no empirical cryptanalytic attack against it has been discovered. The best reasonable attack against the cipher is a biclique attack, which knocks off a few bits from the key but would still take a million years to obtain the plaintext since it takes 2126.2 operations to obtain the 128-bit key. The computational security of an encryption cipher depends on the length of the key used. Longer keys cost exponentially more, in terms of computation cost as compared to shorter keys. To attack AES it would take 288 bits of data, which amounts to 38 trillion terabytes. The memory requirement of such an attack is greater the sum of all data stored on all electronic resources in the world as of 2018. The space complexity of AES has recently been increased to 253 bytes, which adds up to 9007 terabytes of memory power.

III. Related Work

A significant amount of work has been done lately regarding efficient AES implementation on FPGA. For example, the authors present a sequential implementation of the AES algorithm on FPGA[6] that results in a design that is well suited for small embedded applications. Implementation in and focuses on the design of low-power FPGA-based encryption schemes. These schemes try to achieve the best power results without compromising the throughput of the design. Similarly, the authors present an efficient implementation of the AES algorithm on FPGA that results in high throughput and it is well suited for applications requiring high speed and performance. Furthermore, the authors explore pipelining[2], sub-pipelining, and loop unrolling techniques to increase the frequency and throughput of AES implementation on FPGA. The "Maestro" paper has presented a compact hardware-software co-design of Advanced Encryption Standard (AES) on the field-programmable gate arrays (FPGA) designed for low-cost embedded systems. The design uses MicroBlaze, a soft-core processor from Xilinx. The computationally intensive operations of the AES are implemented in hardware for better speed.

IV. The AES Algorithm

AES is based on a design principle known as a substitution–permutation network, and is efficient in both software and hardware. Unlike its predecessor DES[4], AES does not use a Feistel network. The algorithm implemented in this paper takes 128-bit plaintext and a 128-bit key as inputs to give 128-bit ciphertext as output. The number of rounds varies depending on the key size. For a 128-bit key, we require 10 rounds of computation.

Table I
AES ROUNDS FOR KEY LENGTH

Key size (in bits)	Number of Rounds
128	10
192	12
256	14

The initial round consists of the Add Round Key transformation, followed by 9 rounds of four distinct transformations which are inverse Shift Row, inverse Substitute bytes, Add Round Key, and inverse Mix columns, which will be described subsequently. The final round, which is round 10, consists of the transformations inverse Shift Row, inverse Substitute Bytes, and Add Round Key. Each of these transformations takes 4 x 4 matrices and produces 4 x 4 matrices as outputs. The key generation module is used for the expansion of the 128-bit key into 11 128-bit keys, for each of the rounds, computed in the algorithm. A detailed description of each of the operations to be performed is given below.

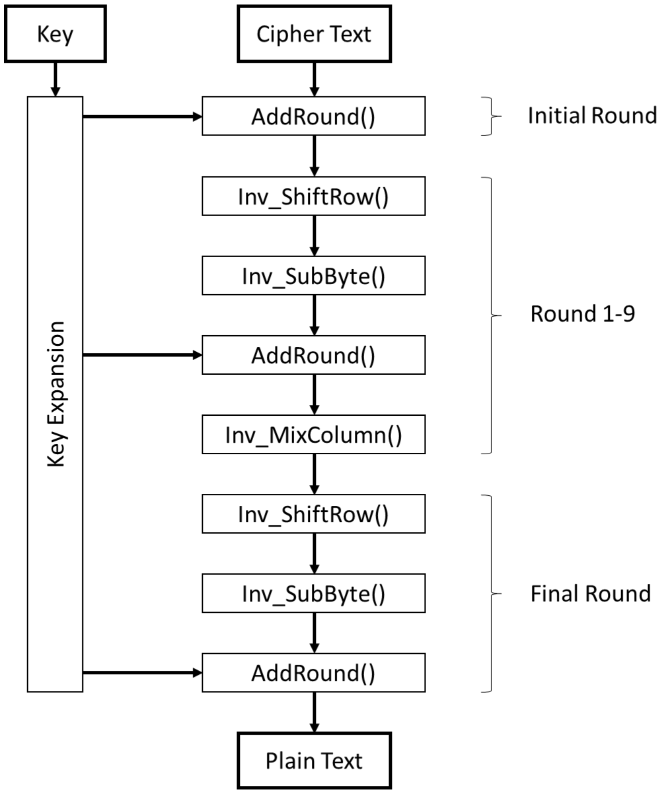
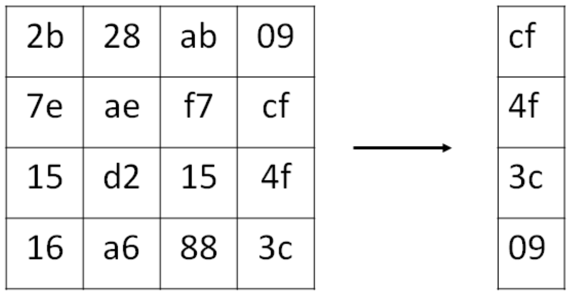


Figure 1. The AES 128-bits descryption algorithm

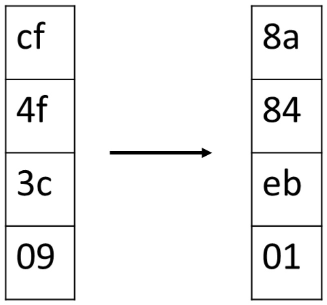
A. Key Expansion

The algorithm for generating the 10 rounds of the round key is as follows: The 4th column of the i-1 key is rotated such that each element is moved up one row.

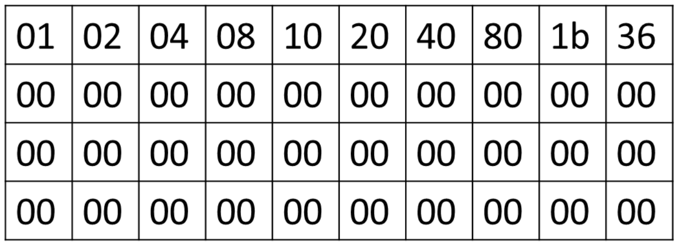
It then puts this result through a forwards Sub Box algorithm which replaces each 8 bits of the matrix with a corresponding



8-bit value from S-Box.

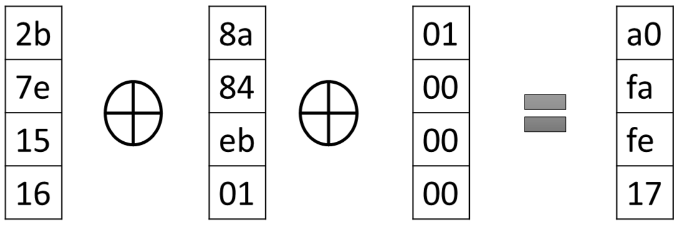


To generate the first column of the ith key, this result is XOR-ed with the first column of the i-1th key as well as a constant (Row constant or Rcon) which is dependent on i.



01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Figure 2. Rcon table



The second column is generated by XOR-ing the 1st column of the ith key with the second column of the i1 th key.

$$\begin{bmatrix} 28 \\ ae \\ d2 \\ a6 \end{bmatrix} \oplus \begin{bmatrix} a0 \\ fa \\ fe \\ 17 \end{bmatrix} = \begin{bmatrix} 88 \\ 54 \\ 2c \\ b1 \end{bmatrix}$$

This continues iteratively for the other two columns in order to generate the entire ith key.

$$\begin{bmatrix} 2b & 28 & ab & 09 \\ 7e & ae & f7 & cf \\ 15 & d2 & 15 & 4f \\ 16 & a6 & 88 & 3c \end{bmatrix} \longrightarrow \begin{bmatrix} a0 & 88 & 23 & 2a \\ fa & 54 & a3 & 6c \\ fe & 2c & 39 & 76 \\ 17 & b1 & 39 & 05 \end{bmatrix}$$

Additionally this entire process continues iteratively for generating all 10 keys. As a final note, all of these keys are stored statically once they have been computed initially as the ith key generated is required for the (10-i)th round of decryption.

B. Add Row Key

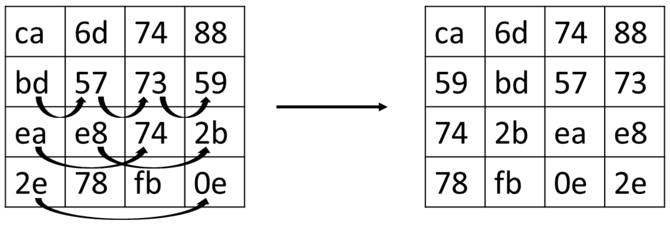
Performs XOR operation between the cipher text and intermediate expanded key corresponding to that particular iteration.

$$\begin{bmatrix} 1a & a4 & 95 & 3e \\ a9 & b9 & 4c & 3a \\ 13 & cd & 78 & 27 \\ 86 & f1 & 33 & a8 \end{bmatrix} \oplus \begin{bmatrix} d0 & c9 & e1 & b6 \\ 14 & ee & 3f & 63 \\ f9 & 25 & 0c & 0c \\ a8 & 89 & c8 & a6 \end{bmatrix} = \begin{bmatrix} ca & 6d & 74 & 88 \\ bd & 57 & 73 & 59 \\ ea & e8 & 74 & 2b \\ 2e & 78 & fb & 0e \end{bmatrix}$$

Figure 3. Add Round Key State

C. Inverse Shift Row

This step rotates each ith row by i elements right wise, as shown in the figure.



$$\begin{bmatrix} ca & 6d & 74 & 88 \\ bd & 57 & 73 & 59 \\ ea & e8 & 74 & 2b \\ 2e & 78 & fb & 0e \end{bmatrix} \longrightarrow \begin{bmatrix} ca & 6d & 74 & 88 \\ 59 & bd & 57 & 73 \\ 74 & 2b & ea & e8 \\ 78 & fb & 0e & 2e \end{bmatrix}$$

Figure 4. Inverse Shift Row State

D. Inverse Substitute Bytes

This step replaces each entry in the matrix from the corresponding entry in the inverse S-Box[2] as shown in figure.

$$\begin{bmatrix} ca & 6d & 74 & 88 \\ 59 & bd & 57 & 73 \\ 74 & 2b & ea & e8 \\ 78 & fb & 0e & 2e \end{bmatrix} \longrightarrow \begin{bmatrix} 10 & b3 & ca & 97 \\ 15 & cd & da & 8f \\ ca & 0b & bb & c8 \\ c1 & 63 & d7 & c3 \end{bmatrix}$$

Figure 5. Inverse Substitute Bytes State

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

Figure 6. Inverse S-box[7] table

E. Inverse Mix Columns

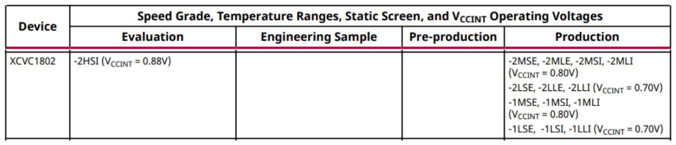
The Inverse Mix Columns operation performed by the Rijndael cipher, along with the shift-rows step, is the primary source of all the 10 rounds of diffusion in Rijndael. Each column is treated as a polynomial over Galois Field (28) and is then multiplied modulo x4 + 1 with a fixed inverse polynomial is c1(x) = 11x3 + 13x2 + 9x + 14. The Multiplication is done as shown below.

$$\begin{pmatrix} r0 \\ r1 \\ r2 \\ r3 \end{pmatrix} = \begin{pmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{pmatrix} \begin{pmatrix} a0 \\ a1 \\ a2 \\ a3 \end{pmatrix}$$

Figure 7. Inverse Mix Columns State

V. HARDWARE IMPLIMENTATION

Verilog has been used as the hardware description language and has been synthesized and implemented using Vivado[8] 2020.3. Simulation tools present of this software have been used to repeatedly test and debug the codes. The xcvc1802-viva1596-2MP-i-S FPGA board (Zedboard) has been used to realize this design.

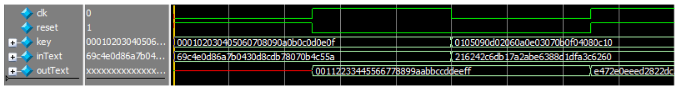


Device	Speed Grade, Temperature Ranges, Static Screen, and V_{CCINT} Operating Voltages			
	Evaluation	Engineering Sample	Pre-production	Production
XCVC1802	-2HSI (V_{CCINT} = 0.88V)			-2MSE, -2MLE, -2MSI, -2MLI (V_{CCINT} = 0.80V) -2LSE, -2LLE, -2LLI (V_{CCINT} = 0.70V) -1MSE, -1MSI, -1MLI (V_{CCINT} = 0.80V) -1LSE, -1LSI, -1LLI (V_{CCINT} = 0.70V)

Figure 8. Speed Grade Designations by Device

VI. Results

128-bit cipherText and a 128-bit key are given as input. The output is 128-bit plain text. The AES block was written from scratch, and is pretty fast, taking just 10 cycles for decrypting 128-bits of data.



```
clk      0
reset    1
key      00010203040506...   000102030405060708090a0b0c0d0e0f   0105090d02060a0e03070b0f04080c10
inText   69c4e0d86a7b04...   69c4e0d86a7b0430d8cdb78070b4c55a   216242c6db17a2abe6388d1dfa3c6260
outText  xxxxxxxxxxxxxx...   00112233445566778899aabbccddeeff   e472e0eeed2822dc
```

Figure 9. Simulation results for the AES 128-bit descryption algorithm, implemented in Verilog

Figure 10. Circuit simulation result for the AES 128-bit descryption algorithm, implemented in Vivado

After performing the analysis on the hardware and performing the simulation on the chip, we get the simulation circuit shown below.

VII. Discussion

This paper was successfully completed with the implementation of decryption for the AES algorithm. We implemented different sub-modules for the AES algorithm by using Verilog code. This implementation will be useful in wireless security like military communication and mobile telephony where there is a gayer emphasis on the speed of communication.

VIII. Conclusion

This paper was successfully completed with the implementation of AES algorithm on 128-bit message.The decrypted text are analyzed and proved to be correct. The encryption efficiency of the proposed AES algorithm was studied and met with satisfactory results.

References

[1] Dworkin, M. , Barker, E. , Nechvatal, J. , Foti, J. , Bassham, L. , Roback, E. and Dray, "Advanced encryption standard (aes)," accessed January 26, 2022. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.197

[2] A. M. S. R. Shrivathsa Bhargav, Larry Chen, "128-bit AES decryption," 2008, accessed January 26, 2022. [Online]. Available: http://www.cs.columbia.edu/~sedwards/classes/2008/4840/reports/AES.pdf

[3] "Rijndael mixcolumns," accessed on january, 26, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Rijndael_MixColumns

[4] S.-J. Han, H.-S. Oh, and J. Park, "The improved data encryption standard (des) algorithm," in *Proceedings of ISSSTA'95 International Symposium on Spread Spectrum Techniques and Applications*, vol. 3, 1996, pp. 1310–1314 vol.3.

[5] D. Crawford, "How does aes encryption work?" February 4, 2019, accessed on january, 26, 2022. [Online]. Available: https://proprivacy.com/guides/aes-encryption

[6] P. Rajasekar and Mangalam, "Dr.h. (2016) efficient fpga implementation of aes 128 bit for ieee 802.16e mobile wimax standards." pp. Circuits and Systems, 7, 371–380., accessed om January, 26, 2022. [Online]. Available: http://dx.doi.org/10.4236/cs.2016.74032

[7] "Rijndael s-box," accessed on january, 26, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Rijndael_S-box

[8] Xilinx, "Versal ai core series data sheet: Dc and ac switching characteristics," p. 17/77, accessed om January, 26, 2022. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds957-versal-ai-core.pdf