IMPLEMENTATION OF AES AND RSA ALGORITHM ON HARDWARE PLATFORM

Supriya Raygonda Rukade [1], Prof. S. D. Joshi[2]
Department of Electronics & Telecommunications,PVPIT, Bavdhan, Pune, India
Email:[1]supriyarukade89@gmail.com, [2] s.d.joshi@hotmail.com

Abstract: Data security is an important issue in computer networks and cryptographic algorithms are essential parts in network security. Reprogrammable devices such as Field Programmable Gate Arrays (FPGA) are highly attractive option for hardware implementation of cryptographic algorithm AES as they offer a quicker and more customizable solution. This paper proposes an efficient FPGA implementation of AES & RSA algorithm. We implement the AES and RSA algorithm on Xilinx Spartan-3 FPGA. The simulation for AES algorithm is done using VHDL language. Implementation of AES and RSA is done using Impulse C language. Advanced Encryption Standard (AES) on FPGA offers a better performance than any other cryptographic algorithms.

Keywords: AES, RSA, FPGA, Encryption, Decryption, Block cipher.

I. INTRODUCTION

Now a day sharing the information over internet is becoming a critical issue due to security problems. Hence more techniques are needed to protect the shared data in an unsecured channel. The present works focus on to secure the data i.e. cryptographic algorithm while transmitting in the network. Firstly the data which is to be transmitted from sender to receiver must be encrypted using the encrypted algorithm in cryptography. Secondly the encrypted data is converted into the original data by using decryption technique. Transmitting data or document can be done through these ways will be secured. Encryption has come up as a solution, and plays an important role in information security system.

For a long time, the Data Encryption Standard (DES) was considered as a standard for the symmetric key encryption. DES has a key length of 56 bits. However, this key length is currently considered small and can easily be broken [5]. For this reason, the National Institute of Standards and Technology (NIST) opened a formal call for algorithms in September 1997. A group of fifteen AES candidate algorithms were announced in August 1998.

In cryptography, the AES is also known as Rijndael. AES has a fixed block size of 128 bits and a key size of 128, 192 or 256 bits. The RSA algorithm was designed by Rivest, Shamir and Adleman in 1978. It is a public key cryptosystem. This algorithm is based on the difficulty of factorizing large numbers that have 2 and only 2 factors (Prime numbers). The system works on a public and private key system. The public key is made available to everyone. With this key a user can encrypt data but cannot decrypt it, the only authorized person who possesses the private key can decrypt it. This paper deals with an FPGA implementation of an AES encryptor/decryptor using an iterative looping approach with block and key size of 128 bits and RSA algorithm.

II. RELATED WORK

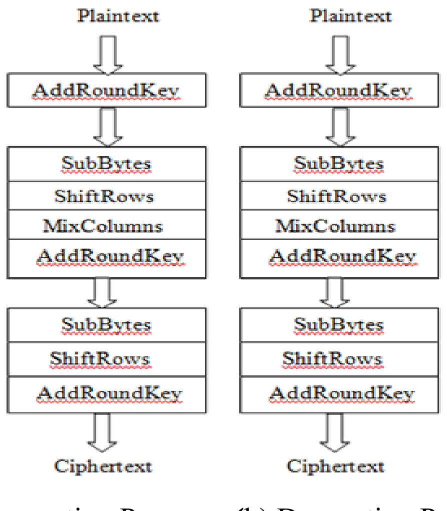
In August 2000, NIST selected five algorithms: Mars, RC6, Rijndael, Serpent and Twofish as the final competitors [1]. These algorithms were subject to further analysis prior to the selection of the best algorithm for the AES. Finally, on October 2, 2000, NIST announced that the Rijndael algorithm was the winner. Field Programmable Gate Arrays (FPGAs) are hardware devices whose function is not fixed which can be programmed in system. Advanced Encryption Algorithm includes efficiency testing of both hardware and software implementations of candidate algorithms. Reprogrammable devices such as field-programmable gate arrays (FPGAs) are highly attractive options for hardware implementations of encryption algorithms, as they provide physical security, and potentially much higher performance than software solutions.

With the development of computer technology and cryptanalysis, the DES (Data Encryption Standard, DES) algorithm is already no longer safe. So AES (Advanced Encryption Standard) substitutes DES and already becomes the new standard. AES algorithm is already supported by a few international standards at present, and AES algorithm is widely applied in the financial field in domestic, such as realizing authenticated encryption in ATM, magnetism card and intelligence card. Furthermore, this system can be widely used in the terminal equipment's which less demand on the throughput. Throughput found for encryption decryption is 593.45Mbps and 267.63Mbps respectively [2].

The design uses an iterative looping approach with block and key size of 128 bits, lookup table implementation of S-box [8]. This gives low complexity architecture and easily achieves low latency as well as high throughput. The algorithm achieves a low latency and throughput reaches value of 1054 Mbps for encryption and 615Mbps for decryption. Latency of encryption is only 13 clock cycles and latency of decryption is 25 clock cycles [1].

III. PROPOSED WORK

A. *AES*



(a) Encryption Process (b) Decryption Process

Fig.1 AES Encryption / Decryption Process

Advanced Encryption Standard (AES) is a Symmetric key cryptography and it is block cipher with a fixed block size of 128 bit and a variable key length i.e. it may be 128,192 or 256 bits. Block cipher means data is encrypted & decrypted if data is in the form of blocks [6]. AES uses a variable number of rounds, which are fixed for key size. A key of size 128 has 10 rounds. A key of 192 has 12 rounds. A key of size 256 has 14 rounds. Each round consists of several processing steps, each containing four similar but different stages, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key. For Encryption each rounds has four operations SubBytes, ShiftRows, MixColumns and AddRoundKey respectively and for decryption it use inverse of these function.

1) *AES Encryption:* The first and last rounds differ from other rounds in that there is an additional AddRoundKey transformation at the beginning of the first round and no MixColumns transformation is performed in the last round. In this paper, we use the key length of 128 bits. Steps of AES encryption is given in Figure 1.

a) SubBytes Transformation:

The SubBytes transformation is a non-linear byte substitution, operating on each of the states bytes independently. The SubBytes transformation uses pre-calculated substitution table called S-box, thus reducing the latency and avoids complexity of hardware implementation. The S-box is presented in hexadecimal form.

Table 1: S-box

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

b) ShiftRows Transformation:

In ShiftRows transformation, the rows of the state are cyclically left shifted over different number of bytes.Row0 is not shifted, row 1 is shifted one byte to the left, row 2 is shifted two bytes to the left and row 3 shifted three bytes to the left.

c) MixColumns Transformation:

The MixColumns transformation operates on the State column-by-column, treating each column as a four-term polynomials. The columns are considered as polynomials over GF (2^8) and multiplied modulo x^4+1 with a fixed polynomial a(x), given by:

$$a(x)=\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}.$$

This can be written as a matrix multiplication. Let

$$s'(x) = a(x) \otimes s(x)$$

d) AddRoundKey Transformation:

In the AddRoundKey transformation, a Round key is added to the state - resulted from the operation of the MixColumns transformation by a simple bitwise XOR operation. The round key of each round is derived from the main key using the Key Expansion algorithm. The encryption/decryption algorithm needs eleven 128-bit rounds.

2) AES Decryption

Decryption is a reverse of encryption which inverse round transformations to computes out the original plaintext of an encrypted cipher-text in reverse order shown in fig.1.b). The round transformation of decryption uses the functions AddRoundKey, InvMixColumns, InvShiftRows, and InvSubBytes successively.

a)AddRoundKey:

AddRoundKey is its own inverse function because the XOR function is its own inverse. The round keys have to be selected in reverse order.

b)*InvShiftRows Transformation*:

InvShiftRows exactly functions the same as ShiftRows, only in the opposite direction. The first row is not shifted, while the second, third and fourth rows are shifted right by one, two and three bytes respectively.

c) InvSubBytes transformation:

The InvSubBytes transformation is done using a once recalculated substitution table called Inv S-box. That Inv S-box table contains 256 numbers (from 0 to 255) and their corresponding values.

Table 2: Inverse S-byte

		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	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
x	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

d) InvMixColumns Transformation:

The InvMixColumns transformation is done using polynomials of degree less than 4 over GF(28), which coefficients are the elements in the columns of the state, are multiplied modulo $x^4 + 1$ by a fixed polynomial

$$d(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\},$$

Where $\{0B\}$, $\{0D\}$; $\{09\}$, $\{0E\}$ denote hexadecimal values.

B. RSA (Rivest, Shamir and Adleman) Algorithm:

First of all, two large distinct prime numbers *p* and *q* must be generated. The product of these, we call miss a component of the public key. It must be large enough such that the numbers *p* and *q* cannot be extracted from it - 512 bit sat least i.e. numbers greater than 10154. We then generate the encryption key *e* which must be co-prime to the number *m* =*E*(*n*) = (*p* - 1)(*q* -1).We then create the decryption key *d* such that *de mod m*= 1. We now have both the public and private keys.

1) RSA Encryption: We let *C*= *E*(*x*) be the encryption function where *x* is an integer and *y* is the encrypted form of *x*

C= M^e mod m

2) *RSA Decryption:* We let *M*= *D*(*y*) be the decryption function where *y* is an encrypted integer and *X* is the decrypted form of y

M= C^f mod m

3) Example:

a) Generator primes create the modulus:

P=47, q=71

$m=p.q$, $m=3337$

b) Public key calculated:

$$(47-1).(71-1)=mx=3220$$
$$Gcd(e,3220)=1;$$
$$E=79$$

c) Private key calculated:

$$(f.79)mod3220 = 1$$
$$F=Euc(79,3220) = 1019$$

d) Message, a data block to the value of 688, encrypted using the public key:

$$M = 688$$
$$C = 688^{179}mod3337 =1570$$

e) Cipher text decrypted with the private key to obtain the original data block:

$$c=1570$$
$$m=1570^{1019}=688$$

IV. IMPLEMENTATION

The design has been coded in VHDL for AES algorithm and results shown in below figure. Simulation is done using ModelSim software.

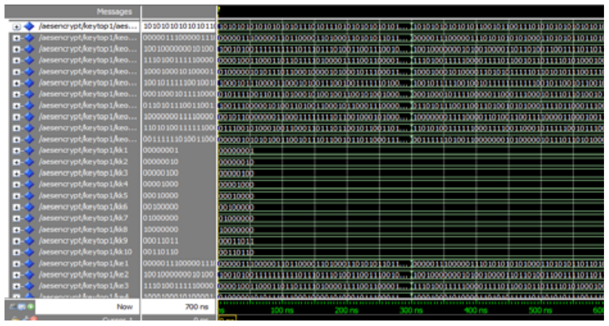


Fig.2 AES Encryption

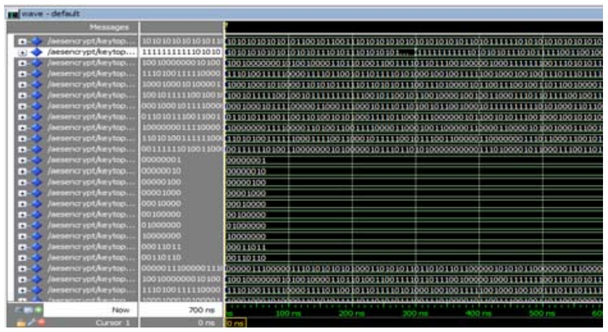


Fig.3 AES Decryption

For this implementation we required Vertex kit. For cost optimization we are coding both the algorithm using Impulse C language and implemented on Sparten3 FPGA kit. For the coding we are using Xilinx platform Studio software.

A. *AES Implementation*:

Generally 128 bit plaintext is an input for AES but we have implemented for 256 bit input and key of length 192 bit.

Encryption Output:

Plain text: I AM LEGEND OF THE NATION INDIA

Key: ABCDEFGHIJKLMNOPQRSTUVW

Cipher text: 70B6B86B D1737A9F 5E0C8512 F551EC09 CC51E5F0 132A9D76 6F34A6DC 318D1CDD

Decryption Output:

Input text: 70B6B86B D1737A9F 5E0C8512 F551EC09 CC51E5F0 132A9D76 6F34A6DC 318D1CDD

Plain text: I AM LEGEND OF THE NATION INDIA

B. RSA Implementation

Plain text: Asymmetric

Private Key: 103

Public Key: 7

Cipher text: 65 80 121 879 879 920 844 478 833 330

Decryption Output: Asymmetric

V. CONCLUSIONS

In Data communication, encryption algorithm plays an important role. AES algorithm is better than RSA because AES includes large number of mathematical operation so it is almost difficult to decrypt by attacker. AES is symmetric key cryptography and RSA is asymmetric key cryptographic. Peak memory usage for AES algorithm is 220MB and for RSA algorithm is 229MB. Thus AES and RSA algorithm used in various applications for security.

Table 3: Comparison between AES & RSA

Sr. No.	Factors	AES	RSA
1	Developed	2000	1978
2	Block Size	128 bits	Minimum 512 bits
3	Key Size	128,192,256 bits	>1024 bits
4	Rounds	10/12/14	1
5	Ciphering & deciphering key	Same	Different
5	Algorithm	Symmetric	Asymmetric
6	Key Used	Same Key used for Encrypt & Decrypt Process	Different key used for Encrypt & Decrypt Process
7	Security	Excellent Secured	Least Secured

Table 4: Resource Utilization

Device Utilization Summary (estimated values)					
	Used			Utilization	
Logic Utilization	AES	RSA	Available	AES	RSA
Total Number of Slices Register	1572	1572	3840	40%	40%
Number of occupied Slices	1452	1517	1920	75%	79%
Number of 4 input LUTs	2286	2317	3840	59%	60%
Number of bonded IOBs	62	62	97	63%	63%
Number of RAMB16s	4	4	12	33%	33%
Number of MULT18*18s	3	3	12	25%	25%
Number of BUFGMUXs	2	2	8	25%	25%
Number of DCMs	1	1	4	25%	25%
Number of BSCANs	1	1	1	100%	100%

APPLICATION

- Medical Report
- Commercial
- Military
- Bank Services
- Cloud computing

ACKNOWLEDGMENT

The authors would like to thank the researchers as well as publishers for making their resources available and teachers and guide for their guidance. We are thankful to the authorities of Savitribai Phule University,Pune. We also thank the college authorities for providing the required infrastructure and support. Finally, we would like to extend a heartfelt gratitude to friends and family members.

REFERENCES

[1] Hoang Trang and Nguyen Van LoiHoChiMinh City, VietNam- "An efficient FPGA implementation of the Advanced Encryption Standard algorithm" 978-1-4673-0309-5/12/ ©2012 IEEE

[2] Yang Jun Ding Jun Li Na GuoYixiong School of Information Science and FPGA Engg. Yunna University Kunming, China- "Based design and implementation of reduced AES algorithm" IEEE 2010

[3] WANG Wei, CHEN Jie& XU Fei, China- "An Implementation of AES Algorithm Based on FPGA" IEEE 2012

[4] John Fry, Martin Langhammer Altera Corporation- "RSA & Public key Cryptography in FPGAs"

[5] O P Verma, RituAgarwal, DhirajDafouti, ShobhaTyagi "Performance Analysis of Data Encryption Algorithms",IEEE 2011 978-1-4244-8679-3/11

[6] Shashi Mehrotra Seth, Rajan Mishra- "Comparative Analysis of Encryption Algorithms for Data Communication", IJCST Vol. 2, Issue 2, June 2011

[7] FIPS 197, "Advanced Encryption Standard(AES)",November 26,2001

[8] Ahmad, N., Hasan, R., Jubadi, W.M, "Design of AES S-Box using combinational logic optimization", IEEE Symposium on Industrial Electronics & Applications (ISIEA), pp. 6966-699.2010.