LEA 192: High Speed Architecture of Lightweight Block Cipher

Zeesha Mishra, Shubham Mishra, Bibhudendra Acharya

***Abstract:** High-throughput lightweight cryptography calculation is the need of the present world to convey between two asset obliged devices Pipelining is the technique have been used to achieve high throughput. In this paper we have target to lightweight block cipher LEA. Block size of LEA is 128 and key size 128, 192, and 256 bit. In this paper we have focus on LEA architecture for 192- bit key size and achieve very good throughput. This method has a higher capability of throughput as compared to previous LEA ciphers. Proposed work is 56% improved version of compared paper for respective Speed and area also less than previous architecture. Graph representation have been shown of different matrices and comparison.*

***Keywords** : ARX, Block Cipher, Cryptography, LEA, Lightweight, Pipeline, Throughput.*

I. INTRODUCTION

The information technology sector is growing at a tremendous pace and has found its application in field of multimedia. The increased usage of smartphone technology has invoked researchers to provide more hardware features and complex applications. Further, highly secure applications such as net banking, online shopping, surfing social media sites and many others have become a part and parcel of today's life. Cryptography allows a person to transmit and store data securely with high privacy. Modern-day cryptography utilizes a key to convert valid data into gibberish so that it cannot be revealed. Then the same data is deciphered at the authenticated recipient through either same key or a different key. For the resource constant application cryptography terns in "lightweight" cryptography. Lightweight cryptography algorithms are specially tailored for IoT RFID tags and WSN. IoT has been one of the most spread platform of the future technology and is getting sufficient attention from a wide range of real-life applications[1] principles having a network of interconnected devices that are interacting with each other and the surroundings to collect and analyse information using the internet. According to some estimates, over 30 billion connected things with more than 200 billion intermittent connections. The communication among connected things must be secure and fast while processing confidential data. Due to a substantial amount of sensitive and valuable data and low computational resources in the network, it has become cumbersome to provide security in the IoT[2].

Resource-constrained applications comprising of sensor nodes in wireless sensor networks, RFID tags require smaller footprint with minimum battery consumption.

LEA[3] is a lightweight block cipher. LEA has 128-bit block size with 128, 192, 256-bit key sizes and rounds 24, 28 and 32 respectively. LEA supports simple ARX (addition rotation XOR) operation, LEA is not using complex S-box structure like AES [4] hence it will give high-speed operation.LEA is independent of all attacks for block ciphers, which makes it more suitable for those places where we need both security with high performance. LEA uses some arbitrary constants in its algorithm which makes it more efficient to provide confusion and diffusion in encryption. Choosing LEA-192 over LEA-128 is having an important purpose here, as key size increases security of system also improves hence key size 192 provides more security as compared to 128 bit key size.

Lightweight cryptography is the implementation of efficient algorithms with as low as possible resources. Hence they are mainly useful in implementation of resource-constrained devices because they require small area high throughput and fast processing. To create a lightweight cipher it is necessary to use less shows the high performance architecture of lightweight block cipher LEA with 192 bit key size, which has increased area and less number of rounds. Some of the Lightweight cryptography algorithms are like PRESENT[5], Piccolo[6], LEA[2], LiCi[7], QTL[8], KATAN KATANTAN[9] etc.

A. Contribution

This proposed work a throughput as compared to previous architecture. We have used Pipelined architecture to increase the throughput to achieve better performance. Which we have used to get those results. Virtex-4 and Virtex-6 devices and compare with different algorithm.

B. Organization of paper

The persisting paper is consist of total V sections. Section II is the LEA lightweight block cipher algorithm for 192 bit key size. Section III is hardware implementation of LEA lightweight block cipher. Section IV is the discussion of result in different devices and comparison with existing works. At the last section V is the conclusion part of the paper which is comprised of all work done so far in this proposed paper.

Revised Manuscript Received on December 15, 2019.

* Correspondence Author

Zeesha Mishra*, Department of Electronics and Communication Engineering, National Institute of Technology, Raipur, India. Email: zmishra.phd2016.etc@nitrr.ac.in

Shubham Mishra, Department of Electronics and Communication Engineering, National Institute of Technology, Raipur, India. Email: shubhammishra.0709@gmail.com

Bibhudendra Acharya, Department of Electronics and Communication Engineering, National Institute of Technology, Raipur, India. Email: bacharya.etc@nitrr.ac.in

II. THE LEA LIGHTWEIGHT BLOCK CIPHER ALGORITHM FOR 192-BIT KEY SIZE

LEA is an asymmetric lightweight block cipher announced in 2013, LEA cipher having 128 bit of block size and the key size of 128 bit, 192 bit, 256 bit. LEA lightweight block cipher is based on ARX operations (Addition, Rotation and XOR), these operations are fast and compatible at 32 bit and 64 bit platforms.

According to the size of key which are 128 bit, 192 bit and 256 bit, there are fixed rounds respectively 24, 28 and 32. As we increase the size of key or number of rounds, strength of encryption improves. Key Generation process provides round keys for each round and size of round key always will be of 192-bit which will be expended from main key. ARX operations happening here on 32 bit parts of algorithm which are P presents plaintext P[0],P[1],P[2] and P[3] and C presents ciphertext C[0],C[1],C[2] and C[3]) and rotated constant constants Y^*.

TABLE I. **NOTATIONS USED IN OPERATIONS**

Notations	Meaning
P	Plaintext
C	Ciphertext
K^*	j^{th} key schedule state
Y*	Rotated Constants Value
RL	Rotation Left
RR	Rotation Right
$\oplus$	XOR Operation
$\boxplus$	Addition modulo 2^{32}

I. **Key scheduling process**: Section ***A*** given below generates round keys R0K_j after processing ($0 \leq j<28$) rounds. In each round of operation, it performs the ARX operations on the main key, and stores it until the number of rounds completes. Each time M stores values, which are 1 bit shifted version of performed result. When any round ends, all values present in Kwill be stored in round key RK_j LEA uses some constants (γ*) which provides better confusion in the process of key scheduling., LEA uses same function for each round.

II. **Encryption Process:** In this process, round function calculation given in section ***B*** does the encryption part using all-round keys RK values and constants values in key scheduling process. Round keys generated in the key scheduling process, will be in operation with plaintexts here. In this section we will process 128-bit plain text and Same operation works for 32 rounds and at last whatever value will be stored in it will be our result as ciphertext. Here RL and RR are rotate operation, which performs here 9 bit left and, 5and 3 bit right shift operation. At last C will be the values of T.

TABLE II: PRESENTED IN ROTATED CONSTANT VALUES

Before Rotation Constant value Y	**After Rotation constant value Y^***
0x3efe9db	0x3efe9db
0x44626b02	0x88c4d604
0x79e27c8a	0xe789f229
0x78df30ec	0xc6f98763
0x715ea49e	0x15ea49e7
0xc785da0a	0xf0bb4158

A. Key Generation using 192-bit key

```
1. $K \leftarrow K^*, Y \leftarrow Y^*$
2. for j : 0 to 28
$K[0] \leftarrow RL1(K^*[0] \boxplus RLi(Y^*[j \bmod 6]))$
$K[1] \leftarrow RL3(K^*[1] \boxplus RL_{i+1}(Y^*[j \bmod 6]))$
$K[2] \leftarrow RL6(K^*[2] \boxplus RL_{i+2}(Y^*[j \bmod 6]))$
$K[3] \leftarrow RL1(K^*[3] \boxplus RL_{i+3}(Y^*[j \bmod 6]))$
    $K[4] \leftarrow RL13(K^*[4] \boxplus RL_{i+4}(Y^*[j \bmod 6]))$
$K[5] \leftarrow RL117(K^*[5] \boxplus RL_{i+5}(Y^*[j \bmod 6]))$

3. $RK_j \leftarrow K[0], K[1], K[2], K[3], K[4], K[5]$
4. End for
5. Return RK
```

B. Round Function Calculation

```
1. $X0 \leftarrow P0, X1 \leftarrow P1, X2 \leftarrow P2, X3 \leftarrow P3;$
2. for j : 1 to 28
3. $X_{j+1[}0] \leftarrow RL9((X_j[0] \oplus RK_j[0]) \boxplus (Xj[1] \oplus RK_j[1])),$
$X_{j+1}[1] \leftarrow RR5((X_j[1] \oplus RK_j[2]) \boxplus (X_j[2] \oplus RK_j[3])),$
$X_{j+1}[2] \leftarrow RR3((Xj[2] \oplus RK_j[4]) \boxplus (Tj[3] \oplus RK_j[5])),$
$T_{j+1}[3] \leftarrow Tj[0],$
4. End for
5. $C \leftarrow T$
6. Return C
```

III. HARDWARE IMPLEMENTATION OF ARCHITECTURE

Pipeline registers have been used to improve the performance of the design and which is the main purpose of this proposed paper.

Section A- Key Scheduling

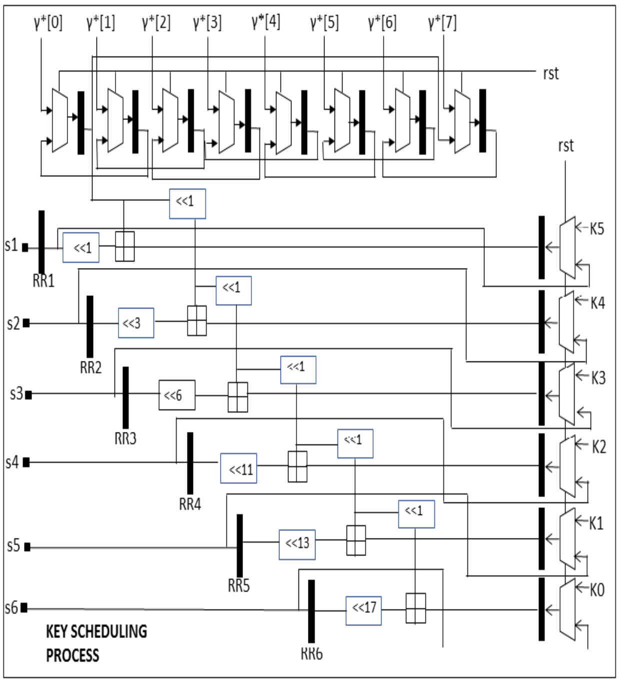

Fig. 1 Proposed Pipeline Architecture key scheduling

Section B- Round processing

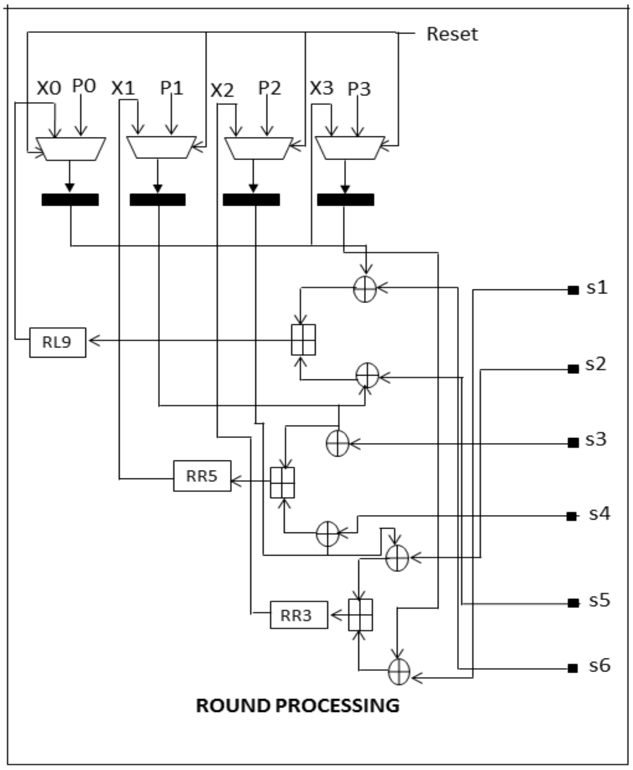

Fig. 2 Proposed Pipeline Architecture Round function

As we can see in architecture divided into two sections first one is key scheduling process and second one is round function, for the connection of the two parts of architecture we gave interconnect node which names are s1, s2, s3, s4, s5, s6. In key scheduling process Y*[0] to Y*[7] is rotated constant key and K0 to K5 are constant key are adding with 32-bit module operation. RR1 to RR6 are rotated register 1 to 17 bit left rotation operation are perform in this registers.

In round function P0 to P3, 32 bit plaintext are used as ain Input. In this section round function are uses XOR operation modulo addition operation and left and right rotation also performed.

IV. RESULTS AND DISCUSSIONS

Table II is the result for Virtex-6 on device xc6vhx255t-3ff1155. These results shows values regarding LUTs, registers, slices, Max frequency, latency, Throughput and power dissipation. Virtex-6 we have used for proposed result and we have compared our result at the same platform at which previous works have been done.

To provide better performance of the algorithm we have uses pipeline architecture in critical path for batter efficiency, and it gave a good speed as compared to previous LEA 192- bit architecture. In this paper we have use states 128-bit plaintext and 192-bit key .Maximum frequency provided by this implementation is very high and speed due to performance of pipelining and Virtex-6 device.

Speed we have achieved in this paper is improved version of previous and having a value of 1927.25, and because of that its throughput per slices value is also increased. Slices shows area of design which is very less in numbers as compared to same key size LEA papers. Using X-power Analyzer we can calculate the value of static and dynamic powers in Watts.

The family used to calculate all these parameters is Virtex- 6 and corresponding device is xc6vhx255t-3ff1155 , where -3 is the speed of the package ff1155. Without pipelining results of speed is very less as compared to using pipelining.

Throughput calculation will be using maximum frequency, block size and clock cycle of the design.

Equation (1) presents the calculation of throughput.

$$\text{Throughput} = \frac{\text{MF}\times\text{BS}}{\text{Cycle}} \qquad (1)$$

MF represents the max frequency, and BS presents the block size.

One more parameter which we have compared here in this paper is to calculate here is throughput per area, below is the equation 2 to calculate throughput per area.

$$\text{Throughput/area} = \frac{\text{Throughput}}{\text{Area (Slices)}} \qquad (2)$$

Comparison of the given design LEA-192 and different cryptographic cipher algorithms have been shown in Table III. we can see that for same design family, proposed works performance is better.

Throughput defines the number of bits passing through a system or process. Fig. 3 given here shows the graphical representation of slices we have got using different platforms in Xilinx, Similarly Fig. 4 given here shows the graphical representation of comparison in maximum frequency of proposed architecture. Fig. 5 shows the power comparison at same platforms like maximum frequency and slices.

Fig 5 having better speed as compared to previous done. Proposed work have been implemented on different devices of family Virtex-5 and Virtex-4, naming xc5vlx50t and xc4vfx12 and spartan-3.

Table- I: Virtex-6 results of the proposed architecture

S.No.	xc6vhx255t-3ff1155	
	Variable	Value
1	No. of LUTs	469
2	No. of Registers	804
3	No. of Slices	222
4	**Max.Frequency @ 100**	**496.87**
5	Cycle	33
6	**Throughput**	**1927.25**
7	Throughput/area	8.68
8	Static Power	4.274
9	Dynamic Power	0.267
10	Total Power	4.541

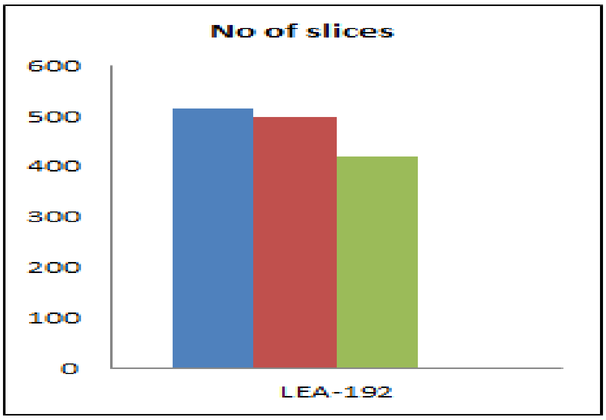

Fig 3. Slices comparison with different dsigns

It is result of using efficient pipeline at ideal position in architecture. As this value of throughput per area increases, design provides better performance.

Fig. 5 shows the speed or maximum frequency comparison of different lightweight ciphers which we have already shown in Table. I . This figure is graphical representation of the comparison in speed, which shows that proposed architecture of LEA-192 is having 59% improvement in speed as compared to LEA[10] of hardware efficient paper.

In all matrices field result of the proposed architecture is better as compared to previous work done on the same cipher or many other ciphers. Results we have calculated on FPGA which is an existing model.

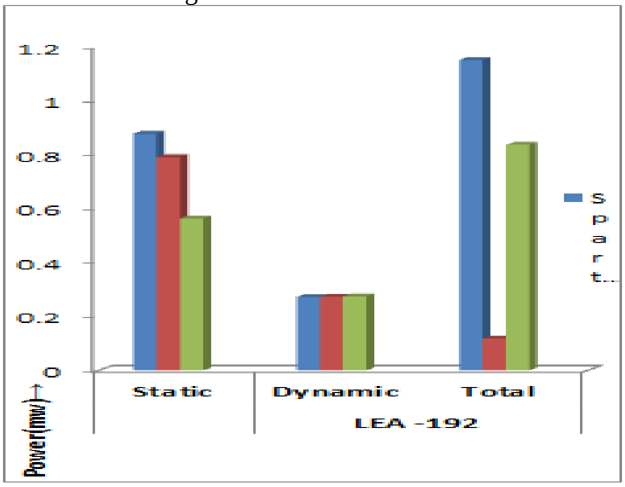

Fig 4. Power comparison with different dsigns

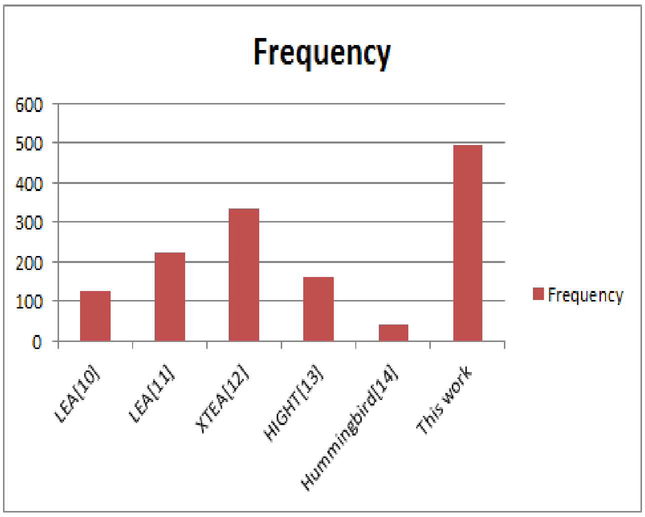

Fig 5. Frequency comparison with different ciphers

Different comparison have been shown to define the comparative analysis of this paper. Slice comparison in Fig.3, Power comparison in Fig .4 and Frequency comparison in Fig 5 vlearly shows the better result of proposed architecture. Power calculation have been compared on all values as static power, dynamic power and total power and in all fields this proposed architecture shows better results.

Table- II: Comparison of synthesized hardware utilization of different algorithms

S.No.	Algorithm	Device Name	LUTs	Registers	Latency	Max frequency	Through put	Thr/Area
1	LEA [10]	xc5vlx330t	1131	645	32	126.23	505	0.071
2	LEA[11]	xc5vlx50t	360	382	25	225.023	1152.11	3.016
3	XTEA[12]	xc5vlx85	-	-	192	333	111	2.14
4	HIGHT[13]	-	-	-	160	163	65.2	0.72
5	Hummingbird[14]	xc3s200	473	120	20	40.17	160.4	0.59
7	**This Work**	**xc5vlx50t**	**432**	**694**	**29**	**340**	**1500**	**7.65**
8	**This Work**	**Xc4vfx12**	**525**	**694**	**29**	**284**	**1252**	**2.51**
9	**This Work**	**xc3s200**	**525**	**704**	**29**	**131.91**	**582**	**1.33**

V. CONCLUSION

In this paper we have presented a high speed pipelined architecture of LEA-196, where 196 is the key size we have chosen to encrypt the block of 128 bit. We have compared our result with the some previous architecture with respect to its speed and FOM (Thr/Area), by comparing we can see that given architecture provides better metrics as compared to them hence this is improved architecture. Also LEA is resistance to all existing cipher block attacks. ARX operations are better and faster as compared to Fiestel and SPN architectures, as LEA uses ARX operations hence we can see that its operation is faster as compared to AES, s-box operation. At last we can conclude that proposed architectures throughput is 1500.11, 1252.41 and 582.22 Mbps respectively for Virtex-5, Virtex-4 and Spartan-3. Hence using pipelined architecture have provided us a throughput which is better with this latency and complexity. As we will use advanced devices of FPGA, results and matrices will be improved more.

REFERENCES

1. J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.
2. J. Choi, S. Seok, H. Seo, and H. Kim, “A fast ARX model-based image encryption scheme,” *Multimed. Tools Appl.*, vol. 75, no. 22, pp. 14685–14706, 2016.
3. D. Hong, J. K. Lee, D. C. Kim, D. Kwon, K. H. Ryu, and D. G. Lee, “LEA: A 128-bit block cipher for fast encryption on common processors,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8267 LNCS, pp. 3–27, 2014.
4. J. S. Banu, M. Vanitha, J. Vaideeswaran, and S. Subha, “Loop parallelization and pipelining implementation of AES algorithm using OpenMP and FPGA,” *2013 IEEE Int. Conf. Emerg. Trends Comput. Commun. Nanotechnology, ICE-CCN 2013*, pp. 481–485, 2013.
5. C. A. Lara-Nino, A. Diaz-Perez, and M. Morales-Sandoval, “Lightweight Hardware Architectures for the Present Cipher in FPGA,” *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 64, no. 9, pp. 2544–2555, 2017.
6. K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, “Piccolo: An ultra-lightweight blockcipher,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6917 LNCS, pp. 342–357, 2011.
7. J. Patil, G. Bansod, and K. S. Kant, “LiCi: A new ultra-lightweight block cipher,” *2017 Int. Conf. Emerg. Trends Innov. ICT, ICEI 2017*, pp. 40–45, 2017.
8. L. Li, B. Liu, and H. Wang, “QTL: A new ultra-lightweight block cipher,” *Microprocess. Microsyst.*, vol. 45, pp. 45–55, 2016.
9. C. De Cannière, O. Dunkelman, and M. Knežević, “KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5747 LNCS, pp. 272–288, 2009.
10. D. Lee, D. C. Kim, D. Kwon, and H. Kim, “Efficient hardware implementation of the lightweight block encryption algorithm LEA,” *Sensors (Switzerland)*, vol. 14, no. 1, pp. 975–994, 2014.
11. Z. Mishra, G. Ramu, and B. Acharya, *Architecture for LEA Encryption Algorithm*. Springer Singapore.
12. J. Kaps, “Chai-tea , Cryptographic Hardware Implementations of xTEA.”
13. P. Yalla and J. P. Kaps, “Lightweight cryptography for FPGAs,” *ReConFig’09 - 2009 Int. Conf. ReConFigurable Comput. FPGAs*, pp. 225–230, 2009.
14. X. Fan, G. Gong, K. Lauffenburger, T. Hicks, and W. Drive, “FPGA Implementations of the Hummingbird Cryptographic Algorithm,” *2010 IEEE Int. Symp. Hardware-Oriented Secur. Trust*, pp. 48–51, 2010.

AUTHORS PROFILE

Zeesha Mishra, has earned her B.E. in Information Technology and M. Tech. (2015) in VLSI Design from Chhattisgarh Swami Vivekananda Technical University, Bhilai India. She is presently pursuing her Ph.D. in Department of Electronics and Telecommunication Engineering, National Institute of Technology Raipur. Her research interests include algorithm and hardware architecture design for digital image and signal processing and embedded system design**.**

Shubham Mishra has received his B-Tech degree in Electronics and Communication (2017) and now pursuing his M-Tech in VLSI and Embedded System from National Institute of Technology Raipur, India. His research interest includes Cryptographic ciphers, Algorithms and Hardware architectures. He has published his one conference in the same field in IEEE international conference.

Bibhudendra Acharya received the M. Tech. and Ph.D. degree in Electronics and Telecommunication from National Institute of Technology, Rourkela, India. He is currently an Assistant Professor in the Electronics and Telecommunication Engineering Department, National Institute of Technology, Raipur, India. His research interests include Cryptography and Network Security, Microcontroller and Embedded system, Signal Processing, Mobile Communication. He has more than 65 research publications in National/ International Journals and conferences**.**