Implementation of new hybrid lightweight cryptosystem

C.G. Thorat
Department of Electronics and Telecommunication Engineering, COEP, Pune, India, and
V.S. Inamdar
Department of Computer and Information Technology, COEP, Pune, India

Received 9 December 2017
Revised 30 April 2018
Accepted 2 May 2018

Abstract

Embedded systems, Internet of Things (IoT) and mobile computing devices are used in various domains which include public-private infrastructure, industrial installation and critical environment. Generally, information handled by these devices is private and critical. Therefore, it must be appropriately secured from different attacks and hackers. Lightweight cryptography is an aspiring field which investigates the implementation of cryptographic primitives and algorithms for resource constrained devices. In this paper, a new compact hybrid lightweight encryption technique has been proposed. Proposed technique uses the fastest bit permutation instruction PERMS with S-box of PRESENT block cipher for non-linearity. An arbitrary n-bit permutation is performed using PERMS instruction in less than log (n) number of instructions. This new hybrid system has been analyzed for software performance on Advanced RISC Machine (ARM) and Intel processor whereas Cadens tool is used to analyze the hardware performance. The result of the proposed technique is improved by the factor of eight as compared to the PRESENT-GRP hybrid block cipher. Moreover, PERMS instruction bit permutation properties result a very good avalanche effect and compact implementation in the both hardware and software environment.

Keywords Lightweight cryptography, PRESENT, PERMS, GRP, Bit permutation

Paper type Original Article

1. Introduction

The increasing use of mobile computing devices in the field of Information and Communication Technology (ICT) has raised concerns about security. Lightweight cryptography has made more overdrive from various cipher proposals such as PRESENT [15], CLEFIA [16], KATAN [19], HEIGHT [18], SIMON/SPECK [20], Fantomas [21], KLEIN

The authors would like to thank the Signal and Image processing Center of Excellence, College of engineering, Pune and JSPM's Rajarshi Shahu College of Engineering, Pune for giving all kind of support. We also express thanks to unknown referees for their helpful comments.

Publishers note: The publisher wishes to inform readers that the article "Implementation of new hybrid lightweight cryptosystem" was originally published by the previous publisher of Applied Computing and Informatics and the pagination of this article has been subsequently changed. There has been no change to the content of the article. This change was necessary for the journal to transition from the previous publisher to the new one. The publisher sincerely apologises for any inconvenience caused. To access and cite this article, please use Thorat, C.G., Inamdar, V.S. (2020), "Implementation of new hybrid lightweight cryptosystem", Applied Computing and Informatics. Vol. 16 No. 1/2, pp. 195-206. The original publication date for this paper was 04/05/2018.

Applied Computing and Informatics
Vol. 16 No. 1/2, 2020
pp. 195-206
Emerald Publishing Limited
e-ISSN: 2210-8327
p-ISSN: 2634-1964
DOI 10.1016/j.aci.2018.05.001

[22] and many other ciphers. Lightweight cryptography aims to offer sufficient security level with an optimum use of resources [11–14]. The optimum use of resources includes area, battery, CPU, memory and power. Among these, power consumption is one of the crucial factors on which lightweight ciphers need to work. Power consumption is strongly dependent on Gate Equivalents (GEs) and CPU cycles. GE is focused by hardware implementation and CPU cycles are focused by software implementation of a lightweight cipher. However, security properties should not be compromised for GEs and CPU cycles. For radio-frequency identification (RFID) tags, GEs are normally 1000–10000 but only 300–2100 GEs allotted for security purpose [23]. In lightweight cryptography, researchers adopts different approaches to develop an lightweight cipher such as modifying existing cipher, optimizing existing cipher or developing an entire new cipher. The third approach was explored in 2007 [15] where an entirely new cipher was designed, and it is known as the PRESENT block cipher. PRESENT block cipher is remain inspiration for many lightweight cryptography researchers, who have added their efforts to make it more better, which is discussed in Section 3. There are many ciphers where entirely new ciphers have been designed such as TEA [27], LED, ZORRO [29], Hummingbird [30], KATAN and KTANTAN [19], Halka [31], TWINE [32], RECTANGLE [28], GOST [33], PRINT [34], PUFFIN [35], Fantomas [36], Midori [37], Twofish [39] and mCrypton [38]. Block ciphers that have used permutation are summarized in Table 1. below.

Based on the algorithm structure, block ciphers are classified into Substitution Permutation Network (SPN), Feistel network and stream and Lai- Massey.

We introduce a new hybrid lightweight block cipher which supports portable and secure software as well as hardware implementation of PRESENT block cipher. A permutation layer of PRESENT block cipher is modified and implemented with PERMS bit permutation instruction to improve the performance. A detailed study is carried out with the help of properties and security aspects for the bit permutation instructions like SWPERM [5], GRP [4], PERMS [1] and OMFLIP [7,9] in the next section. As compared to other bit permutation instructions, PERMS provides an efficient bit permutation instruction in terms of cryptographic properties, CPU cycles and total number of gate counts. PERMS instruction is complex in a nature that makes it more suitable for cryptographic environment. PERMS instruction is most appropriate for cryptographic functions such as encryption and hash techniques, especially applications where continuous encryption-decryption operations are required to be performed. Linear and differential cryptanalysis properties of PERMS instructions are elaborated in the Section 6.

The main aim of this paper is to present the results of a compact hybrid cipher with ample security for resource constrained devices. The proposed block cipher is implemented and tested on the ARM and Intel processors. The experimentations are carried out on ARM processor Cortex-M and more powerful Cortex-A series processors. The performance parameters of the proposed cipher are compared with other existing ciphers such as

Table 1.
Comparison of lightweight block ciphers used in bit permutation.

Name of the block cipher	Input block size	Key size	No. of rounds	Algorithm Design Pattern
HIGHT	64	128	32	GFN
Pickolo	64	80/128	25/31	GFN
PRESENT	64	80/128	31	SPN
DESLX	64	184	16	Feistel
Midori	64/128	128	16/20	SPN
mCrypton	64	64/96/128	12	SPN
AES	128	128/192/256	10/12/14	SPN
Clefia	128	128/192/256	18/22/26	GFN

PRESENT, CLEFIA and PRESENT-GRP [2]. We have obtained better results which are shown in Section 5.

2. Bit permutations

Let B is any arbitrary bit string of length n and $(B_{n-1}, B_{n-2} \cdots B_1, B_0)_2$ where,

$$B \epsilon \{0, 1\}$$

Let P is a sequence of the form $(P_{n-1}, P_{n-2}, \cdots P_2, P_1, P_0)$ comprising random permutation from 0 to $n - 1$.

The permutation of B with P is given by $(bp_{n-1}, bp_{n-2}, \cdots bp_2, bp_1, bp_0)_2$.

Many existing Instruction Set Architectures (ISA) provide limited support for performing such arbitrary bit permutations compared to the regular permutation. There are different alternative ways to perform bit permutation as discussed below:

2.1 Logical operations

In this operation, different logical operations are used to perform the final bit permutation. The use of each logical operation used to perform bit permutation is given as follows:

1. AND operation: To extract the required bits from n bits which have to be selected by a mask.
2. Shift operation: To shift bits to their new position.
3. OR operation: To combine with previously permuted bits.

This technique requires many operations that lead to the increase in the number of instructions and memory required [3].

2.2 Lookups Table

In this method, the input bit stream is partitioned into multiple sections. Then, the bits in each section are permuted simultaneously with the help of lookup table. Finally, the result of each section is combined to produce the final results of permutations. The required instruction count solely depends on the number of sections formed. Less number of sections needs a few instructions but it increases memory requirement.

Apart from these two basic methods, bit permutation can be accelerated with the help of certain instructions like BFLY-IBFLY [6], PPERM-PPERM3R, CROSS, GRP, OMFLIP and SWPERM-SIEVE. All these instructions are compared in Table 2 against various parameters such as the number of instructions required, memory requirement, CPU cycles, Time complexity and mapping. Among these instructions, GRP instruction is used [2] to build a hybrid lightweight encryption. But after detailed study, we found another better instruction that is PERMS instruction. GRP performs 128-bit arbitrary permutation using 64-bit instructions set in 16 instructions. Among these 16 instructions, two are Shift Right Pair (SHRP) instructions which are available only in IA-64 and PA-RISC processors. However, other processors do not have SHRP instruction in their instruction set, thus on other processors GRP needs 4 instructions to perform the same operation. Therefore, GRP utilize total 22 instructions on other processors whereas PERMS instruction requires only 18 instructions to perform 128-bit permutation using 64 bit instructions on any processor. Permutation with repetitions is referred to as 'Mappings'. In this a bit in the input bits can be replicated and can appear at multiple locations in the output. Currently, none of the GRP, CROSS and OMFLIP permutation instructions supports mappings. Only PPERM and PERMS instructions support Mappings. Therefore, it motivates to use PERMS for lightweight cryptography. Table 2 shows a detailed comparison of different bit permutation instructions.

	PPERM3R PPERM	OMFLIP/ CROSS	GRP	PERMS	SWPERM SIEVE
Maximum number of instructions for 64 bit word size permutation	8	6	6	4	11
Mappings	Efficient	N/A	N/A	Efficient	N/A
Maximum number of instructions for 128 bit permutation using 64 bit Instruction	34	24	16*/22**	18	39
Speed	Medium	Fast	Fast	Fast	Medium
Scalability to 2n bits	Inefficient	Less efficient	Efficient	Efficient	Inefficient
Time Complexity for 64 bit word size permutation	O(log (n))	log(n)	log(n)	< log(n)	O(log (n))
Minimum operands required	4	5	3	3	3
Memory requirement (bytes)	64	48	48	32	48
Transistor Count [4]	7 k	3 k	68 k	2.7 k	7.4 k

* On PA-RISC [10] and IA-64 [8] processor only.
** Other Processors.

Table 2.
Comparison of bit permutation instructions.

3. Lightweight cryptography

Many devices have started to become pervasive computing devices which have in built embedded computing power. There is a huge adoption of IoT devices which leads the importance of lightweight cryptography. Lightweight cryptography has become an active research area for researchers since last two decades. For hardware oriented lightweight cryptography techniques performance parameters are area, GEs and power consumption. However, software based cryptography techniques tackles the memory usage, CPU usage and energy constraints. Standard algorithms like AES [24], DES [25], T-DES [26] and SHA-1 have proven their security very well and thus they are used extensively in many standalone pervasive computing as well as mobile devices. However, standard algorithms consumes system resources such as memory or CPU cycles at a very high level which makes them unsuitable for resource constrained devices. Hence, it leads to thrust for a lightweight cipher. In the last two decades, many researchers have come up with their lightweight ciphers, against which different attacks have been proven [11]. PRESENT and CLEFIA are two algorithms which are accepted as International Organization for Standardization (ISO) lightweight cryptography standards ISO/IEC (29192-2P:2012). CLEFIA was developed by Sony in 2007 and it targeted to be used in Digital right managements. PRESENT is developed by the Orange Labs (France), Ruhr University Bochum (Germany) and the Technical University of Denmark in 2007. It is best known for its compact size. PRESENT cipher is used as a benchmark by many researchers.

Various modifications have been carried out to improve hardware as well as software implementation results of a PRESENT cipher. Author Poshmann has implemented PRESENT cipher on different processors ranging from 4-bit to 64-bit [44]. PRESENT-GRP [2] is a hybrid design that has replaced the permutation layer of the original PRESENT cipher by GRP. PRESENT cipher and other twelve block ciphers are implemented and optimized on three different platforms such as 8-bit ATmega, 16-bit MSP430 and 32-bit ARMCortex-M3 [13]. Benadjila et al. provide PRESENT and many other block ciphers implementation to speed up through table based, vector instruction and bit sliced implementations on Intel x86 architectures [45]. It has been concluded that, the bit sliced implementations might not be useful when the amount of data to be enciphered at a time is small. The compatibility between the server and the client is one of the issue arises in bit sliced implementation.

There is another work presented by Tiago et al. [46], where bit sliced implementation and masking technique is used to prevent side channel attacks. In bit sliced implementation, constant time implementation can be achieved that helps to protect against timing attacks. They have modified PRESENT cipher in two ways; first permutation P is decomposed in P_0 and P_1 in alternative rounds and second S-box is implemented through bit-sliced implementation. However, permutation P is applied to some of the round keys and they have restricted their implementation to key size of 80 only.

The prime objective of this research work is to improve the PRESENT cipher hardware based performance. Although our aim is to improve hardware implementation by reducing the GE, we have also achieved satisfactory results for software based implementation. The proposed cipher PRESENT-PERMS is not having any specific processor restriction like GRP instruction or bit sliced implementation. Bit sliced or vector instruction based implementation is supported by only certain higher end processors.

As mentioned in Table 1 many lightweight ciphers uses permutation as their linear layer. Among these PRESENT and CLEFIA are selected to discuss and implement because they are the ISO/IEC standards and both algorithms have deeply been tested against various attacks earlier. PRESENT and CLEFIA lightweight ciphers have proved strong resistance against linear and differential cryptanalysis [15–17]. PRESENT cipher can be optimized with PERMS permutation instruction in software environment. The detailed implementation and analysis of proposed cipher is presented in the next section.

4. Proposed hybrid cipher implementation

The proposed lightweight hybrid cipher (PRESENT-PERMS) block diagram is as shown in Figure 1. A detailed study of bit permutation instructions is carried out and it has been found that PERMS instruction has a greater impact on other instructions. PERMS instruction is superior to GRP instruction in terms of CPU cycles. Different ciphers such as PRESENT, CLEFIA, PRESENT-GRP and proposed hybrid (PRESENT-PERMS) cipher are implemented and tested on 32-bit ARM processor in 'C' language. S-box of PRESENT cipher has a good number of active S-boxes and maximal bias for the linear approximation which makes it resistant against linear and differential cryptanalysis.

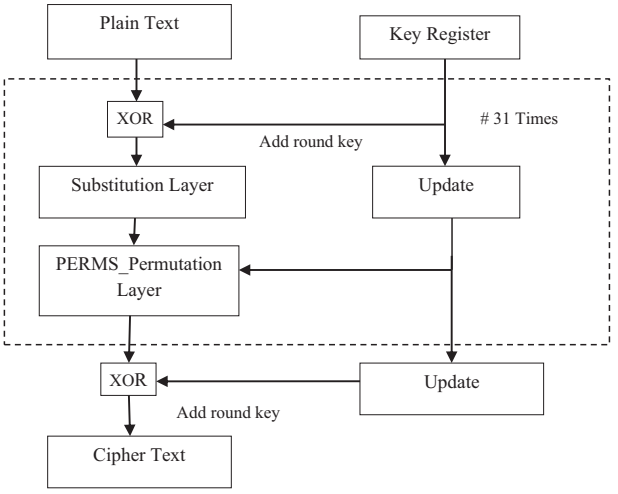

Figure 1.
Proposed block diagram of hybrid (PRESENT-PERMS) cipher.

4.1 Implementation of PERMS instruction

In PERMS instruction, two different algorithms are used to calculate control bits and to perform arbitrary permutation. 'C' language code to calculate control bits is presented below:

4.1.1 'C' language code to generate control bits.

```
for (i = n; i > 0; i−−)
{
  // Find the index of an array input2 [] in array input1 []
    for (m = 0;m < 7;m++)
    {
      if (input1[m] == input2[i]){
      k = m;
      break;
    }
  }
    // Swap input1 [i] with input1 [k].
      int j = input1[i];
      input1[i] = input1[k];
      input1[k] = j;
      control_bits[ind++]=k;
```

Array input1[ ] and input2[ ] are used as an input array and array control_bits [ ] is used to hold control bits. Array input1 [ ] holds monotonically increasing sequence of integers of a size *n* and array input2[ ] is a given permutation array.

4.1.2 'C' language code to perform arbitrary permutation. To perform arbitrary permutation, control bits array control_bits [ ] and sorted array pp[ ] are used as an inputs. Following given 'C' code generates final permutation array as an output:

```
for (i = 7,ind = 0;i > 0;i−−,ind++)
{
  j = pp[i];
  pp[i] = pp[control_bits[ind]];
  pp[control_bits[ind] ]= j;
}
```

It is clear from the above algorithm that PERMS instruction requires only four instructions whereas GRP requires six instructions. The CPU cycles count for both instruction implemented in 'C' language is given in Table 3.

Table 4 shows different parameters used for the implementation of the PRESENT-PERMS hybrid cipher for 64 and 128 bits. The different parameters such as block size, key size and number of rounds are considered for implementations.

Table 3. CPU cycle count for GRP and PERMS.

	GRP	PERMS
No. of CPU cycles to perform 64bit permutation with control bits	245	180

5. Experimentations and result analysis

5.1 Hardware based evaluation

All the cipher design considered for the benchmarking were implemented in Verilog code and the functional verification was carried out using Cadens CDS Encounter v11.10 – p003_1 (64 bit) simulation software. The designs were synthesized using the RTL Compiler for the Standard Cell library of the STM 90 nm Logic Process. Cadens calculates GE more accurately than other approaches used in previous work. The performance parameters used to compare the proposed hybrid cipher and other block ciphers are CPU cycles, GE and power consumption. The other ciphers such AES, PRESENT64, PRESENT128, PRESENR-GRP64, PRESENR-GRP128 and the proposed cipher PRESENT-PERMS are implemented on the same hardware and software platform to have a fair comparison. There are different PRESENT implementations optimized either for latency, area or throughput. Benchmarking results are obtained from the area optimized implementations. From Table 5, it can be noticed that PRESENT-PERMS require lowest GEs and energy per bit for encryption operation as compared to other ciphers. Many devices or applications more often use only encryption than encryption-decryption. For encryption-decryption operations PRESENT-PERMS GEs are less than CLEFIA, AES and PRESENT-GRP. Comparative results of the proposed hybrid cipher PRESENT-PERMS and other ciphers are shown in Table 6.

5.2 Software based evaluation

Permutation is a main building block for SPN based ciphers. Figure 2 shows a comparison of different permutations. It is observed that PERMS takes the least number of CPU cycles for 128-bit permutation. PERMS, OMFLIP and GRP are themselves acting as P-Boxes (Permutation-Boxes). The experimentations are carried with 128-bit input to compare permutation

Table 4. Parameters of PRESENT-PERMS 64 bit and 128 bit Implementations.

Cipher	Block size	Key size	Number of rounds
PRESENT-PERMS_64	64	128	31
PRESENT-PERMS_128	128	128	31

Table 5. Comparative results of the proposed hybrid cipher and other ciphers for the STM 90 nm.

Cipher Name	Block Size/Key Size	Operation	GE	Energy (pJ)	Energy/bit (pJ)
AES	128/128	Enc + Dec	24,234	816	6.4
		Enc	14,895	462.7	3.6
PRESENT	64/80	Enc + Dec	2286	274.2	4.3
		Enc	1518	169.3	2.6
CLEFIA	128/128	Enc + Dec	2838	595.7	4.7
		Enc	1928	352.3	2.8
PRESENT-GRP	64/128	Enc + Dec	2425	315.1	4.9
		Enc	1496	180.5	2.8
PRESENT-PERMS	64/80	Enc + Dec	2368	278.8	4.4
		Enc	1406	162.2	2.5

Table 6. Differential properties of PERMS.

Operation	Type A	Type B	Type C
PERMS	For any s, t p = 1/n	E(|Δ|) = n/4	E(|Δ|) = n/4

algorithms. Software implementation is carried out on ARM cortex M3, ARM cortex-A15 and Intel i5 processor. CPU cycles are measured on ARM processor by accessing performance monitor control register. For compilation GCC compiler is used with O3 optimization level.

CPU cycles needed by the encryption operation and encryption-decryption both the operations are calculated. The CPU cycles required for the PRESENT-PERMS and other ciphers are as shown in Figure 3. PRESENT-PERMS needs the lowest CPU cycles for both encryption as well as encryption-decryption operations. For all the cipher algorithms CPU cycles count, lowest CPU cycle count is considered for the final resultant parameter. From Figure 2 it can be noticed that PRESENT-PERMS requires least CPU cycles whereas CLEFIA needs the highest CPU cycles.

6. Security analysis

Cryptanalysis for any lightweight block cipher is a vital step to be performed. Cryptanalysis helps us to know the relationship among the plain text, key and cipher text. It also aims to find some details about key or cipher text or both. The most popular attacks on block ciphers are differential and linear cryptanalysis. They have been described respectively in [40] and [41] deeply. Since their inception, a significant research has been carried out to show their relationship and to better solutions to thwart

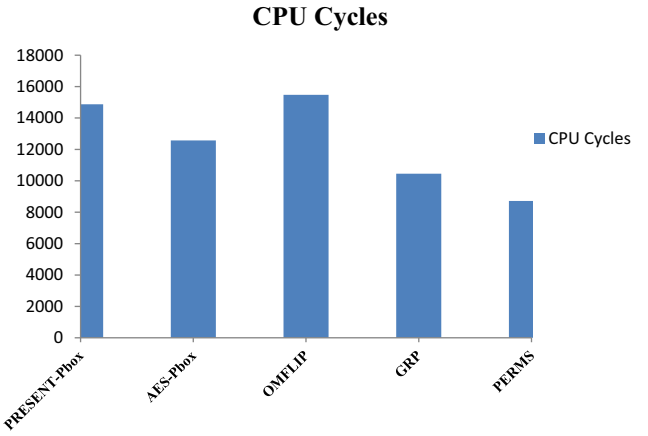

Figure 2.
P-box comparison.

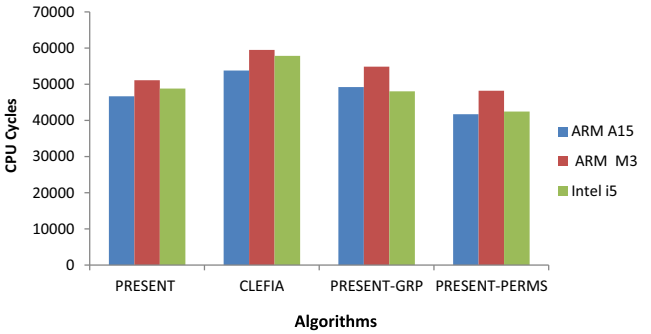

Figure 3.
CPU cycles comparison chart implemented on ARM Cortex A15, ARM Cortex M3 and Intel i5 processors.

them [42]. Matsui's branch and bound search algorithm [23] is one of the most powerful and classic methods for obtaining a security bound with respect to differential and linear attack.

In differential cryptanalysis two plain texts are selected (x_1 and x_2) with some difference as ΔD. Further, it is measured by XOR operation and these two plain texts are converted into cipher texts, where difference between these two cipher texts is denoted as ΔC. The pair $(\Delta D, \Delta C)$ is referred as differential characteristics. The ΔC is expected to be a larger value as compared to average probability. This section analyzes the bit permutation instruction PERMS's differential and linear cryptanalysis. Any permutation operation can described as R = P Opr Q, where,

Opr is bit permutation operation that is PERMS operation,

P is the bits to be permuted according to the Q,

R is the output of a bit permutation operation.

For bit permutation there are three forms of differential characteristics most useful [43]:

Type A: $(e_s, 0) \rightarrow e_t$

Type B: $(0, e_t) \rightarrow \Delta$

Type C: $(e_s, e_t) \rightarrow \Delta$

A differential characteristic of the bit permutation operation PERMS is described with a triplet $(\Delta p, \Delta Q) \rightarrow \Delta R$ with the probability p in which triplet holds true when the inputs are selected at arbitrary. In differential characteristics *es* specifies the n-bit word that has all bits zero excluding for a single one bit which is at position s. Type A specifies about how single bit at position s is shifted, when $Q_1 = Q_2$ are randomly selected. Probability p decides how likely bit at position s in P is shifted to bit t in R. For Type B and C, diffusion effect is compared by calculating the Hamming weight of ΔR. A bigger hamming weight results in an avalanche effect. Table 7 shows differential characteristic of PERMS.

$E(|\Delta|)$ denotes the expected value of a variation while input sequence is random. From the Table 7, it is cleared that PERMS achieves required avalanche effect.

A linear approximation of the permutation $R = P\ Opr\ Q$ *is* a triplet (£p, £q, £r)

where,

£ is a binary vector and length £ and P is equal.

Probability *Pr* holds on arbitrary inputs and described as:

$£_1 \oplus P\ £q.Q = £r.R$

The linear approximation bias is $|pr - 1/2|$. Linear approximations are two types which are used to compare PERMS instruction as follows:

Type D: (*es*, 0, *et*)

Type E: (*es*, *eu*, *et*)

Type D approximations do not include any bits in Q. Thus, it measures how uniformly the permutation moves the bits around. Type E approximations include some bits of Q. It measures how control bit *qu* determines the path of *ps* to *rt*. The biases of Type D and Type E characteristics for PERM instruction are mentioned in Table 7, in which b indicates bias.

In PERMS based bit permutation, input bit can move to any random position with equal probability $1/n$. The bias is $1/(2n)$ for all s and t. PRESENT cipher has achieved a very good

Operation	Type D	Type E
PERMS	$b \leq 1/(2n)$ Maximum with $s = t = 0$	$b \leq 1/(2n)$ Maximum with $s = u = t = 0$

Table 7.
Linear properties of PERM permutation.

security through its compact 4-bit S-boxes. The S-boxes used in PRESENT can be implemented with less GE and power consumption [15]. Novel properties of PRESENT S-boxes succeed to have an expected avalanche effect and enough number of active S-boxes. PRESENT cipher has strong linear and differential characteristics to stand against linear and differential cryptanalysis attack. PRESENT cipher is strongly defended against other attacks such as algebraic attack, structural attack and key schedule attacks [15].

7. Conclusions

PERMS bit permutation instruction performs arbitrary permutations in less than log (n) steps, compared to the all other bit permutation instructions. PERMS takes a less number of CPU cycles and GE which makes it faster and area efficient compared to GRP. Along with speed PERMS also provides a good security properties which makes it a good candidate for lightweight cryptography. However, for any block cipher there is a need of linear and non-linear layers. Therefore, PRESENT cipher has been selected for hybrid design that is an ISO standard and proven cipher. In this hybrid crypto cipher, PERMS is used for a permutation layer and PRESENT S-box is used as a non-linear layer. PERMS instruction not only has good differential and linear cryptanalysis properties but also it is capable to prevent brute-force attack. Furthermore software performance is evaluated on popular ARM and Intel processors. CPU cycles for PRESENT-PERMS implementation results in very less compared to other standard lightweight algorithms. To test proposed cipher for hardware performance, area and energy measurement is carried out on Cadens tool. The proposed cipher is tested for vector implementation on both ARM and Intel processors to speed up the S-box implementation which also provides timing attack protection. This hybrid block cipher proves to be very useful for lightweight cryptography community.

References

[1] S. Kolay, S. Khurana, A. Sadhukhan, C. Rebeiro, D. Mukhopadhvay, PERMS: A bit permutation instruction for accelerating software cryptography, 16th Euromicro Conference on Digital System Design, 2013, pp. 150–156.

[2] G. Bansod, N. Raval, N. Pisharoty, Implementation of a new lightweight encryption design for embedded security, IEEE Trans. Inf. Forensics Security 142–151 (10(1), 2014,) 142–151.

[3] R. Lee, Z. Shi, X. Yang, Efficient permutation instructions for fast software cryptography, Micro, IEEE 21 (6) (2001) 56–69.

[4] Z. Shi, R. B. Lee, Bit permutation instructions for accelerating software cryptography, in: ASAP, pp. 138–148, IEEE Computer Society, 2000, pp.138-148.

[5] John P. McGregor, Ruby B. Lee, Architectural techniques for accelerating subword permutations with repetitions, IEEE Trans. On Very Large Scale Integration (VLSI) System 11 (3) (2003) 80–91.

[6] X. Yang, M. Vachharajani, R.B. Lee, Fast subword permutation instructions based on butterfly networks, Proc. SPIE, Media Processor 2000 (2000) 80–86.

[7] X. Yang, R.B. Lee, Fast subword permutation instructions using omega and flip network stages, in: ICCD, 2000, pp. 15–21.

[8] Intel and HP, "Ia-64 application instruction set architecture guide," vol. Revision 1, 1999.

[9] R.B. Lee, X. Yang, Z. Shi, Fast subword permutation instructions using omega and flip network stages, in: Proceedings of Hot Chips 14 - A Symposium on High Performance Chips, 2002, pp. 15–22.

[10] HP, PA-RISC 1.1 architecture and instruction set reference manual", vol. 3, HP Part Number 09740–90039, 1994.

[11] T. Eisenbarth, S. Kumar, A survey of lightweight-cryptography implementations, IEEE Des. Test. Comput. 24 (6) (2007) 522–533.

[12] G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, C. Manifavas, A review of lightweight block ciphers, J. Cryptogr. Eng. (2017).

[13] D. Dinu, Y.L. Corre, D. Khovratovich, L. Perrin, J. Grobshadl, A. Biryukov, Triathlon of lightweight block ciphers for the internet of things," NIST Lightweight Cryptography Workshop 2015, NIST, July 20–21, Gaithersburg, 2015, pp. 1–18.

[14] Bassam J. Mohd, Thaier Hayajneh, Athanasios V. Vasilakos, A survey on lightweight block ciphers for low-resource devices: comparative study and open issues, J. Network Computer Appl., Elsevier (2015) pp. 1-21, https://doi.org/10.1016/j.jnca.2015.09.001i.

[15] Bogdanov et al., PRESENT—An ultra-lightweight block cipher, in: P. Paillier, I. Verbauwhede (Eds.), Cryptographic Hardware and Embedded Systems (Lecture Notes in Computer Science), vol. 4727, Springer-Verlag, Berlin, Germany, 2007, pp. 450–466.

[16] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, T. Iwata, The 128 bit blockcipher CLEFIA, in: A. Biryukov (Ed.), Fast Software Encryption (Lecture Notes in Computer Science), vol. 4593, Springer-Verlag, Berlin, Germany, 2007, pp. 181–195.

[17] The 128 Bit Blockcipher CLEFIA: Algorithm Specification", Sony Corporation 2007 Tokyo, Japan.

[18] D. Hong et al., HIGHT: A new block cipher suitable for low-resource device, in: L. Goubin, M. Matsui (Eds.), Cryptographic Hardware and Embedded Systems (Lecture Notes in Computer Science), vol. 4249, Springer-Verlag, Berlin, Germany, 2006, pp. 46–59.

[19] D.E. Canniere, C. Dunkelman, O. Knezevic, KATAN and KTANTAN—A family of small and efficient hardware-oriented block ciphers, in: Cryptographic Hardware and Embedded Systems, CHES, Springer, LNCS, 2009, pp. 272–288.

[20] R. Beaulieu, S. Treatman-Clark, S. Douglas, S. Weeks, B. Smith, J. Wingers, The SIMON and speck families of lightweight block ciphers, in: 52nd ACM/EDAC/ IEEE Design Automation Conference (DAC), San Francisco, 2013, pp. 1–6.

[21] V. Grosso, G. Laurent, F.-X. Standaert, K. Varici, LS-Designs: Bitslice encryption for efficient masked software implementations, Fast Software Encryption, FSE 2014 vol. 8540 (2014).

[22] Z. Gong, S. Nikova, Y.W. Law, KLEIN: a new family of lightweight block ciphers, in: RFID Security and Privacy, Springer, LNCS 7055, 2012, pp. 1–18.

[23] A. Juels, S.A. Weis, Authenticating pervasive devices with human protocols, in: Advances in Cryptology, Springer-Verlag, Berlin Germany, 2005, pp. 293–308.

[24] National Institute of Standards and Technology (NIST) Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 1997. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, Nov. 26, 2001.

[25] National Institute of Standards and Technology (NIST). (Dec. 30, 1993). Data Encryption Standard (DES), Federal Information Processing Standards Publication 46-2. [Online]. Available: http://www.umich.edu/~x509/ssleay/fip46/fip46-2.htm.

[26] National Institute of Standards and Technology (NIST). (October 25, 1999). Data Encryption Standard (DES), Federal Information Processing Standards Publication 46-3. [Online]. Available: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf.

[27] D. Wheeler, R. Needham, TEA, a tiny encryption algorithm, in: Fast Software Encryption (FSE 1994), Springer, LNCS, 1994, pp. 363–366.

[28] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, Verbauwhede, " RECTANGLE: a bitslice ultra-lightweight block cipher suitable for multiple platforms", Sci. China Inf. Sci. 58 (12) (2014) 1–15.

[29] B. Gerard, V. Grosso, M. Standaert Naya-Plasencia, Block ciphers that are easier to mask: how far can we go?, in: Cryptographic Hardware and Embedded Systems, CHES 2013, LNCS, Springer, 2013, pp 383–399.

[30] D. Engels, X. Fan, G. Gong, H. Hu, E.M. Smith, Hummingbird: ultra-lightweight cryptography for tresource-constrained devices, in: Financial Cryptography and Data Security—FC 2010, Springer, LNCS, 2010, pp. 3–18.

[31] S. Das, "Halka: a lightweight, software friendly block cipher using ultralightweight 8-bit S-box," IACR Cryptology ePrint Archive: Report 110, 2014.

[32] T. Suzaki, K. Minematsu, S. Morioka, E. Kobayashi, Twine: a lightweight, versatile block cipher, in: ECRYPT Workshop on Lightweight Cryptography (LC11), 2011, pp. 146–169.

[33] A. Poschmann, S. Ling, H. Wang, 256 bit standardized crypto for 650 GE GOST revisited, in: Cryptographic Hardware and Embedded Systems,CHES2010, Springer, LNCS, 2010, pp. 219–233.

[34] L. Knudsen, G. Leander, A. Poschmann, M.J.B. Robshaw, PRINTcipher: a block cipher for IC-printing, in: Cryptographic hardware and embedded systems, CHES 2010, Springer, LNCS, 2010, pp. 16–32.

[35] H. Cheng, H.M. Heys, C. Wang, PUFFIN: A novel compact block cipher targeted to embedded digital systems, in: 11th EUROMICRO Conference on Digital System Design Architectures—DSD 2008, Methods and Tools, Parma, Italy, 2008, pp. 383–390.

[36] V. Grosso, G. Laurent, F. Standaert, K. Varici, LS-Designs: Bitsliced encryption for efficient masked software implementations, in: Fast Software Encryption, FSE 2014, Springer, LNCS, 2014, pp. 210–216.

[37] S. Banik, A. Bogdanov, T. Isobe, et al., Midori: A Block Cipher for Low Energy, Advances in Cryptology–ASIACRYPT, Springer, Berlin Heidelberg, 2015.

[38] C.H. Lim, T. Korkishko, mCrypton–a lightweight block cipher for security of low-cost RFID tags and Sensors, Inform. Security Appl., vol. 3786, Springer, LNCS, 2006, pp. 243–258.

[39] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson, TwoFish: A 128-bit block cipher, NIST AES Proposal, 1998.

[40] E. Biham, A. Shamir, Differential cryptanalysis of DES-like cryptosystems, in: Advances in Cryptology - CRYPTO '90, ser. LNCS, Springer, 1990, pp. 2–21.

[41] M. Matsui, Linear cryptoanalysis method for DES cipher, in: Advances in Cryptology – EUROCRYPT '93, ser. LNCS, Springer, 1993, pp. 386–397.

[42] F. Chabaud, S. Vaudenay, Links between differential and linear cryptoanalysis, in: Advances in Cryptology – EUROCRYPT '94, ser. LNCS, Springer, 1994, pp. 356–365.

[43] B. Kaliski, Y.L. Yin, On the security of the RC5 encryption algorithm RSA Laboratories Technical Report TR-602, 1998.

[44] A.Y. Poshmann, Lightweight Cryptography: Cryptographic Engineering for a Pervasive World, Ruhr University Bochum, 2009 (Ph.D. thesis).

[45] R. Benadjila, J. Guo, V. Lomné, T. Peyrin, in: SAC 2013. LNCS, vol. 8282, Implementing lightweight block ciphers on x86 architectures, Springer, Heidelberg, 2014, pp. 324–351.

[46] B.S. Tiago, A. Diego, Julio Opez, PRESENT runs fast: efficient and secure implementation in software, in: International Association for Cryptologic Research, CHES 2017, LNCS, 2017, pp. 644–664.

Corresponding author
C.G. Thorat can be contacted at: chandrama1684@gmail.com