ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 12, December 2015

Survey on Implementation of Dedicated Hardware for Encryption

[1]Gaurav Berad, [2]Prof. J.S. Raghatwan, [3]Prof. Mayur Aknewar

[1] Research Scholar, Dept. of Computer Engineering, RMD Sinhgad School of Engineering, Savitribai Phule University of Pune, India,

[2,3]Assistant Professor, Dept. of Computer Engineering, RMD Sinhgad School of Engineering, Savitribai Phule University of Pune, India

ABSTRACT: Recently, the number of electronic devices handling confidential information has increased. In these devices, encryption is applied to protect the confidential information. Therefore, technologies to incorporate cryptographic circuits into these cards have become important. This paper proposes a new hardware dedicated to a typical cryptograph. In the proposed RC5 dedicated hardware, by introducing an architecture suitable for each operation used for the encryption, high-speed processing and area reduction can be realized. Experimental results of which proposed hardware architecture is implemented on FPGA proved the validity of the proposed one.
.

KEYWORDS: Cryptography, Dedicated Hardware, FPGA implementation.

I. INTRODUCTION

This study proposed hardware dedicated to Cryptographic Algorithms such as RSA and RC and RSA, which are common-key block cipher. In the proposed hardware, by introducing a new hardware-oriented algorithm into the shift arithmetic, mixture, and round processes during encryption and decryption, high-speed processing and area reduction were realized. Incorporating it into an FPGA verified the proposed hardware's validity. In the future, we will develop an identification system using the proposed hardware.

II. RELATED WORK

Security systems and methods are often described as strong or weak. A strong system is one in which the cost of attack is greater than the potential gain to the attacker. Conversely, a weak system is one where the cost of attack is less than the potential gain. Authentication factors are one of the key systems that work in our favor dual or multi-layered authentication. There is increasing need of information data in Computer Network and Communication Technology. This data is handled by public networks and it is vulnerable. So cryptography becomes important for such sensitive data which should be kept secure and safe against automated spying or hacking[1].

RSA

RSA is one of the first practical public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret. In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem. RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1977. Clifford Cocks, an English mathematician, had developed an equivalent system in 1973, but it was not declassified until 1997. A user of RSA creates and then publishes a public key based on two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime numbers can feasibly decode the message. Breaking RSA encryption is known as the RSA problem; whether it is as hard as the factoring problem remains an open question[4].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 12, December 2015

Key Generation: RSA involves a public key and a private key. The public key can be known by everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key. The keys for the RSA algorithm are generated the following way:

Choose two distinct prime numbers p and q.

For security purposes, the integers p and q should be chosen at random, and should be similar in magnitude but 'differ in length by a few digits' to make factoring harder. Prime integers can be efficiently found using a primality test.

Compute n = pq.

n is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.

Compute $\varphi(n) = \varphi(p)\varphi(q) = (p - 1)(q - 1) = n - (p + q - 1)$, where φ is Euler's totient function. This value is kept private.

Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; i.e., e and φ(n) are coprime.

e is released as the public key exponent.

e having a short bit-length and small Hamming weight results in more efficient encryption – most commonly $216 + 1 = 65,537$. However, much smaller values of e (such as 3) have been shown to be less secure in some settings.

Determine d as $d \equiv e^{-1} \pmod{\varphi(n)}$; i.e., d is the modular multiplicative inverse of e (modulo φ(n)).

This is more clearly stated as: solve for d given $d \cdot e \equiv 1 \pmod{\varphi(n)}$

This is often computed using the extended Euclidean algorithm. Using the pseudo code in the Modular integers section, inputs a and n correspond to e and φ(n), respectively.

d is kept as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and φ(n) must also be kept secret because they can be used to calculate d.

An alternative, used by PKCS#1, is to choose d matching $de \equiv 1 \pmod{\lambda}$ with $\lambda = \text{lcm}(p - 1, q - 1)$, where lcm is the least common multiple. Using λ instead of φ(n) allows more choices for d. λ can also be defined using the Carmichael function, λ(n).

Since any common factors of (p-1) and (q-1) are present in the factorization of p*q-1, it is recommended that (p-1) and (q-1) have only very small common factors, if any besides the necessary [2].

RC5

In cryptography, RC5 is a symmetric-key block cipher notable for its simplicity. Designed by Ronald Rivest in 1994, RC stands for "Rivest Cipher", or alternatively, "Ron's Code" (compare RC2 and RC4). The Advanced Encryption Standard (AES) candidate RC6 was based on RC5[5]. Unlike many schemes, RC5 has a variable block size (32, 64 or 128 bits), key size (0 to 2040 bits) and number of rounds (0 to 255). The original suggested choice of parameters were a block size of 64 bits, a 128-bit key and 12 rounds. A key feature of RC5 is the use of data-dependent rotations; one of the goals of RC5 was to prompt the study and evaluation of such operations as a cryptographic primitive. RC5 also consists of a number of modular additions and eXclusive OR (XOR)s. The general structure of the algorithm is a Feistel-like network. The encryption and decryption routines can be specified in a few lines of code. The key schedule, however, is more complex, expanding the key using an essentially one-way function with the binary expansions of both e and the golden ratio as sources of "nothing up my sleeve numbers". The tantalising simplicity of the algorithm together with the novelty of the data-dependent rotations has made RC5 an attractive object of study for cryptanalysts. The RC5 is basically denoted as RC5-w/r/b where w=word size in bits, r=number of rounds, b=number of 8-bit byte in the key. RC5 encryption and decryption both expand the random key into 2(r+1) words that will be used sequentially (and only once each) during the encryption and decryption processes.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 12, December 2015

Key Generation: The algorithm is illustrated below, first in pseudo code, then example C code copied directly from the reference paper's appendix.

Following the naming scheme of the paper, the following variable names are used:

b - The length of the key in bytes.

K - The key, considered as an array of bytes (using 0-based indexing).

w - The length of a word in bits. Typical values of this in RC5 are 16, 32, and 64. Note that a "block" is two words long.

u - The length of a word in bytes.

r - The number of rounds to use when encrypting data.

S - The expanded list of words derived from the key, of length 2(r+1), with each element being a word.

L - A convenience to encapsulate K as an array of word-sized values rather than byte-sized.

Pw - The first magic constant, defined as Odd((e - 2) * 2^w), where Odd is the nearest odd integer (rounded up) for the given input, where e is the base of the natural logarithm, and w is defined above. For common values of w, the associated values of Pw are given here in hexadecimal:

For w = 16: 0xB7E1

For w = 32: 0xB7E15163

For w = 64: 0xB7E151628AED2A6D

Qw - The second magic constant, defined as Odd((\phi - 1) * 2^w), where Odd is the nearest odd integer (rounded up) for the given input, where \phi is the golden ratio, and w is defined above. For common values of w, the associated values of Qw are given here in hexadecimal:

For w = 16: 0x9E37

For w = 32: 0x9E3779B9

For w = 64: 0x9E3779B97F4A7C15

III. PROPOSED ALGORITHM

AES

AES was standardized by National Institute of Standards and Technology (NIST) in 2001 became Federal Information Processing Standard FIPS-197.Where Rijndael algorithm by Joan Daeman and VicentRijimen was selected as standard AES algorithm[3]. The AES is private or symmetric block cipher which uses the same key for encryption and decryption is more suitable for faster implementation. The AES is a symmetric key for both encryption and decryption. AES cryptography algorithm is capable of encrypting and decrypting block size 128 bit data using cipher keys of 128, 196 or 256 bits (AES128, AES196 and AES256). AES can be implemented in software or hardware but, hardware implementation is more suitable for high speed applications in real time. Main goal AES hardware implementation is high throughput design and low-area design work at highest operating frequency.

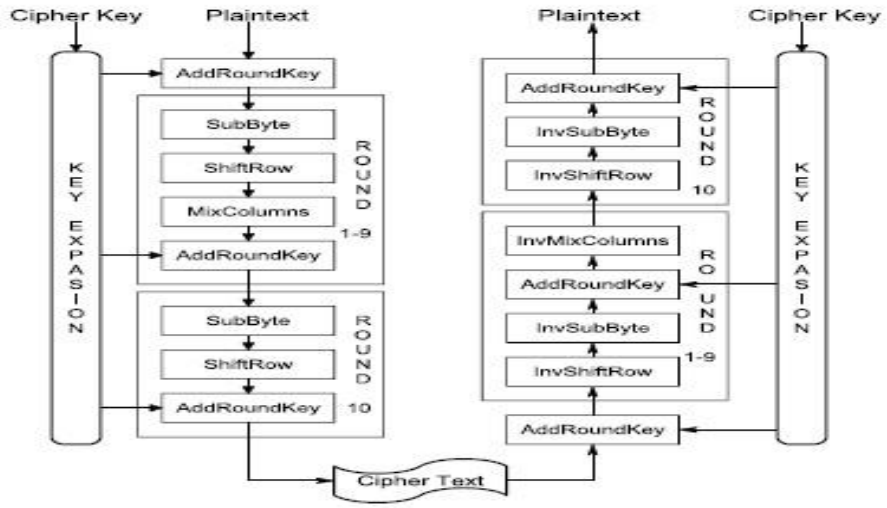
AES algorithm can resist any kinds of password attacks with a strong practicability in information security and reliability. So AES is widely adopted for various applications from high-end computers to low power portable devices. Numerous AES hardware architecture use in computer processor, SAN & Wi-Fi network, ATM, cellular phones and digital video recorders. AES algorithm is symmetric block cipher that processes the state arraying from 128 bits data block using a key of 128, 192 or 256 bits length repeatedly. In encryption process round function consists of four different transformations- SubBytes, ShiftRows, MixColumns and AddRoundKey but last round function without MixColumns transformations. A ShiftRowstransformations are truly dependent on state-wise operation of cyclic row shifting and MixColumns select 4 byte column operation done at simultaneously. Similarly inverse chipper decryption process round function consists of four different transformations- InvSubBytes, InvShiftRows, InvMixColumns and AddRoundKey, but last round function without InvMixColumns transformations. Key Expansion or Schedule generation module is a common unit in AES encryption and decryption core. Key Expansion used to generate a series of Round Keys from the Cipher Key. A State is two-dimensional array of 4 x 4 size having 8 bytes consists of four rows contain block length divided by 32 is presented in hexadecimal format.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 12, December 2015



AES standard encryption & decryption Algorithm

S-box Architecture

S-Box for SubByte and InvSubByte operation is implement using two method - Conventional BRAM implementation or combinational logic. Conventional BRAM has all pre-computed 256 values are stored in a ROM based lookup table and input byte wired to ROM's address bus. But BRAM method suffers from unbreakable delay as fixed access time for read and write operation and low latency due to ROM access time To increase throughput parallel ROMs are leading to large size of chip area requires high amount of memory. Therefore S-box transformation through composite field arithmetic is more suitable for low latency with reduction in area against. A more suitable second method to implementing S-Box is using combinational logic. It has advantage like small area occupancy and pipelined for increased performance in clock frequency. In this paper S-Box architecture based on combinational logic is present.
It is computed by multiplicative inverse in GF(2^8) followed by an affine transformation. Where InvSubByte transformation compute using inverse affine transformation is applied first then multiplicative inverse
The Affine Transformation (AT) and inverse Affine Transformation (AT^{-1}) are computed using following equation

$$AT(a) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

$$AT^{-1}(a) = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 12, December 2015

Both Sbox and InvSbox transformations have same multiplicative inversion module. GF multiplication is compute by decomposing complex GF(2^8) to lower order fields as GF(2^1) and GF(2^2) also GF((2^2)^2). In computation of multiplicative inverse in composite fields cannot be directly applied to GF(2^8) multiplication.

Addition of GF(2^4) elements is compute using simply bitwise XOR operation. Multiplicative Inversion (X-1) is inverse of individual bits compute from larger equation so that pre-computed value can be used. The Isomorphic Mapping (d) and Inverse Isomorphic Mapping (d^{-1}) composite field are compute using following equation

$$\delta \times q = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \times \begin{pmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{pmatrix} \quad \delta^{-1} \times q = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{pmatrix}$$

A GF(2^4) Squaring operation compute using below equation

S2 = d3 XOR d2;

S1 = d2 XOR d1;

S0 = d3 XOR d1 XOR d0;

S3 = d3;

Multiplication with constant (λ) is generate by substitute irreducible polynomial as shown in below expression.

λ3 = S2 XOR S0;

λ1 = S3;

λ0 = S2;

λ2 = S3 XOR S2 XOR S1 XOR S0;

Multiplication of GF(2^4) contain addition and decomposition multiplication operations in GF(2^2).

Multiplication with constant ($) is derived from equation
$1 = q1 XOR q0; $0 = q1;

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

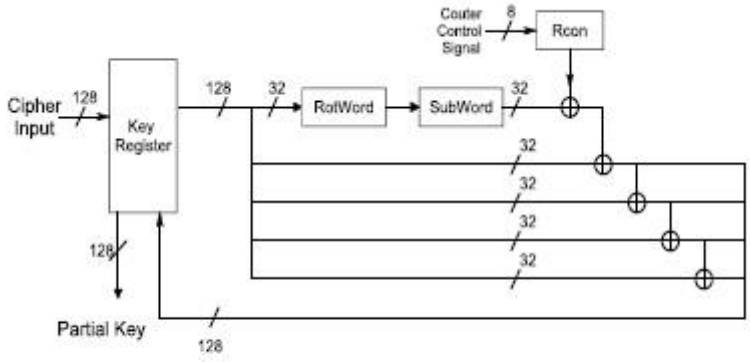
International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 12, December 2015

Key Expansion

Key Expansion routine to perform key scheduling which generate a series of Round Keys from the cipher key as shown figure 6. SubWord present in Key Expansion routine that takes a 4-byte input word gives 4-bytes output word using Sbox. The RotWord function performs a cyclic permutation on input word gives cyclic right shifted 4 bytes output word. Rcon is array of bytes in a word having fixed logical value having size of 128 bit. A 128 bit Key register is fixed signal used to temporal storage of cipher key computed for each round of operation. Here key expansion module generate 10 number of 128 bit size Partial key for each round of operation.

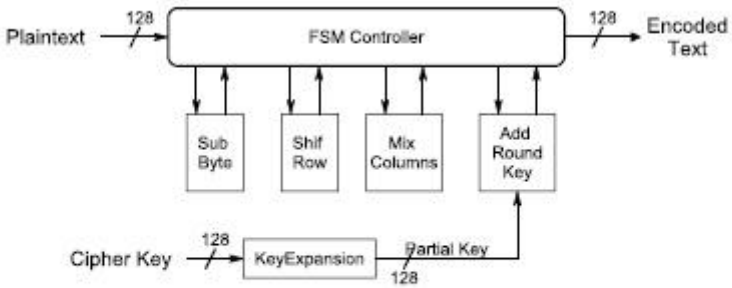


Block diagram of Key Expansion

AES Encryption

AES Encryption has following subsequent steps are: SubByte, ShiftRows, MixColumns and AddRoundKey as shown figure. In SubBytes transformation is cipher undergo process of nonlinear byte substitution table (S-box) that operates on each of the State bytes independently. ShiftRows transformation is cipher that processes the State by cyclically right shifting of last three rows in State. MixColumns is transformation where cipher takes a columns of State and mixes their data independently gives one another to produce new columns using GF (2^8) polynomial. AddRoundKey is transformation cipher and Inverse Cipher is XOR operation with Round Key added to State.

A FSM controller is used for synchronization purpose where clock and reset are input of system. A block 128 bit plaintext input is XOR with partial key repeated for 10 times uses SubByte, ShiftRows, MixColumns, and AddRoundKey which generate encoded text. The main purpose of saving computation period of key expansion operation unnecessary repeated for same cipher key which enhance throughput of system by maintain standard frequency.



Block diagram of AES Encryption

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

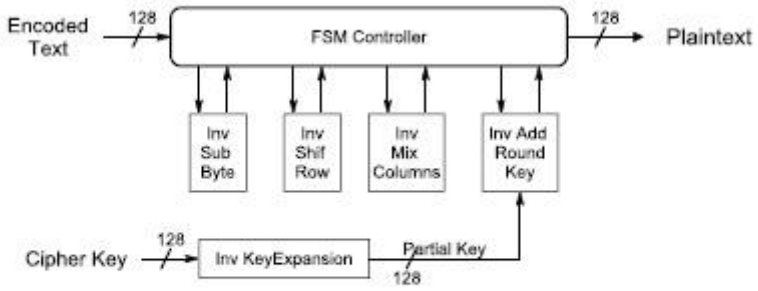
(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 12, December 2015

AES Decryption

It is inverted operation of encryption is implemented using reverse order Inverse Cipher in AES algorithm. AES decryption contain following subsequent steps are: InvShiftRows, InvSubBytes, InvMixColumnsand AddRoundKey as shown figure. InvShiftRows is transformation is inverse of ShiftRows processes the State by cyclically left shifting of last three rows in State.

InvSubBytes is the inverse of byte substitution transformation. An inverse S-box obtained by applying the inverse affine transformation followed by taking multiplicative inverse in GF(2^8) polynomial. InvMixColumns is the inverse of MixColumns transformation operates on column-wise operation four term polynomial and multiplied modulo (X^4+1) with a fixed polynomial. InvAddRoundKey Transformation is own inverse AddRoundKey which involves 128 bitwise XOR operation. InvKeyExpansion module is 128 bit cipher generate 10 number of 128 bit size Partial key for each round of operation similar as KeyExpansion. But major difference is sequence of generated partial key is inverted.



Block diagram of AES Decryption

IV. CONCLUSION AND FUTURE WORK

The simulation results showed that the proposed algorithm performs better with the total transmission energy metric than the maximum number of hops metric. The proposed algorithm provides energy efficient path for data transmission and maximizes the lifetime of entire network.

In order to realize high-speed processing and area reduction, this study introduces arithmetic processes suitable for hardware during encryption and decryption. First, a shift process used for encryption is replaced by a bit selection process. Because of this substitution, the shift process can be realized in wiring. The mixture process usually requires t x 3 = 78 calculations. The proposed architecture divides the processing by inserting a register between calculations and introducing a loop process, which reduce the required calculations to 26. Dividing the processing of each round enables high-speed processing, Cryptographs round processing enciphers a 64-bit plaintext. However, the actual processing is performed every 32 bits. Therefore, the round is divided into the left and right parts to perform left and right processing with two clock signals, which raises the operating frequency and improves latency.

In order to verify the proposed architecture's validity, we incorporated it into an FPGA. The FPGA used in the experiment was XC5VLX30/50 and we used ISE ver.12.3 of Xilinx as the developmental environment. It assumes the target device for implementation. For comparison, the algorithm used as described in Verilog HDL. This paper refers to the algorithm used for comparison as the normal method;

Important information such as account numbers is recorded on magnetic tapes on the back of credit and cash cards. Because magnetic tapes can be forged easily, integrated circuit (IC) chips (cryptographic circuits) have been used in recent years. Therefore, technologies to incorporate cryptographic circuits into these cards have become important. This study proposes hardware dedicated to a typical cryptograph. Such as RC5, RSA, and AES which are common-key block cipher in which the size of the block, the number of rounds, and the size of the key can be changed. RC5 is enciphered by combining three operations of exclusion: exclusive OR, rotation, and residue addition. In the proposed RC5, and RSA dedicated hardware, introducing an architecture suitable for each operation used for encryption enables

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 12, December 2015

the realization of high-speed processing and area reduction. Moreover, the proposed architecture is incorporated into a field-programmable gate array (FPGA), and we verify its validity.

As the performance of the proposed algorithm is analyzed between two metrics in future with some modifications in design considerations the performance of the proposed algorithm can be compared with other energy efficient algorithm.

REFERENCES

[1] A.Klimm, M.Haas, O.Sander, J.Becker, "A flexibleintegrated cryptoprocessor for authentication protocols basedon hyperelliptic curve cryptography", Proc. of International Symposium on System on Chip (SoC), pp.35-42, 2010.

[2] A.Irwansyah, V.P.Nambiar, M.Khalil-Hani, "An AESTightly Coupled Hardware Accelerator in an FPGA-basedEmbedded Processor Core", Proc. of InternationalConference on Computer Engineering and Technology, Vol.2,pp.521-525, 2009.

[3] B.B.Brumley, K.U.Jarvinen, "Conversion Algorithmsand Implementations for Koblitz Curve Cryptography", IEEETransactions on Computers, Vol.59, No.1, pp.81-92, 2010.

[4] M.Rahman, I.R.Rokon, "Efficient hardwareimplementation of RSA cryptography", Proc. of InternationalConference on Anti-counterfeiting, Security, andIdentification in Communication, pp.316-319, 2009.

[5] F.A.G.Muzzi, R.B.Chiaramonte, E.D.M.Ordonez, "TheHardware-based PKCS#11 Standard using the RSAAlgorithm", IEEE Latin America Transactions, Vol.7, No.2,pp.160-169, 2009.

BIOGRAPHY

Gaurav Berad is a Research Scholar, Department of Computer Engineering, RMD Sinhgad School of Engineering, Savitribai Phule University of Pune, India. The research was supported by the senior faculty and Scholars of Sinhgad Technical Education Society's RMD Sinhgad School of Engineering, Pune Department of Computer Engineering.