Chapter 2

GRP: A Bit Permutation Instruction

In this chapter, the results of a design of a compact cipher with adequate security for applications like pervasive computing have been presented. Our research work focuses on a compact hardware implementation of the cipher. Bit permutation instructions are efficient in such kind of implementations and they are complex in nature which gives them an edge in the cryptographic environment. Bit permutations are popularly known to be used in the permutation block known as diffusion property. The properties and security aspects of bit permutation instructions like GRP [26][27] and OMFLIP [64][65] were carefully studied. Among all bit permutation instructions GRP proved to be an efficient instruction in terms of cryptographic properties, memory size and total number of gate counts. Bit permutation instructions are widely studied and currently supported by all word oriented processors. GRP is an extensively researched instruction set, its cryptanalysis is well known, and many attacks have been tried in the past on bit permutation instructions. GRP is known for fast bit permutation [26][27]. Alternative to bit permutation is a simple look up table, but permutation operations are far better than a look up table. A Look up table typically requires nearly 23 numbers of instructions for a 64 bit permutation while GRP does the permutation in only six lines of instructions as mentioned in Table 2.1. Lookup tables have a large footprint and thus increase GEs. GRP is complex in nature that makes it more suitable for the cryptographic environment as compared to operations like shifting, multiply or addition. GRP is suitable specifically for encryption in an application like remote sensors continuously encrypting data and sending it to a server location [26][27]. Moreover, GRP has good differential properties because the paths of data bits totally depends on control bits applied to the structure. Change of even a single control bit will cause all the data bits to change at the output [26][27]. This property helps to achieve the desired avalanche effect and makes the design more robust against attacks. Algorithms like DES [28][29], SERPENT [66] and TWO FISH [67] use bit permutation instructions in their operations which help to resist against linear and differential cryptanalysis. Bit permutation instructions lack the confusion property that is an S-box. According to Shannon having the diffusion property alone is not sufficient to provide a secure cipher [68]. GRP uses subword permutation that not only does permutation efficiently but also accelerates the software cryptography [26][27].
The hybrid structure has been implemented with the help of GRP instructions along with the S-box of PRESENT cipher. As described earlier, GRP [26][27] is one of the most complicated bit permutation instructions that make it an obvious choice to be used in the cryptographic environment. GRP performs n bit permutation with \( \log_2(n) \) steps while other instructions take \( O(n) \) steps [26][27]. Research in this field and papers [25] [26][27] have shown the increased strength of cipher RC5 by introducing GRP instructions. GRP scales very efficiently to 2n bits on n bit systems by using instruction Shift right pair instruction (SHRP) in PA-RISC and in IA-64 processors [69][70]. Table look up is an alternative to bit permutation instructions, but it is much slower as it takes 16 cycles on a superscalar processor for the scheduling of permutation instructions, while GRP does it in only 8 cycles. By loading control bits, GRP requires 13 numbers of instructions while a table lookup needs 31 numbers of instructions. Table 2.1 shows the comparison of GRP with Table lookup on various parameters that shows the edge GRP has over a table look up, in terms of memory space and execution, which are the most important aspects in lightweight cryptography.

Table 2.1 Comparison of GRP with Table Lookup

<table>
<thead>
<tr>
<th>Table LOOK UP</th>
<th>GRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum no. of instructions for 64 bit word size permutation</td>
<td>23</td>
</tr>
<tr>
<td>Maximum no. of instructions for 128 bit word size Permutation</td>
<td>47</td>
</tr>
<tr>
<td>No. of cycles for scheduling of permutation instructions on Super Scalar processor</td>
<td>16</td>
</tr>
<tr>
<td>Maximum number of instructions for permuting 64 bit with control bits</td>
<td>31</td>
</tr>
<tr>
<td>Memory requirement for 64 bit permutation</td>
<td>16 kb</td>
</tr>
</tbody>
</table>

Paper [26][27] shows key generation by GRP achieving tremendous speed because most of the permutation instructions exist in this block.

2.1 GRP Implementation

Papers [25][26][27] show various implementations of GRP at hardware and at software levels. It was possible to achieve implementation of 128-bit encryption with GRP. The algorithm is explained in paper [26][27]. GRP 128-bit permutation is designed by us, and
implemented on 32-bit processor LPC2129, where it is necessary to give a 128-bit input, and GRP performs the exact same operations for 128-bits which are performed for an 8-bit encryption in Fig. 2.1. It is a universal design which generates codewords for n integers. Fig. 2.1 indicates the basic GRP encryption operations in terms of AND, OR and NOT gates. In Fig. 2.1, the key is generated by the key register according to SPECK cipher key scheduling [52][54][55], which is based on the user defined integer sequence, and that key is applied as a codeword to each of the permutations to do the encryption. 128-bit encryption has been designed and performed based on bit permutation instructions. The algorithm and the steps involved while designing the permutation box by using GRP are outlined below:

In a scenario, let us assume that the input is a plain text with a bit length \( w = 128 \) that needs to be permuted with the help of GRP. To permute 128-bits, the operation needs total 7 stages as \( 2^7 = 128 \) bits. 7 stages mean that GRP 128 will perform up to 7 rounds as \( 2^7 = 128 \). Similarly, for 64-bit and 8-bit permutations, we need a total of 6 and 3 stages, respectively. Fig. 2.1 indicates 8-bit permutation. Figure 2.2 shows cipher design based on GRP instructions.
2.2 PRESENT Cipher

The PRESENT cipher was proposed at ‘CHES 2007’, and the cipher design became popular among cryptographic designers due to its ease of use, remarkable hardware performance and robust security [8][9][71]. The PRESENT cipher uses bit permutation to provide the diffusion property. The PRESENT cipher was adopted as ISO/IEC lightweight cipher in 2012 [71]. The PRESENT cipher is a Substitution-Permutation network (SP-network) and consists of 31 rounds [8][9]. The PRESENT cipher supports a block length of 64-bit and key length of 128-bits. The block diagram of the PRESENT cipher is represented in Fig. 2.3.

Operations addRoundKey, sBoxlayer and pLayer are performed in the PRESENT cipher. addRoundKey performs bit wise XOR operation between the key and the current state output. sBoxlayer is the nonlinear substitution layer. The non-linear layer uses a single 4-bit S-box, S, which is applied 16 times in parallel in each round. The nonlinear layer is followed by the
permutation layer which is linear bitwise permutation. Hexadecimal values of the PRESENT cipher S-box are mentioned in Table 2.2.

<table>
<thead>
<tr>
<th>b</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>S[b]</td>
<td>C</td>
<td>5</td>
<td>6</td>
<td>B</td>
<td>9</td>
<td>0</td>
<td>A</td>
<td>D</td>
<td>3</td>
<td>E</td>
<td>F</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

### 2.3 New Hybrid Cryptostructure

After a careful study of the properties of bit permutation instructions like GRP [26][27], OMFLIP [64][65], CROSS [25], it was found that GRP has an edge over all other bit permutation instructions and is useful in accelerating cryptographic implementations. GRP does the fast bit permutation with less memory space and its complex structure makes it an attractive choice in cryptographic algorithms over instructions like multiply and rotate [26][27]. The disadvantage of bit permutation instruction is that it lacks an S-box, which is the most essential ingredient in designing secure block ciphers. This element directed our attention towards lightweight cryptographic algorithms and we carried out many experiments and studied the properties of algorithms like PRESENT [8][9], CLEFIA [10][11], DES [28][29], AES [2][3], SEA [38][39], TEA [18][19] and KANTAN [42][43].

As stated, the aim our design was to develop a compact cipher with adequate security for applications like pervasive computing. In this chapter, the results of our research work which focuses on the compact hardware implementation of a cipher are presented. Bit permutation instructions are efficient in such type of implementations. Bit permutation instructions are complex in nature, and that gives them an edge in the cryptographic environment. A well known cipher like DES is also implemented with the help of bit permutation instructions but falls prey to attacks because of short key lengths. Among all bit permutation instructions, GRP proved to be an efficient instruction in terms of cryptographic properties, memory size and total number of gate counts. OMFLIP has poor differential properties and its structure is easily susceptible to attacks. Bit permutation instructions are widely studied and currently supported by all word oriented processors. Recently, NSA has designed the lightweight ciphers named SIMON and SPECK which are optimized for hardware and software implementations [32]. NSA has been known for interesting key scheduling. Specifically the SPECK cipher has been optimized for efficient performance on microcontrollers.

A new cipher has been designed that was inspired by the PRESENT cipher design which is based on SP-network and has 6 rounds. A single round of hybrid structure has a single
addRoundKey operation and six rounds of the substitution layer and the permutation layer. The block diagram of the structure is represented in Fig. 2.4. The hybrid structure uses a total of 43 keys from $K_0$ to $K_{42}$, each of 64-bits, where $K_{42}$ is the post whitening key. These keys are generated using the SPECK key scheduling algorithm. Detailed operations are mentioned below,

![Fig. 2.4 Hybrid structure block diagram](image)

**addRoundKey**

addRoundKey does the following operation where $j$ varies from 0 to 63 and $d$ is the current state output,

$$d_j \rightarrow d_j \oplus K_j^i$$

**S-layer**

S-layer provides the confusion property; our proposed hybrid structure uses the PRESENT cipher S-box, which is a $4 \times 4$ S-box $S: F_{\mathbb{F}_2}^4 \rightarrow F_{\mathbb{F}_2}^4$. Hexadecimal values of the S-box are mentioned in Table 2.3.
Table 2.3 S-box

<table>
<thead>
<tr>
<th>b</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>S[b]</td>
<td>C</td>
<td>5</td>
<td>6</td>
<td>B</td>
<td>9</td>
<td>0</td>
<td>A</td>
<td>D</td>
<td>3</td>
<td>E</td>
<td>F</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**P-layer**

Bitwise permutation is used to provide the diffusion property. The GRP bit permutation instruction has been used. Swapping of bits depends on key bits for which GRP uses its own key generation algorithm. The GRP key scheduling algorithm involves a larger number of steps to generate a key due to which the requirement of memory is larger. So, we have used the SPECK key scheduling for generation of keys that reduces the overall memory size requirement.

We have mentioned a universal algorithm that will perform permutation operations on n number of bit inputs; in our design input is of 64-bit. The algorithm for encrypting 64-bits by using above mentioned variables is mentioned below:

```plaintext
for (i=0; i<n; i++)
{
    for (j=0; j<P; j++)
    {
        for (k=0; k<C; k++)
        {
            temp = x [(2*P*k) + j];
            x [(2*P*k) + j] = x [(2*P*k) + j + P];
            x [(2*P*k) + j + P] = temp;
        }
    }
    P/=2;
    C*=2;
}
```
Where $n = 6$, $P = 64$ and $C = 1$. Fig. 2.5 represents steps used in our hybrid cipher design.

2.4 Key scheduling algorithm

The proposed hybrid structure uses the SPECK key scheduling algorithm that will generate a total of 43 keys, each of 64-bits, from 128-bits user defined keys [32]. User defined keybits are stored into two key registers $K[0]$ and $K[1]$. $K[1]$ consisting of MSB 64-bits while $K[0]$ consists of LSB 64-bits. The key scheduling algorithm of SPECK is mentioned below:

$L[0] = key[1];$

for $(i=0; i<42; i++)$

{

$L[i+1] = (key[i] + RCS(L[i], 8)) \times I;$

$key[i+1] = LCS(key[i], 3) \times L[i+1];$

}

RCS performs right circular shift operation by $n$-bits and LCS performs left circular shift operation by $n$-bits.

2.5 Linear and Differential attack

One of the most important aspects in designing a block cipher is to do cryptanalysis. Cryptanalysis helps us to know whether the plain text can be derived from the cipher text or if some of the bits in the key can be identified. The most popular attack is brute force attack which decrypts the text by using all possible keys. Risk of this attack can be reduced by
increasing the key size, as then the possible combinations will be more in number and therefore difficult to compute. But, these attacks need a lot of time to compute robustness. Two properties or attacks which are of utmost importance are differential cryptanalysis and linear cryptanalysis.

Linear Cryptanalysis [20] is one of the most significant attacks which is applicable to symmetric-key block ciphers and the cipher needs to resist such kind of attacks. This attack is also referred to as the known plaintext attack. It uses the high probability occurrences of linear expression containing plaintext bits, cipher text bits and sub key bits. This expression is used for mounting a linear attack on a cipher. If \( P_L \) is linear probability then its bias can be given as \( |P_L - 1/2| \), bias \((\varepsilon)\) for S-box is \( 2^{-2} \). Matsui’s Piling-up lemma [20] is used to calculate the probability bias for ‘n’ rounds. An S-box that has non-zero input or non-zero output is referred to as an active S-box.

Differential Cryptanalysis [20] is another most significant attack applicable to symmetric key block ciphers. Biham and Shamir first applied a differential attack on DES in 1990. To mount the differential attack for a specific number of rounds in an encryption system, pairs of high probability input and output occurrences are used to recover the round keys. The S-box is a nonlinear component in our design and it is examined by forming the Difference Distribution Table (DDT).

The best ways to resist against linear attacks and differential attacks are

1] Optimizing the bias in LAT. For an ideal S-box bias, values should be 1/8 which is practically not possible to achieve.

2] By minimizing the differential probability. For an ideal S-box this probability is 1/16.

3] Increase the number of active S-boxes in cipher design.

As the permutation layer is based on key bits, the following three cases have been considered for calculation of the minimum number of active S-boxes:

1] All key bits are ‘0’,

2] All key bits are ‘1’,

3] Alternate key bits are ‘0’ and ‘1’
Case 1: All key bits are ‘0’

Fig. 2.6 represents the calculation of the minimum number of active S-boxes for a hybrid structure when all key bits are ‘0’. We found that for a single round of hybrid structure there are minimum 6 active S-boxes. So, for a total of six rounds, there will be minimum 36 active S-boxes.

Similarly for the other two cases there are minimum 6 active S-boxes for a single round of hybrid structure.
2.6 Complexity calculation

**Theorem 1:**
The proposed hybrid structure has a total of 6 rounds and for one round of the hybrid structure there are minimum 6 active S-boxes. For 36 active S-boxes, total differential probability (Pd) is $2^{-72}$. The required number of chosen plaintext / ciphertext will be $2^{72}$.

**Proof:**
For one round there are minimum 6 active S-boxes, so for 6 rounds there will be minimum 36 active S-boxes, the total differential probability is given as $(2^2)^{36} = 2^{-72}$.

By calculating the required number of chosen plaintext / ciphertext we can find the complexity of the differential attack and can be given as

$$N_d = C/P_d.$$  

Where $C = 1$ and $P_d = 2^{-72}$, so the number of chosen plaintexts required are

$$N_d = 1/2^{-72} = 2^{72}.$$  

As the required number of chosen plaintexts is greater than the available number of plaintexts i.e. $2^{64}$

$$2^{72} > 2^{64}$$

As the chosen plaintexts are more than the available number of plaintexts it is not possible to recover sub keys for complete rounds of hybrid cipher.

**Theorem 2:**
For 6 rounds of hybrid structure, complexity for known plaintext attack is $2^{74}$ and maximum bias is $2^{-37}$.

**Proof:**
For a single round, there are minimum 6 active S-boxes. By using Matsui’s [20] Piling-up lemma, we will get maximal bias for a single round linear trail and is given as,

$$2^5 \times (2^2)^6 = 2^7$$

Applying the same lemma again, we will get the maximal bias of 6 round linear trail:

$$2^5 \times (2^7)^6 = 2^{-37}$$
The required number of known plaintexts can be given as

\[ N_L = (1/P_L)^2 \]

For 27 rounds, the number of known plaintexts can be given as

\[ N_L = (1/2^{-37})^2 = 2^{74} \]

As the required number of chosen plaintexts is greater than the available number of plaintexts i.e. \(2^{64}\)

\[ 2^{74} > 2^{64} \]

Subkeys are not possible to recover because the required number of plaintexts is of order \(2^{74}\) which is more than the available plaintexts.

### 2.7 New Hybrid Cryptostructure, Implementation and Comparison

The compactness and mapping interface of GRP [26][27] with PRESENT [8][9] and CLEFIA [10][11] was investigated. In this chapter, the results of implementing most of the standard algorithms are presented in order to identify their memory requirements, gate equivalents and power consumption. All the standard algorithms are implemented and compared on the same platform which is LPC2129, a 32 bit processor by NXP (Philips).

We have implemented AES 128-bit, GRP for 128-bit and 64-bit, PRESENT for 64-bit, CLEFIA for 128-bit and DES for 128-bit block sizes with different key combinations of 64, 128 and 80-bits. Fig. 2.7 indicates the memory requirement of P-box of AES [2][3] and PRESENT [8][9] with GRP [26][27] and OMFLIP [64][65] computed based on the KEIL 4.0 simulator and LPC2129.
Fig. 2.8 shows the merger of the S-box of PRESENT with OMFLIP, AES and GRP. Results clearly show that PRESENT-GRP [73] has a much lower memory requirement as compared to other hybrid systems. LED [48][49] is the latest cipher which is a combination of PRESENT-AES which is also a SP-network. It has 64-bit block size and 128-bit key length. LED is faster than PRESENT in software but it is slower in hardware. Moreover, LED consumes high energy per bit in embedded applications which dissipates and consumes more power.

Fig. 2.9 shows a graphical representation of the memory size requirements of standard algorithms on a 32 bit processor. The hybrid module structure, PRESENT-GRP, needs lesser memory as compared to other standard algorithms [73]. In this hybrid module, we have designed the permutation box (P-Box) by using GRP for 128 and 64-bit block sizes. This
result shows the compactness of a hybrid cryptosystem which is the combination of PRESENT-GRP. 2980 bytes of flash memory are needed for PRESENT-GRP while PRESENT alone, which is lightweight compared to other structures, needs 3200 bytes of memory [73]. CLEFIA also has higher memory requirements which consume nearly 4708 bytes. PRESENT-GRP in Fig. 2.10 indicates 128-bit block size and 128-bit key generation while PRESENT-GRP 64 indicates 64-bit block size and 128-bit key. The KLEIN [44][45] algorithm is also implemented on a 32-bit processor and its comparison is also shown in Fig. 2.10.

2.7.1 PRESENT-GRP: A New Hybrid Lightweight Design

In this chapter, a hybrid cryptosystem design is discussed, which combines the S-box of PRESENT and the P-box of GRP, to produce a compact secure structure with adequate security. This fusion structure has both the properties of PRESENT-GRP and results in a tinier version than the original algorithm PRESENT [73]. Cryptanalysis for both the structures is studied and analysed properly and it shows good resistance to linear and differential attacks. The same PRESENT-GRP design is compared with various standard algorithms like PRESENT [8][9], AES [2][3], DES [28][29] and CLEFIA [10][11]. All these algorithms were implemented in embedded C on the same platform. This was done so that we do not observe any kind of artifacts due to the platform issues, either in the results or in the comparisons. Results of all algorithms were tested and verified through the 8-bit UART.
module of ARM7. An ‘UART’ module was used in the design merely as a demonstrator in order to verify and to be able to see the encryption / decryption outputs. The baud rate for this application was set at 9600 bps.

In the PRESENT-GRP module [73], 64-bit / 128-bit blocks were passed through the S-box of PRESENT and after mapping according to PRESENT, the output was passed to the permutation layer which performed encryption based on GRP algorithms. Keys at each stage were applied based on key generation by the SPECK cipher [52][54][55].

We have designed an optimized version of GRP by doing small changes at the algorithmic level. The GRP design presented in previous papers [26][74][75] need 3224 bytes of FLASH memory. In this chapter, a compressed version of GRP is reported, a design that consumes only 3088 bytes of memory [27]. This optimized structure has been achieved by reducing many arrays to just two arrays. An entirely new logic has been designed which supports very less number of arrays and works very fast as compared to the existing structure. Execution time for our design is nearly half that of the old GRP design at the software level. Few observations made by us during code optimization are listed as follows:

- Using character data type instead of integer data type
- Reduced instructions i.e. making one instruction that can perform three operations, instead of using three different instructions for different operations
- Using local variables instead of global variables; global variables consume more space
- Using minimum number of functions
- Making binary to hex / hex to binary functions instead of using “Power” functions (defined)
- Using minimum number of variables as Recursive
- Making complex logic for long processes
- Passing only one data type into the function instead of multiple. For example, GRP (a)

All these steps help to achieve a more optimized structure of GRP which results in 1789 GEs for 64-bit permutation. Table 2.4 shows memory requirements for past implementations of GRP and optimized GRP design. To the best of our knowledge, this is the most optimized design for GRP 128-bit.
Table 2.4 Optimized GRP Design

<table>
<thead>
<tr>
<th>Memory Size of Old GRP 128</th>
<th>Memory Size of Optimized GRP 128</th>
</tr>
</thead>
<tbody>
<tr>
<td>3224 bytes</td>
<td>2944 bytes</td>
</tr>
</tbody>
</table>

The S-box of PRESENT is very compact in design and uses a 4x4 box in order to reduce gate complexity and power consumption. Larger S-boxes result in high GEs as is seen in the case of DES [28][29]. The higher bit S-box has more Boolean equations resulting in a high gate count. Moreover, implementation of the S-box of PRESENT has low hardware cost and less GEs.

![Fig. 2.10: New recent lightweight algorithms implemented and compared with PRESENT-GRP on LPC2129](image)

Fig. 2.10 shows the comparison of PRESENT-GRP with the latest lightweight ciphers. LED [48][49] and ZORRO [58][59] are recent lightweight ciphers. All ciphers in Fig. 2.10 are of SP-network type. ZORRO is similar to the AES algorithm. ZORRO has an 8-bit S-box while others have 4-bit S-boxes. It has 128-bit block size and 128-bit key. As ZORRO has an 8-bit S-box, it needs higher GEs compared to other ciphers using 4-bit S-boxes. ZORRO needs 24 rounds to secure its structure which results in higher memory requirement as shown in Fig. 2.10. Hummingbird [60][61] has 128-bit block size and 896-bit key length. It is a combination of block cipher and stream cipher. Hummingbird requires a long initialization process as compared to block ciphers like PRESENT, which introduces latency and higher execution time. Hummingbird has less encryption speed and its authentication mechanism is less efficient. The 4-bit to 4-bit S-box of PRESENT needs 21 to 28 GEs when it is
implemented with the UMCL18G212T3 library. Table 2.5 indicates the gate count of the standard cell of the UMCL18G212T3 library [76].

Table 2.5 Gate Count of UMCL18G212T3 Library

<table>
<thead>
<tr>
<th>Standard Cell</th>
<th>Process</th>
<th>Library</th>
<th>Cell Name</th>
<th>GE</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOT</td>
<td>0.18µm</td>
<td>UMCL18G212T3</td>
<td>HDINVBD1</td>
<td>0.67</td>
</tr>
<tr>
<td>AND</td>
<td>0.18µm</td>
<td>UMCL18G212T3</td>
<td>HDAND2D1</td>
<td>1.33</td>
</tr>
<tr>
<td>OR</td>
<td>0.18µm</td>
<td>UMCL18G212T3</td>
<td>HDOR2D1</td>
<td>1.33</td>
</tr>
<tr>
<td>MUX</td>
<td>0.18µm</td>
<td>UMCL18G212T3</td>
<td>HDMUX2D1</td>
<td>2.33</td>
</tr>
</tbody>
</table>

For the 64-bit operations in our design, 16 four bit S-boxes of PRESENT whose output is given to GRP for permutation have been used. By combining PRESENT-GRP, the total gate count needed is 2125 GEs by considering S-box gate count as 21 [8][9][71]. PRESENT’s S-box consumes nearly 21 GEs for a single 4-bit S-box. PRESENT-GRP gate counts results in 2125 which is less than other algorithms except for PRESENT which is implemented based on the ‘round based’ architecture and whose gate count is 1884 [8][9][71].

GEs for other algorithmic design like CLEFIA [10][11], SEA [38][39], TEA [18][19], HIGHT [34][35], mCRYPTON [36][37], ICEBERG [40][41], DESX and DESXL [32][33] based on the specific library functions, is computed and a comparison is made which is illustrated in Fig. 2.11.

Fig. 2.11: GEs comparison chart of all standard algorithms with PRESENT-GRP
In this chapter, CLEFIA-GRP, where S-box of CLEFIA is used with GRP as permutation layer, has also been designed and implemented. CLEFIA-GRP is compared with PRESENT-GRP to understand the compactness of this hybrid structure. Fig. 2.12 represents the hybrid implementations and their comparisons with standard lightweight designs like PRESENT and CLEFIA. In Fig. 2.12, PRESENT-GRP and PRESENT-GRP 64 are compared with other lightweight standard algorithms like SEA-GRP, PRINCEcore-GRP, KLEIN-GRP and CLEFIA-GRP [73]. The hybrid structure of PRESENT-GRP has very low memory requirement as compared to the other algorithms. The SEA-GRP algorithm results in lesser memory size as compared to PRESENT-GRP as it has a 3-bit S-Box while others have 4-bit S-Boxes. CLEFIA-GRP fusion results in 4536 bytes of flash memory requirement which depicts a heavy cryptographic design and may not be suitable for lightweight applications like RFID tags. The total number of GE’s available in pervasive design for security purpose would range from 200-2100 GEs. Based on the GE limit, our hybrid design makes a mark in that the GE range consumes less memory as compared to the entire standard lightweight and other cryptographic algorithms at the software level. Hybrid structure, if mapped properly, always has an edge over other structures as the structure carries the good and robust properties of the individual designs. Paper [26] suggests that the combination of GRP and DDR (Data Dependent Rotation) may show good resistance against differential and linear cryptanalysis which is later proved by combining RC5 and GRP [74][75].

Fig. 2.12: Hybrid Crypto structure comparison chart
Bit permutation instructions increases strength of block cipher by allowing them to perform any arbitrary permutations efficiently with ‘log(n)’ steps compared to ‘n’. It performs fast bit permutation and uses subword sorter that makes the operation faster and can increase throughput in applications like scanning an image, performing bubble sort and in permutation layer in block ciphers. GRP have all these good properties that provide strength in cryptographic environment. But, it lacks S-box which is necessary to provide secure design. This shifted our focus to find a lightweighted S-box that can be mapped to GRP to get secure and efficient hybrid crypto structure. In this section, we have presented the design with a permutation box (P-box) by using GRP for 128 and 64-bit block size. This paper proposes a novel approach by introducing a compact hybrid system in terms of memory requirements that is best suited for lightweight cryptographic design.