Improving diffusion power of AES Rijndael with 8x8 MDS matrix

R.Elumalai
Vinayaka Mission University
Salem, TamilNadu, India

Dr.A.R.Reddy
Department of Electronics and Communication
Madanapalli Institute of Technology and Science
Madanapalli, Andhra Pradesh, India

Abstract— AES Rijndael is a block cipher developed by NIST as the Advanced Encryption Standard (AES) replacing DES and published as FIPS 197 in November 2001 [5] to address the threatened key size of Data Encryption Standard (DES). AES-Rijndael was developed by Joan Daemen and Vincent Rijmen, Rijndael [4, 5] and was selected from five finalists. Advancement in computation speed every day puts lots of pressure on AES and AES may not with stand attack for longer time. This work focuses on improving security of an encryption algorithm, beyond AES. Though there are various techniques available to enhance the security, an attempt is made to improve the diffusion strength of an algorithm. For enhancing the diffusion power AES Rijndael in MixColumn operation the branch number of MDS matrix is raised from 5 to 9 using a new 8X8 MDS matrix with trade off of speed [8, 9] and implemented on R8C microcontroller.

Keywords- diffusion, MDS matrix, AES Rijndael, security

I. INTRODUCTION

The AES Rijndael algorithm basically consists of four byte oriented transformation for encryption and inverse transformation for decryption process over number of rounds depending on plain text size and key length namely [1,2],

1) Byte substitution (S-box) a non linear operation, operating on each of the State bytes independently.

2) Shifting rows (Row transformation) is obtained by shifting row of states cylindrically.

3) Mix Column transformation, the columns of the State are considered as polynomials over GF(28) and multiplied modulo $X^4 + 1$ with a fixed polynomial c(x), given by
$c(x) = \text{'03'}\, x^3 + \text{'01'}\, x^2 + \text{'01'}\, x + \text{'02'}$
The inverse of MixColumn is similar to MixColumn. Every column is transformed by multiplying it with a specific multiplication polynomial d(x), given by
$d(x) = \text{'0B'}\, x^3 + \text{'0D'}\, x^2 + \text{'09'}\, x + \text{'0E'}$.

4) Add round key, a Round Key is applied to the State by a simple bitwise EXOR. The Round Key is derived from the Cipher Key by means of the key schedule. The Round Key length is equal to the block length.

The round transformation in C pseudo can be written as [1]

```
 Round(State,RoundKey)
{
ByteSub(State);
ShiftRow(State);
MixColumn(State);
AddRoundKey(State,RoundKey);
}
```

The final round of the cipher is slightly different. It is defined by:

```
FinalRound (State,RoundKey)
{
ByteSub(State) ;
ShiftRow(State) ;
AddRoundKey(State, roundkey);
}
```

In AES Rijndael confusion and diffusion are obtained by non- linear S-Box operation and by the linear mixing layer over rounds respectively.

II. Diffusion in AES Rijndael

The linear mixing layer guarantees high diffusion over multiple rounds. Rijndael in his proposal approved by NIST replacing DES in the 2001 proposed MixColumn which operates on space of 4-byte to 4-byte linear transformations according to the following criteria[1,2]:
1. Invertibility;
2. Linearity in GF(2);
3. Relevant diffusion power;
4. Speed on 8-bit processors;
5. Symmetry;
6. Simplicity of description.

Criteria 2, 5 and 6 have lead to the choice of polynomial multiplication modulo x4+1. Criteria 1, 3 and 4 impose conditions on the coefficients. Criterion 4 imposes that the coefficients have small values, in order of preference '00', '01', '02', '03'…The value '00' implies no processing at all, for '01' no multiplication needs to be executed, '02' can be implemented using xtime and '03' can be implemented using
xtime and an additional EXOR. The criterion 3 induces more complicated conditions on the coefficients.

In Mix Column, the columns of the State are considered as polynomials over GF (2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial c(x) [1,2]. The Mix Column transformation operates independently on every column of the state and treats each sub state of the column as term of a(x) in the operating equation $b(x)=c(x)\otimes a(x)$, where c(x)= '03'X^3+'01'X^2+'01'X+'02'.This polynomial is co-prime to (X^4 + 1) For example, in the figure1. a(x) is $a_{0,j}X^3+a_{i,j}X^2+a_{2,j}X+a_{3}j$ and it is used as multiplicand of operation.

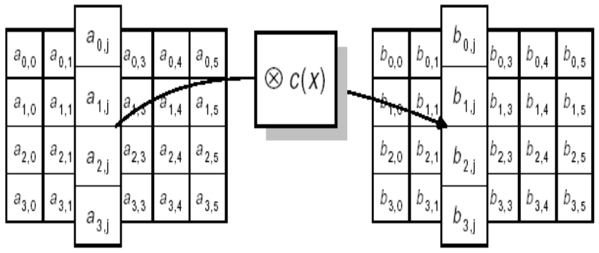

Figure1. MixColumn operation

This can be represented as

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Where c(x) has been represented by a circular matrix of 4x4.

Pseudo C code for Mixcolumn operation with Rijndael circular matrix is [10]

```
MixColumn (S)
{for (c=0 to3)
  Mixcolumn ($s_c$)
}
Mixcolumn (col)
    { copycolumn (col, t)   //t is temporary column
      $Col_0 \leftarrow (0x02) \bullet t_0 \oplus (0x03) \bullet t_1 \oplus t_2 \oplus t_3$
      $Col_1 \leftarrow t_0 \oplus (0x02) \bullet t_1 \oplus (0x03) \bullet t_2 \oplus t_3$
      $Col_2 \leftarrow t_0 \oplus t_1 \oplus (0x02) \bullet t_2 \oplus (0x03) \bullet t_3$
      $Col_3 \leftarrow (0x03) \bullet t_0 \oplus t_1 \oplus t_2 \oplus (0x02) \bullet t_3$
    }
```

Inverse of MixColumn in Rijndael uses a MixColumn transformation with different polynomial i.e.

$c(x) = \text{'03'}\, x^3 + \text{'01'}\, x^2 + \text{'01'}\, x + \text{'02'}$ for MixColumn and

$d(x) = \text{'0B'}\, x^3 + \text{'0D'}\, x^2 + \text{'09'}\, x + \text{'0E'}$ for inverse MixColumn

The Branch Number of a linear transformation is a measure of its diffusion power. Let F be a linear transformation acting on byte vectors and let the byte weight of a vector be the number of nonzero bytes [1,2]. The byte weight of a vector is denoted by W (a).

Definition: The branch number of a linear transformation F is

$\mathrm{Min}_{a\neq 0} (W ((a) + W (F (a)))$

A non-zero byte is called an active byte. For MixColumn it can be seen that if a state is applied with a single active byte, the output can have at most 4 active bytes, as MixColumn acts on the columns independently. Hence, the upper bound for the branch number is 5. The coefficients have been chosen in such a way that the upper bound is reached. If the branch number is 5, a difference in 1 input (or output) byte propagates to all 4 output (or input) bytes, a 2-byte input (or output) difference to at least 3 output (or input) bytes. Moreover, a linear relation between input and output bits involves bits from at least 5 different bytes from input and output.

III. OBJECTIVE OF THE PAPER

The Mixcolumn transformation of Rijndael so constructed has the property that the upper bound for the Branch number, which is 5, is reached. But the Inverse Mixcolumn transformation is not the same as the Mixcolumn transformation i.e. Mixcolumn does not have self inverse. This has led to degradation in performance on 8-bit processors for the Inverse cipher because it uses Inverse Mix Column and a modified Key schedule. Since the cipher and its inverse use two different transformations, a circuit that implements Rijndael does not automatically support the computation of the inverse of Rijndael and consumes more hardware when implemented on FPGA.

This work involves two important parameter for improving the diffusion strength of AES.

1. To find 8x8 matrix which has branch number 9 greater than 5 which is the case in AES Rijndael so that diffusion strength of the algorithm increases?
2. The matrix should be self inverse so that same matrix can be used for inverse MixColumn operation, which decreases the complexity of circuit and occupies less silicon area when implemented on hardware.

IV. MDS MATRIX

In AES linear transformations in the form of mappings based on Maximum Distance Separable (MDS) codes are used to achieve diffusion. A linear code over Galois field $GF(2^p)$ is denoted as an (n, k, d)-code, where n is the symbol length of the encoded message, k is the symbol length of the original message, and d is the minimal symbol distance between any two encoded messages[10].

Theorem 1: An (n, k, d)-code with generation matrix G = [I | C] is MDS if, and only if, every square sub matrix of C is nonsingular.

An (n, k, d)-code is MDS if d = n – k + 1. A (2k, k, k+1)-code with generation matrix G = [I | C], where C is a k×k matrix and I is an identity matrix, determines an MDS mapping from the input X to the output Y through matrix multiplication over a Galois field as follows:

$f_M: X \rightarrow Y = C \cdot X$ (1)

where

$$X = \begin{bmatrix} x_{k-1} \\ \vdots \\ x_0 \end{bmatrix} \quad Y = \begin{bmatrix} Y_{k-1} \\ \vdots \\ Y_0 \end{bmatrix} \quad C = \begin{bmatrix} C_{k-1,k-1} & . & . & . & C_{k-1,0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ . & . & . & . & . \\ C_{0,k-1} & . & . & . & C_{0,0} \end{bmatrix}$$

Each entry in X, Y, and C is an element in $GF(2^p)$. For a linear transformation, the branch number is defined as the minimum number of nonzero elements in the input and output when the input elements are not all zero.

There are different types of matrices which exhibit this property we will discuss two important types one used in Rijndael and other used in the work.

1. Circulant matrices: Given k elements $\alpha_0,\ldots,\alpha_{k-1}$, a circulant matrix A is constructed with each entry $A_{i,j} = \alpha_{(i+j)}$ mod k. The probability that a circulant matrix is suitable for an MDS mapping C is much higher than that of a normal square matrix.

   The Matrix used in Rijndael is a circulant MDS Matrix. In Rijndael, substitution permutation network (SPN) uses optimal non-involution MDS mappings. When an SPN uses a non-involution MDS mapping optimized performance only for encryption, the inverse MDS mapping used in decryption has a higher complexity

2. Hadamard matrices: Given k elements $\alpha_0, \ldots, \alpha_{k-1}$, a Hadamard matrix A is constructed with each entry $A_{i,j} = \alpha_{i\oplus j}$. Each Hadamard matrix A over a finite field has the following properties: $A^2 = \gamma \bullet I$ where γ is a constant. When $\gamma = 1$, A is an involution matrix. An involution MDS mapping is required by an involution SPN. When used in an SPN, the involution MDS mapping produces equally optimized performance for both encryption and decryption.

V. Mathematical Model

The matrix with high branch number and involution characteristics is found by applying Brute force method after arriving at required polynomial. The obtained polynomial H= had(01x, 03x, 04x,05x,06x,08x,0Bx,07x) Using this polynomial 8×8 matrix is constructed. This matrix is checked for the involution property. This matrix is constructed based upon the types explained above. Then the MDS property of the matrix is calculated. i.e. an (n, k, d)-code is MDS if d = n – k + 1.This can be done by checking the branch number of the transformation. The input with one or two active byte column is multiplied with the matrix and the output column is checked, if the total number of active bytes including input and output bytes is equal to 9 then it satisfies the property of MDS. Finally this matrix is found out and verified that it satisfies the involution property and the MDS property with (16, 8, 9) code.

The linear transformation matrix is $GF(2^8) \rightarrow GF(2^8)$ is a linear mapping based on the [16,8,9] MDS code with generator matrix G_H=[IH],
where H=had(01x, 03x, 04x,05x,06x,08x,0Bx,07x) is the polynomial found using brute force technique. The Hadamard matrix with involution property based on above polynomial shown below;

$$H = \begin{bmatrix} 01 & 03 & 04 & 05 & 06 & 08 & 0B & 07 \\ 03 & 01 & 05 & 04 & 08 & 06 & 07 & 0B \\ 04 & 05 & 01 & 03 & 0B & 07 & 06 & 08 \\ 05 & 04 & 03 & 01 & 07 & 0B & 08 & 06 \\ 06 & 08 & 0B & 07 & 01 & 03 & 04 & 05 \\ 08 & 06 & 07 & 0B & 03 & 01 & 05 & 04 \\ 0B & 07 & 06 & 08 & 04 & 05 & 01 & 03 \\ 07 & 0B & 08 & 06 & 05 & 04 & 03 & 01 \end{bmatrix}$$

A simple inspection shows that matrix H is symmetric and unitary. Therefore it is an involution transformation. It may be verified that this transformation has the branch number equal to 9 and also satisfies the criterion 3.

In Rijndael the state matrix to each function is considered as a 4×4 matrix. Here inside the MixColumn function the state matrix is converted into an 8×2 matrix and multiplied with the MDS matrix. Then the resulting 8×2 matrix is converted into a 4×4 matrix and passed to the next function.

The following Example describes about the multiplication of the state matrix with the above MDS matrix. Let the state matrix input to the MixColumn state be

$$\begin{bmatrix} 01 & 01 & 01 & 01 \\ 02 & 02 & 02 & 02 \\ 03 & 03 & 03 & 03 \\ 04 & 04 & 04 & 04 \end{bmatrix}$$

In MixColumn transformation the state matrix has to be multiplied with the standard matrix generated by the polynomial. In this scheme the multiplication of the matrix is performed with new 8x8 matrix generated by brute force technique which satisfies the involution property as shown below;

$$H=\begin{bmatrix} 01 & 03 & 04 & 05 & 06 & 08 & 0B & 07 \\ 03 & 01 & 05 & 04 & 08 & 06 & 07 & 0B \\ 04 & 05 & 01 & 03 & 0B & 07 & 06 & 08 \\ 05 & 04 & 03 & 01 & 07 & 0B & 08 & 06 \\ 06 & 08 & 0B & 07 & 01 & 03 & 04 & 05 \\ 08 & 06 & 07 & 0B & 03 & 01 & 05 & 04 \\ 0B & 07 & 06 & 08 & 04 & 05 & 01 & 03 \\ 07 & 0B & 08 & 06 & 05 & 04 & 03 & 01 \end{bmatrix} X \begin{bmatrix} 01 & 01 \\ 02 & 02 \\ 03 & 03 \\ 04 & 04 \\ 01 & 01 \\ 02 & 02 \\ 03 & 03 \\ 04 & 04 \end{bmatrix}$$

$$= \begin{bmatrix} 08 & 08 \\ 3F & 3F \\ 2E & 2E \\ 1D & 1D \\ 08 & 08 \\ 3F & 3F \\ 2E & 2E \\ 1D & 1D \end{bmatrix}$$

The state matrix after the Mix Column transformation is

$$\begin{bmatrix} 08 & 08 & 08 & 08 \\ 3F & 3F & 3F & 3F \\ 2E & 2E & 2E & 2E \\ 1D & 1D & 1D & 1D \end{bmatrix}$$

The difference between the diffusion powers of the Mix column step of Rijndael algorithm and the new proposed Mix column are explained in the Figure 2 and 3. At the end of first round, active bytes are 8 in case of 8x8 MDS, whereas it is 4 in case of 4x4 MDS. This shows an encryption algorithm designed using 8x8 MDS will provide more security compared to 4x4 MDS. However, this is achieved at the cost of additional computation.

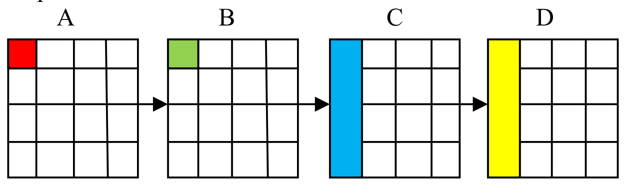

Figure 2 - Diffusion states for encryption in the AES in the first round. A – S-BOX, B – Shift Row, C – MixColumn and D – ADDRoundKey

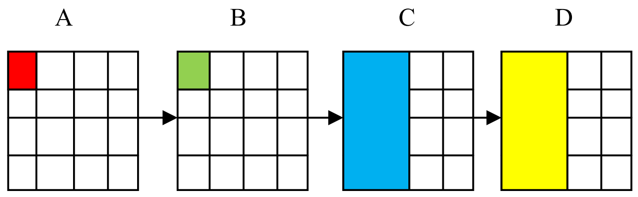

Figure 3 - Diffusion states for revised encryption with 8x8 matrix in the AES in the first round. A – S-BOX, B – Shift Row, C – MixColumn and D – ADDRoundKey.

VI. IMPLENTATION ON MICROCONTROLLER

A. Renesas R8C microntroller

The R8C/Tiny Series of single-chip microcomputers was developed for embedded applications by Renesas. The R8C/Tiny Series supports instructions tailored for the C language, with frequently used instructions implemented in one-byte op-code. It thus allows development of efficient programs with reduced memory requirements when using either assembly language or C. Furthermore, some instructions can be executed in a single clock cycle, enabling fast arithmetic processing.

R8C has features like CPU core operating at 20MHz, on chip ROM,RAM, and data Flash, programmable I/O ports, 9 interrupts with 7 priority levels, 14 bit watch dog timer, 3 timers, 4 UARTs, synchronous communication port, I^{2}C bus, LIN module, USB, 10 channel 10 bit ADC and 2 comparators which makes it most preferred industrial application microcontroller.

For implementation on R8C microcontroller, Renesas High performance Embedded workshop V.4.07.01 with simulator version 4.1.04.00 which is provided by Renesas was chosen for convenience. It provides integrated development environment composed of compiler and simulator also. Every cycle number and code size output depends on embedded workshop.

B. *Simulation result*

Code was run on the Renesas High performance embedded workshop V.4.07.01 with simulator version 4.1.04.00 with test vector from Brian Gladman's technical paper [11]. The implementation was optimized many times.

Module	Cycle	Code (Byte)
Precomputation	2178	1649
ByteSub	115	32
ShiftRow	51	256
MixColumn	201	134
AddRoundKey	127	48
branch	19	12
Total	8276	2135
(round0-10)	5917	784

Figure 4 – Simulation results for 10 round AES.

The total number includes consideration of the number of rounds. In this implementation, the message block and key length are 128-bit each. Therefore 10 rounds constitute one encryption procedure. In the 10 rounds of encryption procedure, precomputation is executed just once and other modules have all different execution times as in Cycle column of the Table2. Whereas, code length weight factor is irrelevant with cycle number weight factor because some modules are reused every time while the other modules are not. For example, AddRoudKey module executed at round0, round1-9 and round10 but MixColumn executed only at round1-9, which makes the different code weight factor 3 and 1 respectively.

Module	Cycle	Code (Byte)
Precomputation	1	1
ByteSub	10	2
ShiftRow	10	2
MixColumn	9	1
AddRoundKey	11	3
branch	1	1

Table2. Weight factor of each module

With weight factor total number of cycle and code are computed by equation below.

Total cycle number = $\sum$Cycle number(i) * weight factor(i)

Total code size =$\sum$Code number(i) * weight factor(i)

Where ' i' is every module

Module	Cycle	Code (Byte)
Precomputation	2178	1649
ByteSub	115	32
ShiftRow	51	256
MixColumn	243	149
AddRoundKey	127	48
branch	19	12
Total	8376	2195
(round0-10)	6171	824

Figure 4 – Simulation results of revised 10 round AES with 8x8 MixColumn..

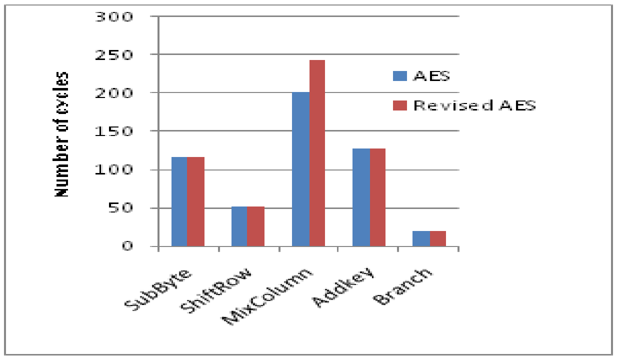

Figure 4 – Simulation results of revised 10 round AES with 8x8 MixColumn..

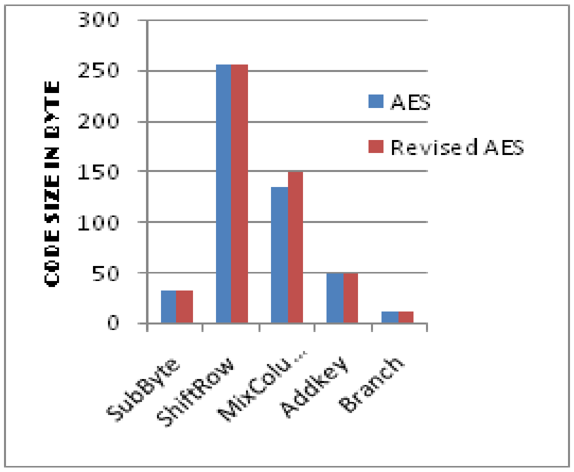

Figure 5 – Comparison of number of code consumed by two schemes.

VII. Conclusion

The MixColumn module is still takes much part of cycle number. This module is also the critical part for register scheduling because it need 8 multiplication with 8 different sub state at the same time while keeping their initial state.

The result shows the number of cycle required is 20.08% more and code area consumed is 11.19% more in the case of revised AES with 8x8 MixColumn compared to AES Rijndael. The increase in cycle and code memory is the trade off for the increase in diffusion strength which increases the security of the algorithm.

References

[1] Daemen and V. Rijmen, AES Proposal: Rijndael (Version 2). NIST AES
[2] NIST, Advanced Encryption Standard (AES), (FIP PUB 197), November 26, 2001
[3] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[4] K. Ohkuma, H. Muratani, F. Sano, and S, Kawamura, "The Block Cipher Hiero-crypt", Workshop on Selected Areas in Cryptography. SAC 2000, Lecture Notes in Computer Science 2012, Springer-Verlag, pp. 72-88, 2001.
[5] P. Barreto and V. Rijmen, "The Anubis Block Cipher", NESSIE Algorithm Submission 2000, available on: www.cosic.esat.kuleuven.ac,be/nessie.
[6] P. Barreto and V. Rijmen, "The Khazad Legacy-Level Block Cipher", NESSIE Algorithm Submission, 2000, available on: www.cosic.esat,kuleuven.ac.be/nessie
[7] A. Rudra, P.K. Dubey, C.S. Jutla, V, Kumar, J. R. Rao, and P. Rohatgi, "Efficient Rijndael Encryption Implementation with Composite Field Arithmetic", Cryptographic Hardware and Embedded Systems - CHES 2001, Lecture Notes in Computer Science 2162, Springer-Verlag, pp. 171-184, 2001
[8] Lu Xiao and Howard M. Heys “Hardware Design and Analysis of Block Cipher Components”
[9] Aarti Singh “Study of MDS Matrix used in Twofish AES algorithm and its VHDL implementation” M.Tech thesis.
[10] Behrouz A.Forouzan “Cryptography and network security “ TATA-Mcgraw hill publication 2007 edition.
[11] A Specification for Rijndael, the AES Algorithm v3.3, Brian Gladman, May 2002
[12] P. Barreto and V. Rijmen, “The Khazad Legacy-Level Block Cipher”, NESSIE Algorithm Submission, 2000, available on: www.cosic.esat.kuleuven.ac.be/nessie.

[13] R. Anderson, E. Biham, and L. Knudsen, “Serpent: a Proposal for the Advanced Encryption Standard”, AES Algorithm Submission, available on: www.cl.cam.ac.uk/¢rja14/serpent.html
[14] A. Youssef, S. Mister, and S. Tavares, “On the Design of Linear Transformations for Substitution-Permutation Encryption Networks”, Workshop on Selected Areas in Cryptography - SAC '97, Ottawa, 1997.

AUTHORS PROFILE

Dr.A.R.Reddy is professor in the department of Department of Electronics and Communication Engineering, at Madanapalli Institute of Science and Technology, Madanapalli, Andrapradesh, India. He received M.Tech and Ph.D from IIT Kharagpur, India. His field of interest is Cryptogharaphy, Embedded system and VLSI.

R.Elumalai is Ph.d student in Department of Electronics and Communication Engineering, at Vinayaka Mission University, Salem, Tamilnadu, India. He received M.Tech from University Visvesvaraya College of Engineering, Bangalore University. India. His field of interest are Embedded system, VLSI and cryptography.