Novel Memory Efficient Key Expansion-Inversion Technique for Cryptography Applications using Extended Hamming Code

B. Senthilkumar [#1], V. Rajamani [*2]

[#] Research Scholar, Department of Electrical and Electronics Engineering,
St. Peter's University, Avadi, Chennai, Tamilnadu, INDIA – 600054.
Email: Senthilkumar_05@yahoo.co.in

[*] Principal, Veltech Multitech Dr.Rangarajan Dr.Sakunthala Engineering College,
Avadi, Chennai, Tamilnadu, INDIA – 600062.
Email: rajavmani@gmail.com

Abstract—This paper describes about novel key expansion and its inversion technique for private key cryptosystems. Our design uses (8, 4) Extended Hamming Code and its error control logic to produce memory efficient key schedule generation algorithm. A mathematical relationship between 4bit word and its corresponding 4bit parity bits is shown. Simplicity, symmetry elimination, diffusion and non-linearity of the proposed key expansion technique are described as the key schedule generation criteria. Proposed method removes the usage of S-box to reduce the working memory of the algorithm. High nonlinearity penetration of original input message bits is achieved by applying modulo2 addition of code based key schedules for each round transformations. Security strength among these key schedules is achieved by intentional bit inversions among them with beyond the error correcting limitations of chosen code. Comparative results between proposed design and Rijndael algorithm is illustrated with the aid of Xilinx Simulation tool. This paper concludes that novel key generation technique by Error Control Algorithm of wireless communication channel is an alternative solution to the cryptosystems without S-box substitution and any lookup tables.

Keyword - Key expansion, Key generation, Message-Parity correlation, Extended Hamming Code, Crypto Coding.

I. INTRODUCTION

All the Substitution-Permutation based block cipher cryptosystems [1] comprise the S-box that has no direct functions with secret key. Secret key is the only user dependent and changeable factor in symmetric key cryptosystem. The use of code-depended key generation is not extensively discussed in the literature as it will lead the complexity in the cryptographic algorithms. But the existence of well-developed and mathematically proved known cryptanalysis [2] lead the need of alternative methods of key generation methods that should not be influenced by any known properties and structures of present algorithms. Our design uses the coding theory concept [3] to generate key schedules by bit expansion and its inversion. In the substitution-permutation based cryptosystem, security strength is mainly depends on its internal structure [4] rather than the key expansion technique. Therefore, the key expansion technique is centered on four major design objects such that simplicity, symmetry elimination, diffusion and non-linearity as a common implementation requirement of any cryptosystem.

In this paper, a new method is presented to generate random output vector from the novel s-box with a function of the secret key. Rest of the paper is organized as follows: Section 2 briefly introduces the Advanced Encryption Standard (AES) algorithm; Section 3 describes about the (8, 4) extended hamming code and its error control algorithm; Section 4 describes the novel logical method of generation of reversible 4bit word; Section 5 describes the key schedule generation technique of proposed design; Section 6 illustrates design criteria for the security strength of proposed design against known attacks; Section 7 expresses the mathematical and constructional details of the proposed design; Section 8 describes the comparison of proposed design with AES algorithm and section 9 provides conclusion on our work.

II. EXISTING RIJNDAEL (AES) ALGORITHM

In AES, 128 bit key length requires 10 times of round repetition. The linear and non-linear operations in round transformations are reversible to allow decryption using their inverses. Every transformation affects all bytes of the State. The byte substitution transformation is a nonlinear function that operates on each byte of the 128 bit State using a S-box table. The elements of the table are computed by a finite field inversion followed by

an affine transformation. The Shifting Rows transformation is a row index based circular shifting operation that rotates the rows of the State. Second row is shifted by one byte to the left, the third row is shifted by two bytes to the left, the fourth row is shifted by three bytes to the left, and first row is shifted by four bytes to the left. Mixing Columns transformation mixes the bytes in each column by multiplying the coefficients of the polynomial with modulo $x^4 + 1$. The Round Key adding transformation is an XOR operation that adds the 128 bit round key to the 128 bit state in each round. The round keys are generated during the key expansion process. The initial round key equals to secret key [5].

A. Round key generation technique in AES

Advanced Encryption Standard (AES) technique is a symmetric key algorithm where same key is used for both encryption and decryption. For 128 bit key size, there are 10 round key schedule generations with the help of Substitution box look-ups, round constants and modulo2 additions. The mathematical expression of one of the round key generation process is shown below.

If K_0 is the 128 bit 0^{th} round key , K_1 is the128 bit 1^{st} round key, RC_1 is the 8 bit round constant and $K_1(b_i) = i^{th}$ bit among 128 bit of K_1 where $i = 127 \geq i \geq 0$, then

K_1 (b_{127} to b_{120}) = SBOX_LOOKUP (b1) xor "RC_1" xor K_0 (b_{127} to b_{120}); K_1 (b_{119} to b_{112}) = SBOX_LOOKUP (b2) xor K_0 (b_{119} to b_{112}); K_1 (b_{111} to b_{104}) = SBOX_LOOKUP (b3) xor K_0 (b_{111} to b_{104}); K_1 (b_{103} to b_{96}) = SBOX_LOOKUP (b0) xor K_0 (b_{103} to b_{96}); K_1 (b_{95} to b_{64}) = K_1 (b_{127} to b_{96}) xor K_0 (b_{95} to b_{64}); K_1 (b_{63} to b_{32}) = K_1 (b_{95} to b_{64}) xor K_0 (b_{63} to b_{32});K_1 (b_{31} to b_0) = K_1 (b_{63} to b_{32}) xor K_0 (b_{31} to b_0);

where b1 = K_0 (b_{31} to b_{24}); b2 = K_0 (b_{23} to b_{16}); b3 = K_0 (b_{15} to b_8); b4 = K_0 (b_7 to b_0); SBOX_LOOKUP = function of Substitution box look-up for set of bits and "xor" = function of modulo 2 addition.

Here, s-box look-ups, round constant and modulo 2 additions provide necessary nonlinearity property for symmetric key algorithm on the generations of round keys.

III. Extended Hamming Code and its Error Control Algorithm

A (8, 4) Extended Hamming Code 'C' is constructed by transforming all the 4 bit of information (**i**) over GF(2^4) into sixteen 8 bit code words and $C \in \{c_j ; 0 \leq j \leq 15\}$, where c_j is a code word (1)

If 4bit information **i** = (i_1, i_2, i_3, i_4), then codeword '**c**' = **iG =** (i_1, i_2, i_3, i_4, p_1, p_2, p_3, p) (2)

where $p_1 = i_1 + i_2 + i_3$, $p_2 = i_2 + i_3 + i_4$, $p_3 = i_1 + i_2 + i_4$, $p = i_1 + i_3 + i_4$, **'G'** is 4x8 Generator matrix and '+' is modulo2 addition.

From the equation 2, for C is the (8, 4) Extended hamming code, then all the vectors over GF(2^4) will have the 16 code words of code C such that $C \in$ {00,17,2d,3a,4e,59,63,74,8b,9c,a6,b1,c5,d2,e8,ff}.

A. Hamming Code as Forward Error Control Code (FEC Code)

The error correction limits 't' and error detection limits '*l*' of Forward Error Control codes are bounded by hamming distance or minimum distance (d_{min}) of a code. Codes with error correction limit 't' and error detection limit '*l*' are referred to as t- error correcting codes and *l*- error detecting codes respectively.

Mathematically, $t = l/2$ where $l = d_{min} - 1$ (3)

IV. Conversion of Message Bits to Parity Bits and its Inversion

If the 4bit message is represented by $m_3m_2m_1m_0$ and 4bit parity is represented by $p_3\ p_2\ p_1\ p_0$, then from equation number 2,

p0 = (m3 + m1 + m0); p1 = (m3 + m2 + m0); p2 = (m2 + m1 + m0); p3 = (m3 + m2 + m1); (4)

From equation 4, message bits $m_3m_2m_1m_0$ are reversible as given in the equation 5.

$m_0 = (p_2 + p_1 + p_0)$; $m_1 = (p_3 + p_2 + p_0)$; $m_2 = (p_3 + p_2 + p_1)$; $m_3 = (p_3 + p_1 + p_0)$; (5)

where the symbol '+' is modulo2 addition.

Proof:

p2 + p1 + p0 = m2 + m1 + m0 + m3 + m2 + m0 + m3 + m1 + m0 = m0 (6)

p3 + p2 + p0 = m3 + m2 + m1+ m2 + m1 + m0 + m3 + m1 + m0 = m1 (7)

p3 + p2 + p1 = m3 + m2 + m1+ m2 + m1 + m0 + m3 + m2 + m0 = m2 (8)

p3 + p1 + p0 = m3 + m2 + m1+ m3 + m2 + m0 + m3 + m1 + m0 = m3 (9)

From equation 4 to 9, it is evident that 4bit message can be converted into 4bit parity. Similarly, any 4bit parity can be reversed into its corresponding unique 4bit message.

As described in the section 3, the uniqueness among the pair of message and parity bits over the finite field GF(2^4) is the main advantage of using the (8, 4) Extended Hamming Code in the proposed design.

V. GENERATION OF KEY SCHEDULE FOR PROPOSED DESIGN

The single error correcting limit of hamming code is effectively used in our design for generation of key schedule. If the 8bit code word of (8, 4) extended hamming code is purposefully altered with more than one error, there is no existence of algorithms to retrieve the original code word. It can only be retrieved by corresponding error correction vector as per the channel coding theory. The required size of key schedule from 64bit sub-key (K_s) is prepared by three steps. In the first phase, 128bit key expansion is done by using equation 4 and in the second phase, 128bit main key (K_m) is cyclically right shifted by 4bit and in the third phase, every 8bit vector of 128bit key expansion derived from step1 is intentionally altered by more than one bit inversion by modulo2 addition of shifted main key as shown in the figure1.

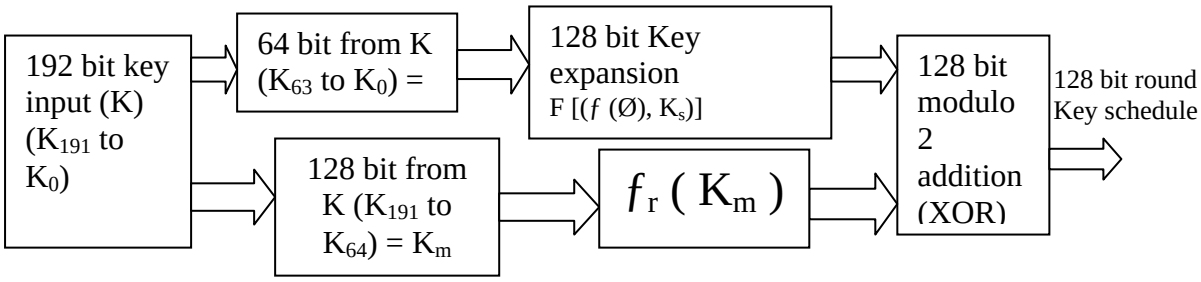

Fig. 1. 128 bit round key generation technique for the proposed design

In the figure1, fr (Km) is the function of 4bit right cyclic shift on the 128 bit main key (Km).

A. *Key expansion technique for the proposed design*

As shown from the figure 1, initially 64bit Sub-key (K_s) from original key (K) is expanded to another 64 bit for every 4bit of sub-key using the equation 4. This process is repeated for required number of iteration by considering output of 'i'th iteration to the input of 'i+1'iteration. For example, if the required total number of bits in the entire key schedule per encryption is 1280, then the required number of 64 bit key expansion iteration is 19 excluding the initial 64bit sub-key (K_s).

This process is reversible for every 4bit of 1280 bit key schedule iteration by considering output of 'i'th iteration to the input of 'i-1'iteration using the equation 5. Because of its symmetrical process, this technique can be easily adopted for 8bit processors.

B. *Key schedule generation for the proposed design*

As shown in the figure1, 128 bit output from iterations of key expansion is added with cyclic shift based main key (K_m) for the intentional bit inversion. This step assures that security strength on retrieval of round keys using cryptanalysis without the knowledge of original key.

C. *Application of the proposed design in cryptosystem with key-dependent S-box.*

The proposed technique for generation of round keys does not require any substitution boxes and round constant. However, it uses simple 4bit expansion technique as well as 8bit modulo2 addition. This facilitates the adoptability of the proposed design for memory restricted chip based cryptosystem. Few cryptosystems with key-dependent S-boxes have been proposed in the past such as Blowfish [6] and Khufu [7]. If the S-box elements are key dependent, usage of such s-boxes for generation of key schedules is more complex. Our design uses the channel coding theory to generate unique elements over GF(2^4) for round keys and it uses error control algorithm to randomized byte inversion without using any substitution and lookup table techniques.

VI. DESIGN CRITERIA FOR THE PROPOSED KEY SCHEDULES

The four design criteria comprises of simplicity, symmetry elimination, diffusion and non-linearity are considered for generation of key schedules in the proposed design.

A. *Simplicity*

Simplicity is considered in terms of consumption of working memory by key schedule generation algorithm and its performance in wide range of processor. As our design does not require Substitution box look-up table and has only 4bit logical expansion and 8bit modulo2 addition process, algorithm of proposed design needs only less working memory. So, it can comfortably be implemented on 8bit processor.

B. *Symmetry elimination*

Originality of the input key is removed by 64bit key expansion iteration technique. For example, 1280 bit of key schedule needs 19 times repetition of 64bit key expansion process.

C. *Diffusion and Non-linearity*

As shown in the figure1, necessary diffusion among key schedules is achieved by performing modulo2 addition with cyclic shifted main key (K_m). This function eliminates need for predefined round constant for diffusion. A high level of non-linearity is achieved by keeping ten number of round transformation per encryption as prescribed the existing Advance Encryption Standard.

VII. MATHEMATICAL AND CONSTRUCTIONAL FRAME WORK OF THE PROPOSED DESIGN

Let the 64 bit message to 64 bit parity generation by the equation 4 is represented by the function $f(m \rightarrow p) = f(Ø)$ and the 64 bit parity to 64 bit message generation by the equation 5 is represented by the function $f(p \rightarrow m) = f^{-1}(Ø)$ where $f(Ø)$ is the key generation function and $f^{-1}(Ø)$ is the inverse key generation function.

Let Sub-key $K_s = K_{63}$ to K_0 and Main-key $K_m = K_{191}$ to K_{64} from 192 bit original key (K) where $K = K_{191}$ to K_0

Let 4 bit of K_3 to $K_0 = I_0$ and its corresponding 4 bit parity bits are P_3 to $P_0 = R_0$.

Then, K_7 to $K_4 = I_1$ and its corresponding 4 bit parity bits are P_7 to $P_4 = R_1$.

Similarly, K_{11} to $K_8 = I_2$, K_{15} to $K_{12} = I_3$ …… K_{59} to $K_{56} = I_{14}$ and K_{63} to $K_{60} = I_{15}$ and its corresponding parity bits are P_{11} to $P_8 = R_2$, P_{15} to $P_{12} = R_3$ …… P_{59} to $P_{56} = R_{15}$ and P_{63} to $P_{60} = R_{15}$

Then the placement of 128 bits of the function $F[(f(Ø_i), (f(Ø_{i+1}))]$ from 4bit parity bits (R) and 4bit Sub-key bits (I) is illustrated in the figure 2.

R_{15}	I_{15}	R_{14}	I_{14}	R_{13}	I_{13}	…….	R_2	I_2	R_1	I_1	R_0	I_0
W_{15}		W_{14}		W_{13}		…… ..	W_2		W_1		W_0	

Fig. 2. 128 bit formation of function $F[(f(Ø), (f(Øi+1))]$ for key generation schedule

From the figure2, 'W' is the 8bit word formed by concatenation of 4bit parity bits (R) and 4bit Sub-key bits (I).

A. *Generation of key schedule for 128 bit encryption with 10 rounds.*

Key schedule for '0'th round is, $Ksh(0) = \{F[(f(Ø0), m0)] \text{ XOR } Km0\}$ (10)

where $f(Ø_0) = f(m_0 \rightarrow p_0)$, $m_0 = K_s$, $K_{m0} = K_m$ and 'XOR' is modulo2 addition

For the next round key schedule, $K_{sh}(1) = \{F[(f(Ø_1), f(Ø_2))] \text{ XOR } K_{m1}\}$ (11)

Where $f(Ø_1) = f(p_0 \rightarrow p_1)$, $f(Ø_2) = f(p_1 \rightarrow p_2)$ and $K_{m1} = f_r(K_{m0})$

Similarly for the 10th round key schedule, $Ksh(10) = \{F[(f(Ø19), f(Ø20))] \text{ XOR } Km10\}$ (12)

Where $f(Ø_{19}) = f(p_{18} \rightarrow p_{19})$, $f(Ø_2) = f(p_{19} \rightarrow p_{20})$ and $K_{m10} = fr(K_{m9})$

B. *Generation of inverse key schedule for 128 bit decryption with 10 rounds.*

Let 'p20' be the 64 bit input to the inverse key schedule generation process for decryption.

Then, $Ksh(10) = \{F[(f\text{-}1(Ø19), p20)] \text{ XOR } Km10\}$ Where $f\text{-}1(Ø19) = f(p20 \rightarrow p19)$ (13)

For the previous 9th round inverse key schedule, $Ksh(9) = \{F[(f\text{-}1(Ø17), f\text{-}1(Ø18))] \text{ XOR } Km9\}$ (14)

Where $f\text{-}1(Ø_{17}) = f(p_{18} \rightarrow p_{17})$, $f\text{-}1(Ø_{18}) = f(p_{19} \rightarrow p_{18})$ and $K_{m9} = f_L(K_{m10})$

In the equation14, $f_L(K_{m10})$ is the function of 4bit left cyclic shift on the 128 bit 10th right shift of main key (K_{m10}).

Similarly for the 0th round, $Ksh(0) = \{F[(f\text{-}1(Ø_0), f\text{-}1(Ø_{-1}))] \text{ XOR } K_{m0}\}$ (15)

Where $f\text{-}1(Ø_0) = f(p_1 \rightarrow p_0)$, $f\text{-}1(Ø_{-1}) = f(p_0 \rightarrow p_{-1}) = m_0$ and $K_{m0} = f_L(K_{m1})$

The function $F[(f\text{-}1(Ø_i), f\text{-}1(Ø_{i-1}))]$ is similar to the function $F[(f(Ø_i), (f(Ø_{i+1}))]$ since both function resembles the code word using the message and parity bits as shown in figure2.

VIII. GENERATION OF OUTPUT RESULTS OF THE PROPOSED DESIGN USING XILINX SIMULATION TOOL

The proposed algorithm is simulated using Xilinx ISE simulation tool with VHDL coding. The figure 3 shows the simulation output for the last round of key schedule.

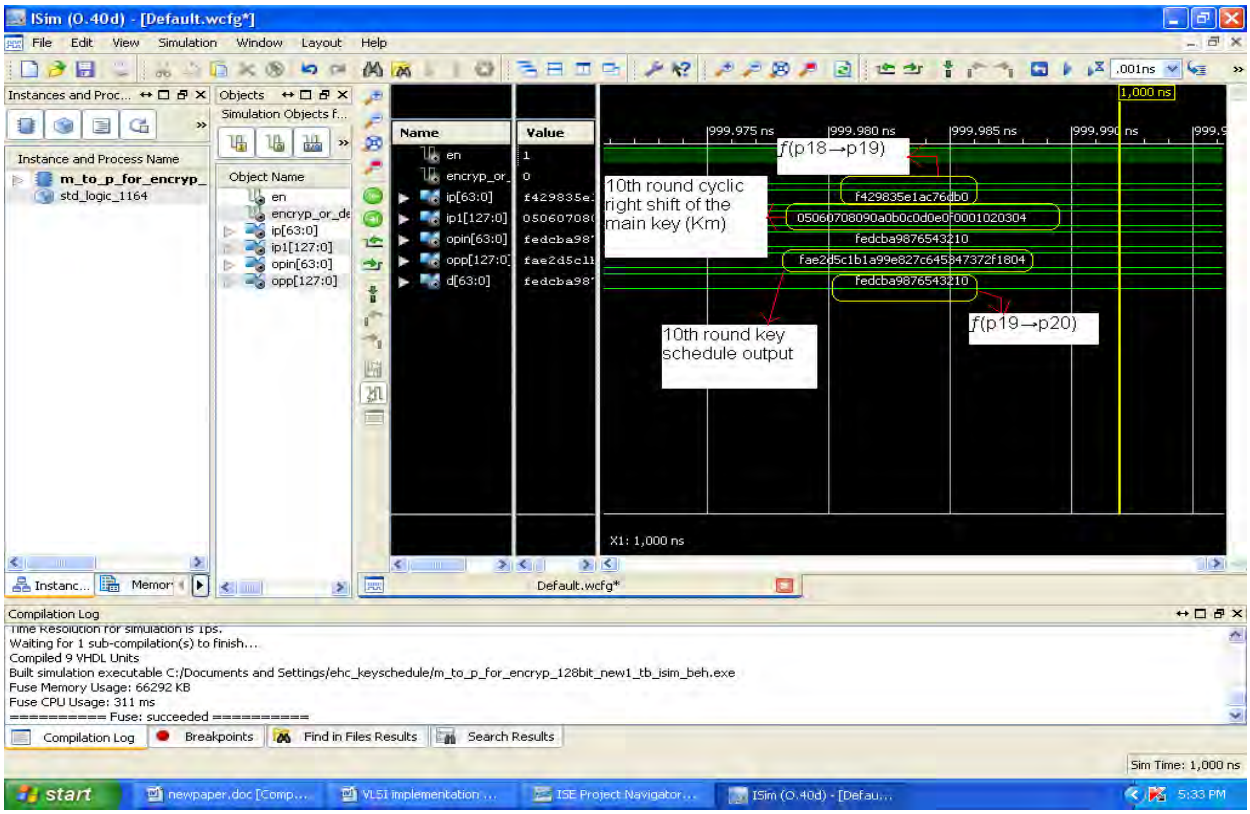

Fig. 3. Output generation of the proposed design using Xilinx ISE simulation tool

TABLE I.
Key schedules of proposed algorithm and Rijndael algorithm

Key Round number (r)	**Round Keys of 128bit Rijndael algorithm[8]**	**Round Keys of 128bit proposed algorithm**
	128 bit Key input (Kin): 000102030405060708090a0b0c0d0e0f	**128 bit main key input [Km] 000102030405060708090a0b0c0d0e0f**
		64 bit sub key input [Ks] f82516cb439ead70
0	000102030405060708090a0b0c0d0e0f	ffb9d09675335a1cecaac3856620490f
1	d6aa74fdd2af72fadaa678f1d6ab76fe	ff0b0df907f3f5010efafc08fe0204f0
2	b692cf0b643dbdf1be9bc5006830b3fe	fe73d158e16cce4fb13c9e17ae238100
3	b6ff744ed2c2c9bf6c590cbf0469bf41	ef671dd5de562c648b0379b1ba324800
4	47f7f7bc95353e03f96c32bcfd058dfd	fde7d6ccbea49d877b61504a38221b01
5	3caaa3e8a99f9deb50f3af57adf622aa	dfbe6d0c7b1a4928e786553443227110
6	5e390f7df7a69296a7553dc10aa31f6b	fcbcd793763e5511efafc480652d4602
7	14f9701ae35fe28c440adf4d4ea9c026	cf5b7da9372305d13eaa8c58c6d2f420
8	47438735a41c65b9e016baf4aebf7ad2	fb74d45bec63c34cb43b9b14a32c8c03
9	549932d1f08557681093ed9cbe2c974e	bf174de50ea6fc54db7329816ac29830
10	13111d7fe3944a17f307a78b4d2b30c5	fae2d5c1b1a99e827c645847372f1804

Table I shows the key schedule outputs generated by the proposed design in the right hand side and key schedules of the Rijndael [AES] algorithm is shown in the left hand side. From the table I, it can be concluded that any round key schedules of the proposed design would not reveal the originality of the main key. As our design uses the 64bit sub-key with 128bit main key, there are 2^{64} possible set of the key schedules for the same 128 bit main key. This is the major security strength of the proposed key schedule without any lookup table or substitution box.

IX. CONCLUSION

A novel implementation of key schedules by mathematical expressions over Galois Field GF (2^4) for private key cryptosystems was described. (8, 4) Extended Hamming Code and its error control logic was used to produce memory efficient key schedule generation algorithm. A mathematical relationship between 4bit code word and 4bit parity bits was shown for key expansion and its inversion technique. Different design criteria based on simplicity, symmetry elimination, diffusion and non-linearity of the proposed key expansion technique were described. Symmetrical method of algorithm was used for both encryption and decryption to reduce the working memory of the algorithm. High nonlinearity penetration of original input message bits was achieved by applying code based key schedules for each round transformations with cyclic shift register. Output results were shown for proposed design and Rijndael algorithm with the aid of Xilinx Simulation tool. It was concluded that random key generation by exploiting Error Controlling limit of channel coding could be memory efficient solution to the non-substitution based cryptosystems without any lookup table algorithm.

REFERENCES

[1] J. Daemen and V. Rijmen.: The Design of Rijndael, Springer, New York, NY, USA, 2002.

[2] S. Lucks.: Attacking seven rounds of rijndael under 192-bit and 256-bit keys, in Proceedings of the 3rd Advanced Encryption Standard Candidate Conference, pp. 215–229, New York, NY, USA, April 2000.

[3] F. J. MacWilliams and N. J. A. Sloane.: The theory of error correcting codes I and II, Amsterdam: North-Holland Publishing Co. North-Holland Mathematical Library, Vol. 16, 1977.

[4] L. Keliher: Linear Cryptanalysis of Substitution-Permutation Networks, PhD thesis, Queen's University, Kingston, Canada, 2003.

[5] X. Zhang and K.K. Parhi.: Implementation approaches for the advanced encryption standard algorithm IEEE Circuits Syst. Mag., 2(4), 24–46, 2002.

[6] B. Schneie.: Description of a new variable-length 64-bit block cipher, Fast Software Encryption, 191–204, 1996.

[7] R. Merkle.: Fast software encryption functions. In Advances in Cryptology: Procee- dings of CRYPTO'90. Springer-Verlag, Berlin, pp. 476–501, 1991.

[8] Rajender Manteena.: A VHDL implementation of AES – Rijndeal Algorithm, Thesis, University of South Florida, (2004).

[9] Pravin B. Ghewari., Jaymala K. Patil and Amit B. Chougule.: Efficient Hardware Design and Implementation of AES, Cryptosystem, International Journal of Engineering Science and Technology, 2010.

[10] Ayoub Otmani, Jean-Pierre Tillich. and L´eonard Dallot.: Cryptanalysis of TwovMcEliece Cryptosystems Based on Quasi-Cyclic Codes, arXiv: 0804.0409v3 [cs.CR], 2010.

[11] Keklik Alptek and Bayam Berna.: Differential power analysis resistant hardware implementation of the RSA cryptosystem,Turk J Elec Eng & Comp Sci, Vol.18., 2010.

[12] K.V. Pramod and C. Manju.: A Cryptosystem Using the Concepts of Algebraic Geometric Code, Journal of Computer Science 6 (3): 244-249, 2010.

[13] P. Kitsos N. Sklavos, M.D. Galanis and O. Koufopavlou.: 64-bit Block ciphers: hardware implementations and comparison analysis, VLSI Design Laboratory, Electrical and Computer Engineering Department, University of Patras, 26500 Rio, Patras, Greece, Computers and Electrical Engineering, 2004.

[14] Prasun Ghosal, Malabika Biswas and Manish Biswas.: Hardware Implementation of TDES Crypto System with On Chip Verification in FPGA, Journal Of Telecommunications, February, 2010.

[15] Gael Rouvroy, Jean-Jacques Quisquater and Jean-Didier Legat.: Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements and Design Tradeoffs, Springer-Verlag Berlin Heidelberg. 2003.

[16] J. Buchmann, R. Lindner, M. Ruckert and M. Schneider.: Post-quantum cryptography: Lattice signatures, pp. 147-191, Springer, 2009.

[17] Chetan Nanjunda Mathur, Karthik Narayan and K.P. Subbalakshmi.: On the Design of Error-Correcting Ciphers, Eurasip Journal on Wireless Communications and Networking, Volume, Article ID 42871, 2006.

[18] Francois-Xavier Standaert.: Secure and Efficient Implementation of Symmetric Encryption Schemes using FPGAs, Fran¸cois-Xavier Standaert, UCL Crypto Group. 2008.

[19] T. Hwang. and T. Rao.: Secret Error-Correcting Codes (SECC), In: Advances in Cryptography – Crypto 1988.

[20] Rajashri Khanai and Dr. G. H. Kulkarni.: Performance Analysis of Conventional Crypto-coding, International Journal of Latest Trends in Computing (E-ISSN: 2045-5364), Volume 2, Issue 1, 2011.