Implementation of Efficient Bit Permutation Box for Embedded Security

NISHCHAL RAVAL, GAURAV BANSOD, DR. NARAYAN PISHAROTY
Electronics and Telecommunication
Symbiosis Institute of Technology, Symbiosis International University
Lavale, Pune, Maharashtra
INDIA
nishchal.raval@sitpune.edu.in, gauravb@sitpune.edu.in, narayanp@sitpune.edu.in

Abstract: - Security in every real time applications is of utmost importance. The secure architecture implemented in the automobiles such as EVITA (E-safety Vehicle Intrusion protected Application), SEVECOM (Secure Vehicle Communication) has rich cryptographic properties, but has more footprint area and high power consumption. This existing architecture uses standard engines like AES (Advanced encryption standard), Elliptical curves, Hash Engines which are heavy in memory requirement and consumes more power. So its reach is limited only to high end systems that consisting of large bit processors and coprocessors. Role of a bit permutation instruction in cryptographic environment is well proven. GRP (Group Operations) and OMFLIP (Omega-Flip) networks are bit permutation instructions and its implementation in hardware not only accelerates software cryptography but also results in less footprint area and low power consumption. This paper proposes a novel implementation and analysis of GRP and OMFLIP architecture for security in small scale embedded networks. In this paper a hybrid implementation is analysed and its results are compared with 'P' box of standard algorithms like AES and DES (Data Encryption Standard). This paper shows that GRP needs very less memory space as compared to other bit permutation instructions and will be useful to design lightweight ciphers.

Key-words: - OMFLIP, GRP, Embedded Security, Automotive, Encryption, Bit permutation

1 Introduction

Every system in the digital world is connected to many nodes and servers through which a large volume of data is exchanged and stored. This data can be transferred from one node to another over a wired or wireless medium. As data is transferred from one node to other it is open to threats from the environment [1]. Many systems are connected to external world through internet, Bluetooth or by any other media. This opens a wide gateway for mounting an attack and disrupts the operation. Data or information security is therefore an area of grave concern [2]. In application like automobiles where large number of microcontrollers are used popularly known as electronic control units (ECU's) and these ECU's communicate through a popular bus called CAN (Control Area Network) bus. CAN bus is extensively researched bus and the results from the papers shows that frames inside the bus can be manipulated to disrupt the communication or to have intended communication, that could results in damaging a system [3]. In present automobiles, there are at least 30 to 40 microcontroller that can communicate with each other over a CAN bus. This communication between nodes must be encrypted, so that only an authenticate controller can participate in communication. With the growing need to secure this communication a number of encryption standards and algorithms have been developed, designed and implemented to make a system secure. For automobile security projects like EVITA, SEVECOM have given deep insight in the area of secure communication. These models are based on threat to automobile communication from inside as well as from external environment [4]. These models uses heavy cryptographic algorithms like AES -128 (Advanced Encryption System), Hash engines, Elliptical Curve based system which are standard algorithms and are approved and endorsed by NIST (National Institute of Standard and Technology) [5]. For these algorithms and engines, cryptanalysis has been done and attacks are not yet proven. These cryptographic engines have a huge footprint area which makes the system slower and thus reduces throughput. Because of huge memory requirement, they are overkill for 8 bit systems [6]. As technology is advancing the need and necessity in cryptography is towards less

bandwidth, less area and low power consumption [7]. In embedded systems accurate information must be conveyed with lesser number of bits and low power to the destined node without falling prey to any attacks [3]. Based on the need, an approach is needed to introduce a secure design that has less memory requirements, less footprint area and less power consumption.

A role of bit level permutation in cryptographic environment is well proven. This bit permutation instruction has good cryptographic properties, is easy to design and have good flexibility. The instructions like GRP (Group operations) and OMFLIP (Omega and Flip) are bit permutation instruction whose implementation in hardware not only provides good cryptographic design, but also accelerates software cryptography [8]. Introduction of bit permutation instructions in algorithms like RC5 has improved the performance in terms of memory consumption and also results in higher throughput. OMFLIP network is a fast permutation network as compared to GRP [9]. These network consist of different stages of 2x1 multiplexers which are used for swapping bits based on control words applied as select lines to each multiplexer. Selection of 2x1 multiplexers results in low power consumption as compared to using heavy multiplexers. In this paper, we are focussing on GRP and OMFLIP network and its implementation on 32 bit processor. GRP has been shown to be resistant to linear and differential cryptanalysis [10] and is best suited for key generation for embedded security as compared to other standard key generation algorithms. OMFLIP is used to speed up the performance of fixed permutation. In OMFLIP, 'n' stages of omega and flip are used. It consumes less power and memory space and has less latency which increases throughput of system. OMFLIP is the best suited for fast permutation operation to make good cryptographic design in small scale embedded system for light weight encryption.

By using bit permutation instruction design can be hardware efficient as it saves memory. Lightweight cryptography is the emerging field where new or existing ciphers are used with less number of RAM and ROM bytes. The applications like in Internet of Things (IOT), specifically in RFID tags one cannot implement AES 128 as memory requirement for this standard algorithm is huge. Total number of Gate Equivalents (GEs) available in RFID would be till 10000. It is infeasible to use the block cipher which consumes nearly 3000 GEs alone for providing security. The structures which are designed with the help of bit permutation instructions are always hardware efficient and needs less memory space as compared to other instructions like look up table. Bit permutation instructions are complex in nature that makes them obvious choice to use in cryptographic environment.

2 Cryptographic Properties of GRP and OMFLIP

Bit permutation instructions are complex subword permutation that makes them best suitable in securing an environment. The use of bit permutation instructions like GRP and OMFLIP is useful to design efficient and secure ciphers. Ciphers like DES, SERPERNT and TWO-FISH uses bit permutation instruction in its operation. Bit permutation instructions are very effective in obtaining diffusion operation [10]. GRP instruction can perform any *n* bit permutation but no longer than log (*n*) steps. OMFLIP network uses omega and flip stages and can perform permutation of n bits by omega-flip network of 2log (*n*) stages. Two omega stage and two flip stages are sufficient to implement OMFLIP instruction. Flip stage is exactly opposite to Omega stage in performing permutation of any word size permuting 2n bits on n bit systems. GRP is more efficient than any other bit permutation instruction in key generation and has shown resistant to differential attacks. GRP and OMFLIP are complex instructions and are having edge over simple permutation instructions like rotation, look up tables. The fig. 1 from paper [10] shows differential properties of bit permutation instructions.

Operation	Type A $(e_s,0) \rightarrow e_t$	Type B $(0, e_t) \rightarrow j$	Type C $(e_s, e_t) \rightarrow j$
GRP	$0 < i \leq 1/2 + 1/2^n$	For any t, $E(\|j\|) = n/4$	For any t, $E(\|j\|) = n/4$
OMFLIP	$i = 1/4$ or 0	$\|j\| = 0$ or 2	$\|j\| \leq 1$ or 3

Fig. 1: Differential properties of bit permutation instruction

Type A characteristics shows that "how the single bit is moved around when control bits are randomly chosen". Type B and C shows the "diffusion effect which is compared by computing the expected Hamming weight of the output difference 'j' ". A large Hamming weight indicates that operation is more effective and causes the desired avalanche effect. GRP helps to achieve avalanche effect that the small change in an input causes a lot of change in the bit position at the output. GRP has shown better differential properties because of the paths of data bits depends on all the

control bits. Changing a single bit at the input will change the bit positions at the output. OMFLIP does not possess good Type B and Type C characteristics as compared to GRP. Fig. 2 shows linear properties of bit permutation instructions [10].

Operation	Type L (e_s, 0, e_t)	Type M (e_s, e_u, e_t)
GRP	$k \leq 1/4 + 1/2^{n+1}$ Maximum with s = t = 0	$k \leq 1/4 - 1/2^{n+1}$ Maximum with s = u = t = 0
OMFLIP	$k \leq 1/8$ Maximum with s = t = 0	$k \leq 1/8$ Maximum with s = u = t = 0

Fig. 2: Linear properties of GRP and OMFLIP

Fig. 2 shows OMFLIP has a little better linear property than GRP. Hence, GRP has better differential property while OMFLIP has good linear property. Their combination can help positively to avoid both types of attacks.

3 Key Generations by GRP

Key Generation is the core part of cryptographic environment. Key should be long enough to avoid attacks. DES has the disadvantage of shorter keys, but because of that some attacks has been proven on the system. AES is known for its 128 bit and 256 bit versions and till today because of its robust architecture, it is difficult to mount an attack. AES uses RCON matrix where it store hexadecimal number in matrix forms and generate key with the help of Rijndael cipher. It adds round keys and performs a total of 10 rounds of substitution and permutation network. AES network is heavy and infeasible for small scale embedded processors [11]. GRP gives the compact solution on hardware and would be useful for providing security to small bit embedded processor. Key generation in GRP is based on the bit positions. Different bit positions will generates different control words [12]. If 8 bit positions are introduced we will have 256 different combinations. Each 8 bit combination generates 3 control words of length 8 bits.

GRP consist of total 5 stages for key generation that are mentioned below:

1. User defined bit positions.
2. MISes (Monotonically Increasing sequence)
3. Generating Ti (Temporary sequence)
4. Formation of Qi (Sorting of Ti)
5. Generating Cj (Generate Control bits)

The fig. 3 shows key generation by GRP which is illustrated in paper [9][13].

Iteration	1	2	3
P	(7,6,5,4,3,2,0,1)	(3,7,2,6,0,1,5,4)	(0,1,3,5,7,2,4,6)
MISes in P	(7)(6)(5)(4)(3)(2)(0,1)	(3,7)(2,6)(0,1,5)(4)	(0,1,3,5,7)(2,4,6)
From Alg. 1, Ti	(7,3)(6,2)(5,0,1)(4)	(3,7,0,1,5)(2,6,4)	(0,1,3,5,7,2,4,6)
From Alg. 1, Qi	Q=(3,7)(2,6)(0,1,5)(4)	Q = (0,1,3,5,7)(2,4,6)	Q = (0,1,2,3,4,5,6,7)
From Alg. 1, Cj	C = 10101100	C = 11010010	C = 00101010

Fig. 3: Control word generation by GRP

GRP algorithm can generate the different keys at different rounds from one integer sequence. The number of rounds depends on the number of bits for example, for 8 bit integer sequence, the rounds are 3 from $2^3 = 8$.

3.1 Implementation of GRP Algorithm on 32 Bit Processor

In this paper we discuss and presented implementation of GRP on a 32 bit processor ARM7. Universal structure of GRP is designed which generates control word for different bit positions. The algorithm is coded in embedded C and results are shown through 8 bit UART module. In this paper UART module act as demonstrator with 9600 baud rate. Fig 4 indicates the output of key generation which is implemented in *"Keil 4.0"* simulator by using UART serial window of LPC2129. 128 bit key is generated with GRP with input as 16 hexadecimal values.

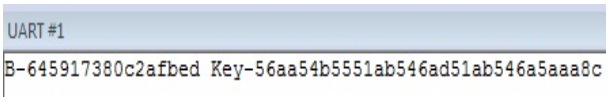

Fig. 4: Key generation display on UART

Fig. 4 indicates the initial 'B' which represent bit position integer sequence as input to the GRP algorithm. They are the Hex values. User can make any possible combination of integer sequence by using Hex values. It will generate the key based on that integer sequence.

These hex values have to be converted in integer format from 0 to 127. So, if the integer value is 'F' then it contains highest value "1111" binary value and the number we will assign on exactly the same bit position will be as "63 62 61 60". Similarly, for 'E' = "1110" the integer sequence will be "59 58 57 56". By assigning integer sequence, we have generated 64 integer values by giving only 16 letters or integers.

The integer sequence will be arranged in reversed order. As shown in Fig. 4, reverse order integer sequence is "9e8d4b0613fc2a57". Then, if the integer value is 'F', it contains highest value

"1111" and the number is assigned exactly at the same bit position as "127 126 125 124". Similarly, for 'E' = "1110" the integer sequence will be "123 122 121 120". By assigning integer sequence in the mentioned way, we have generated 128 integer values by giving only 16 letters or integers.

GRP has random integer numbers from 0 to 127. So, it can generate 128 bit keys. In the Fig 3, "Key" indicates the 128 bit keys in Hex value form that are 32 Hex values and these keys generated are as per GRP algorithm. For 128 bit key size, GRP has to go seven rounds only and these seven rounds can generate seven keys, each of 128 bit size. It will give 7 different 128 bit keys. It is difficult to mount attacks, because inputs are alphanumeric integer and each of its combination generates a huge number which is difficult to predict. Similarly GRP can generate 64 bit based on input bit positions.

4 Encryption-Decryption by using OMFLIP Network

OMFLIP has better linear properties than GRP and know for fast bit permutation operation. OMFLIP is having minimum latency as compared to other bit permutation instructions [9]. OMFLIP offers the fastest bit permutation operation with a minimum delay. In the Omega stage and the Flip stage, ω bits are divided into (ω/2) pair bits. In the Omega stage, the bits i and (i , i + (ω/2)) are mapped to the bit positions of 2i and (2i+1). At the same way, in the Flip stage, the bits 2i and (2i+1) mapped to bit positions of i and (i , i + (ω/2)). The Omega stage is exactly the inverse of the Flip stage. Any bit permutation till 'n' stages is possible by using only these two stages.

The OMFLIP operation $Z = X_{(a)} \cdot Y$ uses two stages (omega stage – flip stage) to permute the data bits X (input data) and Y (control bits) for two stages. The subscript part (a) represents the stage, a=0 for Omega stage and a=1 for flip stage. These give the four possible states of permutation (Omega, Flip), (Omega, Omega), (Flip, Omega) and (Flip, Flip) [13].

4.1 Encryption by OMFLIP Design

The fig. 5 shows the 8 bit cryptographic operation using OMFLIP. There are two Omega stages and two Flip stages. OMFLIP cryptographic design contains 2x1 Multiplexer (which indicates by "M" in the figure). The concept of using only 2x1 multiplexer is that it consumes less power as compared to other structures. Moreover, low power design is the necessity of small scale embedded processor [10].

The input bits are fed to the Omega stage. The output of omega stage will become the input of the next flip stage. Different combinations of omega and flip stages can be chosen to give different cryptographic design. In this paper we have implemented flexible and universal structure of OMFLIP network that is based on 2 Hex value input by user. Permutation operation and stages will be decided based on these hex values and different encrypted result can be produced. User need to give 2 Hex values which indicates 8 bit input. For example 2 Hex values is "34" means "0011 0100", the function and stages selected will be Omega, Omega, Flip, Flip, Omega, Flip, Omega and Omega. Out of these maximum seven stages needed for performing 128 bit encryption. Block size can be varied and stages also can be selected based on the 2 digit hex value.

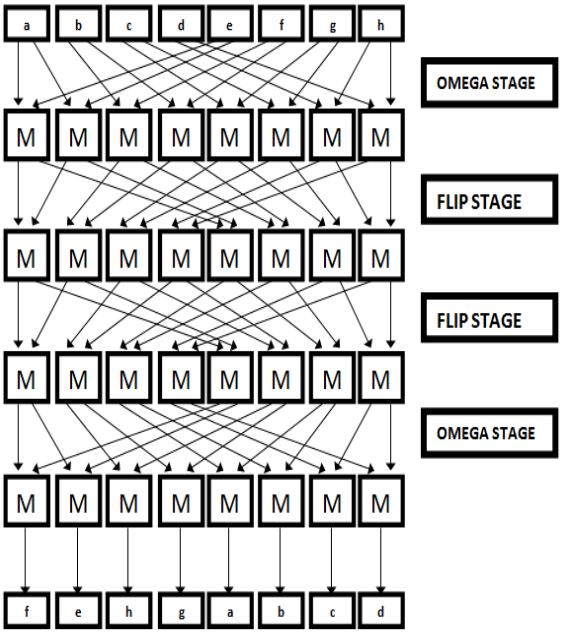

Fig. 5: Encryption operation by using 2x1 multiplexer

4.2 Decryption using Inverse OMFLIP Stages

The desire encrypted data from the OMFLIP based on the user selected input stages, will be transmitted at receiver end where reverse operation is performed to get the original data. The fig. 5 shows the 8 bit cryptographic permutation using Inverse OMFLIP. There are two Omega stages and two Flip stages

similar to OMFLIP operation at transmitter end [9]. The fig. 6 indicated decryption for 8 bit block size.

At decryption, a = 0 for flip stage and a = 1 for omega stage exactly opposite to OMFLIP encryption design. The mapping of bits is same as of transmitter. However, the control bits should be applied also in the reverse order. It means the last control bit used at the encryption stage will be the first control bits in the first stage of Inverse OMFLIP at receiver end. If there is '1' in control bits, swapping will be done between mapped bits position otherwise it remain unchanged [9][13].

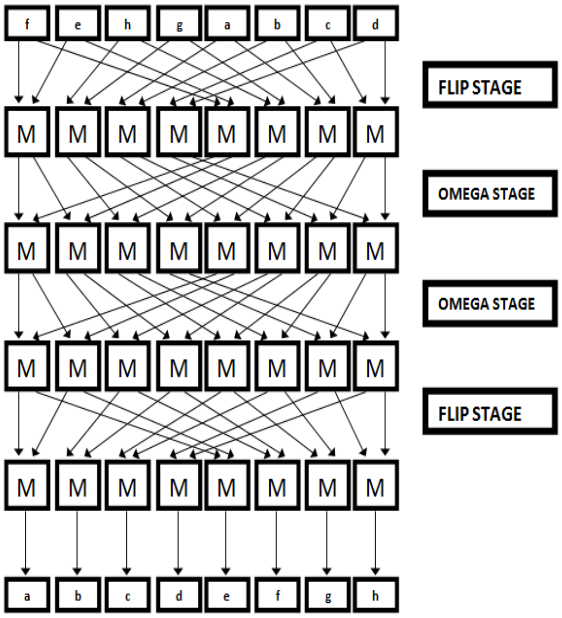

Fig. 6: Decryption operation by using 2x1 multiplexer

OMEGA Stage:

In OMEGA stage bits i and (i + ω/2) will be mapped to 2i and 2i+1, where ω is half the number of bit position. Table 1 shows bit positions, data and control bits for 8 bit OMFLIP stage.

Table 1: Mapped data and control bits

Bit Position	0	1	2	3	4	5	6	7
Data	a	e	b	f	c	g	d	h
Control Bits	1	0	1	0	0	1	1	0

These four control bits are responsible for doing permutation.

If the

$$y\,[j + ((a*\omega)/2)] = 1 \quad \text{- (1)}$$

Where 0 < i < (ω/2) and a = 0 for OMEGA Stage which is described through equation number (1), means if the control bit is '1' corresponding bits will be swapped. 2i and 2i+1 bit positions will get swapped which is shown in equation number (2).

$$\text{Swap } (2i, 2i+1) \quad \text{- (2)}$$

The output of OMFLIP stage, after applying control bits is shown in Table 2.

Table 2: OMFLIP stage permutation

Bit Position	0	1	2	3	4	5	6	7
Data	a	e	b	f	c	g	d	h
Control Bits	1		0		1		0	
O/P	e	a	b	f	g	c	d	h

FLIP Stage:

For Flip stage same permutation and mapping is performed, the only difference is for swapping is that we have to consider the last four bits of control bits.

Table 3: Mapped data and control bits

Bit Position	0	1	2	3	4	5	6	7
Data	e	b	g	d	a	f	c	h
Control bits	1	0	1	0	0	1	1	0

These last four bits are responsible for swapping the bits.

If the

$$y\,[j + ((a*\omega)/2)] = 1 \quad \text{- (3)}$$

Where 0 < i < (ω/2) and a = 1 for FLIP Stage which is described by equation number (3), then the mapped bits will be swapped between each other based on below logic. i and i + ω/2 bit positions will be swapped which is shown with the help of equation number (4).

$$\text{Swap } (i, (i + (\omega/2))" \quad \text{- (4)}$$

The output of flip stage, after applying control bits, is shown in Table 4. It shows swapping of data based on last four control bits which are 0110. Positions will be swapped only when control bit is '1', otherwise bits will retain their original positions. OMEGA stage is exactly opposite to FLIP stage. For permutation in OMEGA, first four bits are considered for swapping and in FLIP last four bits are considered for swapping. Swapping among bits will be done only if the corresponding control bit is '1'.

Table 4: FLIP stage permutation

Bit Position	0	1	2	3	4	5	6	7
Data	e	b	g	d	a	f	c	h
Control Bits	0		1		1		0	
O/P	e	f	c	d	a	b	g	h

4.3 OMFLIP Operation

OMFLIP does the following operations to permute the bits.

Arranging*:* In this module, the 'n' bit data is arranged in array form. This block act as input stage to mapping stage.

Mapping*:* After arranging, the bit positions are mapped to the new bit positions which is based on either omega or flip stage

Control bits*:* The control bits are the key part of the OMFLIP network. It is used for swapping the new bit position bits with each other. It contains the bits length as same as the data length [10]. In OMFLIP network, the first four bits are considered for swapping at the omega stage. Similarly, the last four bits are considered for swapping at the flip stage. Exactly, the reverse logic in inverse OMFLIP network. In Inverse OMFLIP network, the first four bits are considered for swapping at the flip stage. And the last four bits are considered for swapping at the omega stage.

Swapping*:* After applying control bits, based on respective control bits of omega stage and flip stage, the swapping is done between those data that are just mapped.

Rearranging*:* After swapping the mapped bits with each other, the output of data array, which is in *n* bit form, is stored as input of next stage till the number of stages, which are already defined, will be finished [8][9][13].

The Fig. 7 indicates the OMFLIP algorithm and its flowchart.

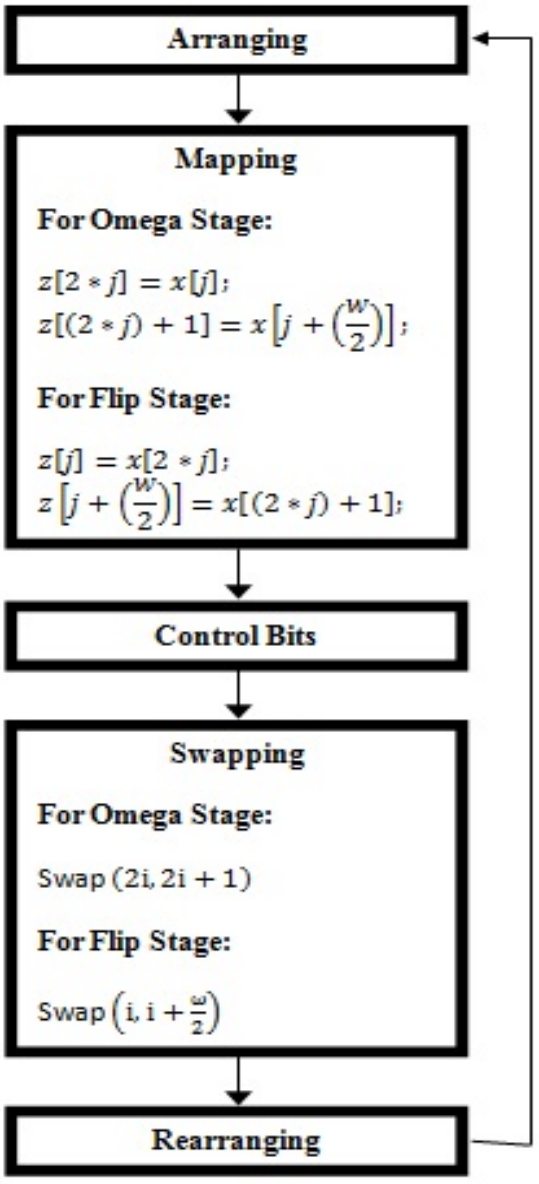

Fig.7: Flow chart for OMFLIP network

5 Hybrid Cryptosystem by using GRP and OMFLIP

GRP has better differential property while OMFLIP has good linear property, the combination of these two algorithms will make good permutation box that can be suited for light weighted embedded security [9][10]. The OMFLIP has 128 bit inputs for Encryption-Decryption and GRP will generate 128 bit (16 Hex values) key based on 16 bit hex value. OMFLIP-128 works exactly the same as OMFLIP-8 network. GRP generates the key based on 16 hex values and it is fed to OMFLIP network which performs encryption. User has to enter two hex values to decide number of stages of OMFLIP network. Two hex value results in 8 stages OMFLIP network which consisting of OMEGA and FLIP state. If user inserts 4F as input then system take it as 01001111 which selects three OMEGA and four FLIP stages. '0' represent OMEGA stage while '1' represents FLIP stage. OMFLIP network has less latency as compared to GRP [10] that is why it is used for performing encryption. GRP is used for key generation as it has shown good resistance to linear and differential cryptanalysis. This hybrid implementation is implemented on 32 bit processor LPC2129 by using KEIL 4.0 simulator. The fig. 8 shows block diagram of AES-128 which is implemented in Embedded C on LPC2129 at bit level. All standard algorithms in this paper we have implemented and compared on the same platform.

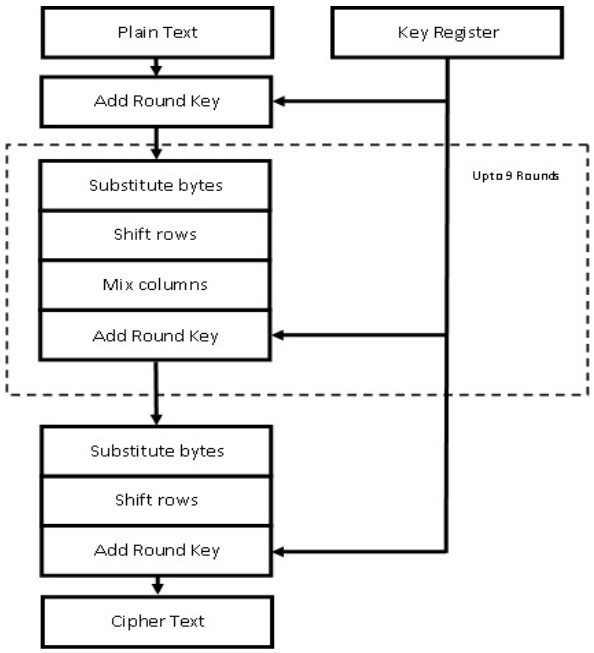

Fig. 8: AES-128 block diagram

In order to compare this hybrid system, we have also implemented AES-128 bit and DES on ARM processor. The fig. 8 shows comparative study of AES, DES, with our hybrid cryptosystem.

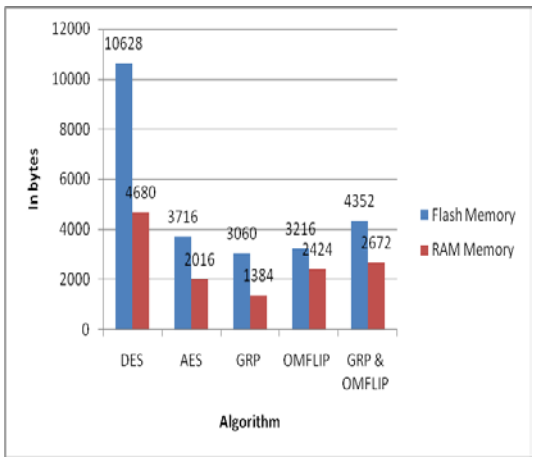

Fig. 9: Comparison with standard algorithm

From the Fig. 9, GRP and OMFLIP memory requirement is clearly depicted which requires around 3060 and 3216 bytes while AES needs 3716 and DES needs 10628 bytes of Flash memory. As GRP and OMFLIP have only diffusion properties, in Fig 9, only 'P' box of standard algorithms is compared with GRP and OMFLIP. 'P' box of GRP consumes very less memory among all algorithms which is around 1520 bytes of RAM while AES 'P' box results in 2384 bytes of Flash memory which is higher than GRP and near about same as OMFLIP. To the best of our knowledge this is the most optimized version of GRP which need 3060 bytes of Flash memory. From the fig. 10, 'P' box of GRP and OMFLIP does only permutation while in Fig 9 GRP and OMFLIP does key generation and encryption both individually that results into higher memory requirement.

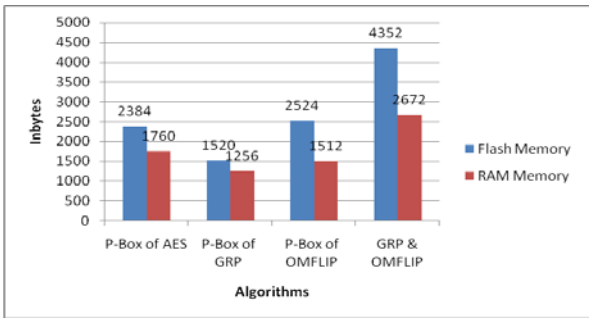

Fig. 10: 'P' box comparison

Bit permuation instructions are well known for compact hardware implementation. Even in the

lightweight cipher like PRESENT, bit permutation instructions are used in designing its 'P' box. The fig. 9 also depicts the memory required for implementation of GRP 128 bit would be very less compared to other bit permuatation instructions and also with AES. This work and result clearly shows a importance of bit permutation instructions in designing lightweight ciphers. By using GRP, one can optimized cipher memory requirements, thus helping reducing power consumption and number of processsing elements like transistors.

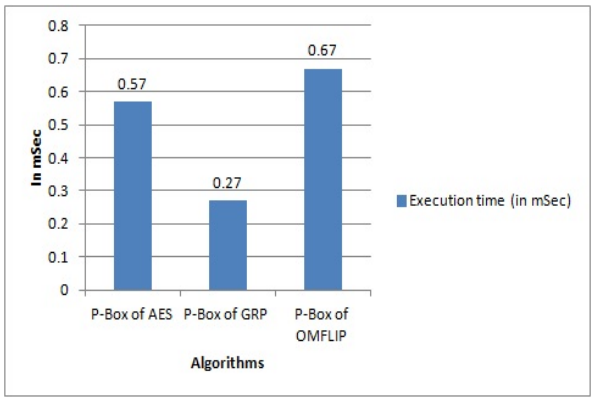

Fig. 11: 'P' box Comparison of execution time with standard algorithm

Fig 11 shows performance of GRP, OMFLIP and comparison with AES on 32 bit processor with clock frequency as 60Mhz. GRP takes only 0.27 milliseconds which is least as compared to other bit permutation instruction. OMFLIP have higher execution time as compared to AES and GRP due to number of Omega and Flip stages. Fig 11 depicts efficiency of GRP over other standard algorithms in terms of execution time.

D	E	F	G	H
On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	40	7168	1
Signals	0.000	60	---	---
IOs	0.000	36	141	26
Leakage	0.060			
Total	0.060			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	35.2	82.9	27.1

Fig. 12: Power consumption

Fig 12 shows power consumption of GRP bit permutation instruction which comes around 60milliwatts which is best suitable for small scale embedded applications. Power consumption is calculated based on Xilinx Xpower tool. For this hardware implementation and power consumption, code is written in Verilog. Previous papers on bit permutation instructions shows around 200-300milliwatts power consumption [12].This power consumption also includes method for reducing side channel attacks by using dual in rail package(DRP) by using balance instructions[21].Differential power analysis(DPA) attacks can be reduced by using balanced instructions.

6 Discussions and Conclusion

Role of bit permutation instructions in cryptographic environment is indisputable. Implementation of these instructions will accelerate not only cryptography, but also results in faster throughput, less power consumption and less footprint area. In the past, bit permutation instruction have been suggested and implemented for multimedia processing. Group operations (GRP), Omega flip (OMFLIP) networks are the more popular network in cryptographic environment. These networks require less memory space, less power consumption and have good cryptographic properties. Comparison shows OMFLIP architecture performs faster than GRP and requires less footprint area, while GRP has a minimum delay and can be implemented with different techniques by modifying or inserting elements in an existing basic model. These instructions are having edge over the existing secure architecture in terms of area, power and throughput of a system.

In this paper, OMFLIP design is made to be more flexible and robust. User can select the number of stages and which are the stages will be included in design and in what order, based on these parameters OMFLIP design will produce encrypted results. Moreover control words that are applied to each stages of OMFLIP are varied randomly each time [13]. This makes design more robust and induced good cryptographic properties. At the decryption end, the reverse process is applied and original data will be retrieved out. As in block ciphers, same key is used for decryption which is used at encryption level, but this key is stored at the receiver side. In this paper, we are dynamically during run time. We are sending encrypted data and the keys which are generated randomly at transmitter end to perform permutation. This avoids tampering of data at receiver end as keys are not stored but increases a channel over head. To the best of our knowledge, this is the first implementation of OMFLIP design on 32 bit processor. OMFLIP

design will be preferred because of less memory requirements and less power consumption for small scale embedded system security which operates on 8 to 16 bits of data. This design provides good performance with less power consumption which is generally a constraint for small hand held devices. OMFLIP can perform permutation for 'n' bits with maximum of log (*n*) steps.

OMFLIP hardware implementation performs faster than GRP as there are fixed number of stages, so size of 'n' can be of any length. Latency is also less as compared to GRP permutation and it completes permutation in one machine cycle [8][9][13]. These all parameters will provide an edge to OMFLIP network over GRP. OMFLIP network with 4 stages has latency of 13.8 while GRP has 22.7 clearly depicted in paper [13]. Two stage implementation of OMFLIP is the fastest solution and its implementation for 64 bit permutation requires only 48 bytes of memory [10].

In this paper, we have implemented a bit level permutation in hardware and evaluated its superiority by comparing it with an existing encryption method for a set of parameters and also for their resilience under an external and internal threat. OMFLIP and GRP design is best fit for providing rich security solution as efficient permutation box for small scale embedded system. It guards the constraints like less power consumption, less footprint area, faster throughput which are core of any embedded system design.

Implementation of bit permutation instructions will always results in compact hardware design. The RAM and ROM requirement for GRP would be very less as compared to other bit permutation instructions. Number of processing elements required for bit permutation instruction is also less. Lightweight cryptography is the field where use of bit permutation instructions will result in more compact hardware design. Only disadvantage about GRP is its latency. Time required for execution of GRP would be slightly more compared to other instructions. Future work will include implementation of bit permutation instructions in designing lightweight block ciphers. GRP lacks substitution property. Designing 'S' box for GRP may make the design more compact and more robust. Implementation of bit permutation instruction for embedded system security will emerge as lone robust solution in an eternal world of low power security design.

Acknowledgement

The authors would like to thank Symbiosis Institute of Technology, Pune, Symbiosis International University, Pune for providing resources to carry out this research successfully.

References:

[1] Marko Wolf, Andre Weimerskirch, Christof Paar, "*Secure –In Vehicle Communication*", Embedded Security in Cars, Springer, 2006.

[2] Tobias Hoppe, Stefan Kiltz, Jana Dittmana, "*Security Threats to automotive CAN network*", SAFECOMP 2008, LNCS5219, Springer, 2008.

[3] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, and Tadayoshi Kohno, "*Experimental security Analysis of a Modern automobile*", IEEE symposium on security and privacy, Oakland, CA, May 2010.

[4] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, "*Comprehensive Experimental Analysis of Automotive Attack Surfaces*" , USENIX Security, August 10–12, 2011.

[5] Hendrik Schweppe, Yves Roudier, "*Security issues in Vehicular System*", SAR-SSI 2010, 5TH Conference on Network architectures and information system security, EVITA project, 2010.

[6] Alireza Hodjat, Ingrid Verbauewhede, "*The Energy cost of secrets in adhoc Network*", Proceedings of the IEEE Circuits and Systems Workshop on Wireless Communications and Networking, 2002.

[7] Hwang P. Schaumont, K. Tiri, and I. Verbauwhede, "*Securing Embedded System*", IEEE Security & Privacy, 4(2):40–49, 2006.

[8] Zhijie Shi and Ruby B. Lee, "*Sub word sorting with Versatile Permutation Instructions*", Proceedings of the 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'02).

[9] Ruby B. Lee, Z. J. Shi and Y. L. Yin, Ronald L. Rivest, M.J.B. Robshaw "*On Permutation Operations in Cipher Design*" Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference 5-7 April 2004.

[10] Zhijie Jerry Shi, "*Bit Permutation Instructions: Architecture, Implementation and Cryptographic Properties*", Princeton, June 2004.

[11] Miller Alexander, Prof. Dr.-Ing Gunar Schorcht, *Embedded systems security: Performance Investigation of various cryptographic techniques in embedded systems*, http://www.kaspersky.com/images/miller,_alexander_embedded_systems_security_performance_investigation_of_various_cryptographic_techniques_in_embedded_systems-10-98478.pdf

[12] Gaurav Bansod**,** Aman G, Arunika G, Gajraj B, Chitrangdha S, Harshita A, "*Experimental analysis and implementation of bit level permutation instructions for embedded security*". *WSEAS Transactions on Information Science and Applications*, 10(9): 303-312 (Scopus; ISSN: 1790-0832), 2013.

[13] Z.J.Shi and R.B.Lee, "*Implementation Complexity of Bit Permutation Instructions*", in Proc. Asilomar Conf.Signals Stst. Computer, pp 879-886, 2003.

[14] Yedidya Hilewitz, Zhijie Jerry Shi and Ruby B. Lee, "*Comparing Fast Implementations of Bit Permutation Instructions*" Signals, Systems and Computers, *2004.* Conference Record of the Thirty-Eighth Asilomar Conference 7-10 NOV 2004.

[15] Zhijee Shi, Ruby B. Lee, "*Bit Permutation Instructions for accelerating Software cryptography*", Application-Specific Systems, Architectures, and Processors, 2000. Proceedings. IEEE International Conference, JULY 2000.

[16] Giorgos Dimitrakopoulos, Christos Mavrokefalidis, Kostas Galanopoulos and Dimitris Niolos, "*Sorter based permutation units for Media-Enhanced Processors*" IEEE Transactions on VLSI systems, vol 15, no. 6, pp 711-715, June 2007.

[17] Jer Min Jou, Yun Lung Lee, Chen Yen Lin and Chien Ming Sun, "*A Novel Reconfigurable computation unit for DSP applications*", IEEE comp. society annual symp. On VLSI, ISVLSI'07, pp 439- 444, 9-11 March 2007.

[18] Navid Lashkarian, Ed Hemphi, Helen Tarn, Hemang Parekh and Chris Dick, "*Reconfigurable Digital Front End Hardware for wireless base-station transmitters: Analysis, Design and FPGA implementation*", IEEE transactions on circuits and systems, vol 54, No. 8, pp 1666-1677, Aug 2007.

[19] Souvik Kolay, Sagar Khurana, Anupam Sadhukhan, Chester Rebeiro and Debdeep Mukhopadhyay, "*PERMS: A Bit Permutation Instruction For Accelerating Software Cryptography"*, 16th Euromicro Conference on Digital System Design, Sept 04-06, 2013.

[20] Philipp Grabher, Johann Grobschadl, and Dan Page, "*Light-Weight Instruction Set Extensions for Bit-Sliced Cryptography*", Lecture Notes in Computer Science - CHES2008, Volume 5154, 2008, pp 331-345.

[21] Zhimin chen, Ambuj sinha,Patrick schaumont, "*Using virtual secure circuit to protect embedded software from side channel attacks"*IEEE Transactions on Computers, VOL 62, No 1, JAN 13.