International Journal of Modern Electronics and Communication Engineering (IJMECE) ISSN: 2321-2152
Volume No.-7, Issue No.-3, May, 2019

PRESENT Encryption with Dynamic S-Box using Karnaugh Mapping

Utkarsh Savardekar
Department of Electronics and Communication Engineering
Bangalore Institute of Technology
Bangalore, India
E-mail: savardekaru@gmail.com

Kalpana A B
Department of Electronics and Communication Engineering
Bangalore Institute of Technology
Bangalore, India
E-mail: abkalpana@gmail.com

Abstract: In this work, the goal is to implement a PRESENT cipher onto a Field Programmable Gate Array (FPGA) platform. The proposed design uses a 4-bit data-path to reduce hardware size instead of the conventional 64-bit path. Instead of traditional look-up table (LUT) based S-Box, the S-Box is implemented through Karnaugh mapping. Further factorization is also done to reduce the size of Boolean S-Box. Instead of using one static S-Box, 16 key dependent S-Boxes were used. As a result, FPGA implementation of the PRESENT cipher requires 229 slices on Virtex-5 XC5VLX50.

Keywords: PRESENT, Boolean S-Box, Karnaugh mapping, FPGA

I. INTRODUCTION

As technology improves, more and more devices are connected to the internet in the new technological revolution called "IOT". So, the need for security has increased. From contactless "RFID" to remote sensing operations, there is a need for encryption. Encryption is defined as the method of encoding the data or information which ensures that only legitimate users can access the data thereby securing the information against the third parties. Apart from the prevention of the snooping, encryption also rejects the information or data which is intelligible from the interceptors or non-legitimate users. Though the conventional encryption is sufficient for a normal operation, the resource constraint nature of sensors forbids their use [16]. Hence the need for a new branch of cryptography called lightweight encryption has been evolved. This branch specializes in encryption for resource constraint fields like sensors and RFID. One of the prominent candidate is the PRESENT encryption algorithm.

Developed in 2007 by Andrey Bogdanov, Present algorithm consists of 31 rounds based on the substitution-permutation network (SPN). The key can be either 80-bit or 128-bit depending upon the level of security and application required. The plain text and cipher text are 64-bit. Each round of 31 rounds consists of XOR operation to introduce a round key Ki for $1 \leq i \leq 32$, where K_{32} is used for post-whitening, a linear bitwise permutation and a non-linear substitution layer. The top level algorithm description is given in Fig. 1.

The non-linear layer consists of 4-bit S-Box S which is applied 16 times in parallel in each round. The present S-Box has 4-input and 4-bit output and is one of the smallest one S Box in encryption. The permutation layer is simply rewiring so it's very efficient to implement in hardware since it does not take any area. The S-Box and permutation layer is given in Table I and Table II respectively.

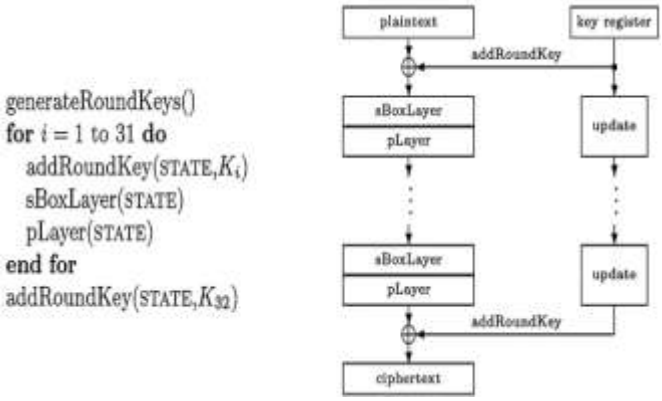

Figure 1. Top level algorithm

In 2008, Carsten Rolfes et al. [2] showed that the datapath of the architecture can be reduced from 64-bit to 4-bit in the serialized architecture. The division can be by any multiple of 4. This architecture has been the basis of many which followed having achieved the lowest areas at the time.

Sufyan Salim Mahmood et al. [5] proposed that having 16 S-Box rather than one, the complexity of each round of S-Box in the algorithm increases. This in turn increases the amount of time required in attacks. J. J. Tay et al. [10] introduced a novel way of storing the S-Box. Rather than using LUTs or BRAMs, they used Boolean functions which were further factorized to store the S-Box. This enabled them to have an architecture which has one of the lowest area at that time.

Andres Lara et al. [11] showed that the size of both the S-Box and the permutation layer can be reduced to a 16-bit data-path. Though this architecture lacked the provisions for key

scheduling. This idea was further developed by them in [13] where they allowed a key scheduling mechanism.

Table 1.THE PRESENT S-BOX [1]

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(X)	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 2. THE BIT PERMUTATION IN THE PRESENT [1]

I	P(i)	I	P(i)	i	P(i)	I	P(i)
0	0	16	4	32	8	48	12
1	16	17	20	33	24	49	28
2	32	18	36	34	40	50	44
3	48	19	52	35	56	51	60
4	1	20	5	36	9	52	13
5	17	21	21	37	25	53	29
6	33	22	37	38	41	54	45
7	49	23	53	39	57	55	61
8	2	24	6	40	10	56	14
9	18	25	22	41	26	57	30
10	34	26	38	42	42	58	46
11	50	27	54	43	58	59	62
12	3	28	7	44	11	60	15
13	19	29	23	45	27	61	31
14	35	30	39	46	43	62	47
15	51	31	55	47	59	63	63

In the proposed system instead of a single S-Box, 16 S-Boxes are used and their values are chosen carefully to resist the linear and differential cryptanalysis. Since S-Boxes are non-linear in nature and constitutes the highest area in the cipher, we use karnaugh mapping to map the S-Box using logical gates rather than LUTs AND BRAMs. The four output bits of PRESENT S-Box give us four sum-of-product equations for each S-Box. To further reduce the area, we factorized the common factors of the 64 expressions. Hence a total of 16 factors were derived. Section 2 gives a brief overview of the proposed algorithm, the S-Boxes and the factors for the S-Boxes. Section 3 displays the implementation results and comparison with related works.

II. PROPOSED ALGORITHM

The overall architecture closely follows the serial architecture in [2]. Two 64-bit registers were used to store the key and the plaintext. The data-path is chosen to be 4-bit since it allows us to use only one S-Box. This is the reason why the [11] architecture was not chosen as it would have made us use four S-Boxes. The S-Box consists of the 16 S-box proposed by [9].

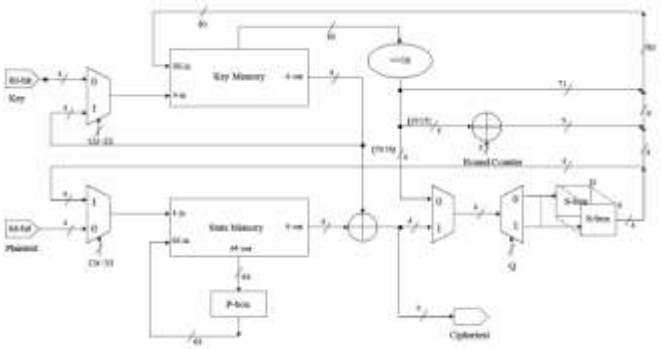

Figure 2. Proposed algorithm

Using the techniques presented in [10] we can decrease the size of the S-Box by using karnaugh mapping on each of the outputs of the 16 S-Box and then factorizing them accordingly to give the most efficient design. The S-Box are then stored in terms of logical gates rather than in the form of LUT or BRAM. Since the proposed S-Box is made up of factorized 16 S-Boxes, it is a better idea to just have one, inherent to its size. The architecture for proposed algorithm is shown in Fig. 2.

A. The S-Box

S-Box stands for Substitution-Box. It is a basic element of the algorithms which does substitution operation in the symmetric key architectures. They are used particularly to make the relationship between the key and the ciphertext unintelligible in block ciphers. This is termed as “property of confusion” by Shannon. Generally, an S-Box transforms the input bits, m into the output bits, n. An S-Box of dimension mxn can be formed as a Look-Up-Table with 2m words (for n number of bits). The 16 S-Box should be able to meet the same characteristics as the present S-Box. These characteristics are related to linear and differential cryptanalysis. After analysis 16 S-Boxes were chosen [9]. The 16 S-Boxes are given as follows:

S_0 = [3,8,15,1,10,6,5,11,14,13,4,2,7,0,9,12];
S_1 = [15,12,2,7,9,0,5,10,1,11,14,8,6,13,3,4];
S_2 = [8,6,7,9,3,12,10,15,13,1,14,4,0,11,5,2];
S_3 = [0,15,11,8,12,9,6,3,13,1,2,4,10,7,5,14];
S_4 = [1,15,8,3,12,0,11,6,2,5,4,10,9,14,7,13];
S_5 = [15,5,2,11,4,10,9,12,0,3,14,8,13,6,7,1];
S_6 = [7,2,12,5,8,4,6,11,14,9,1,15,13,3,10,0];
S_7 = [1,13,15,0,14,8,2,11,7,4,12,10,9,3,5,6];
S8 = [0,3,5,8,6,9,12,7,13,10,14,4,1,15,11,2];
S_9 = [0,3,5,8,6,12,11,7,9,14,10,13,15,2,1,4];
S_{10} = [0,3,5,8,6,10,15,4,14,13,9,2,1,7,12,11];
S_{11} = [0,3,5,8,6,12,11,7,10,4,9,14,15,1,2,13];

S_{12} = [7,12,14,9,2,1,5,15,11,6,13,0,4,8,10,3];
S_{13} = [4,10,1,6,8,15,7,12,3,0,14,13,5,9,11,2];
S_{14} = [2,15,12,1,5,6,10,13,14,8,3,4,0,11,9,7];
S_{15} = [15,4,5,8,9,7,2,1,10,3,0,14,6,12,13,11];

The S-Boxes S1−S8 are Serpent S-boxes, but the S-Boxes S_9 − S_{12} are from [12] while the S-Boxes S_{13} − S_{16} are Hummingbird S-Boxes.

B. Boolean S-Box

As previously done in [10], we also used karnaugh mapping to make simplified S-O-P expressions. Throughout the paper the input bits are denoted as {a,b,c,d} while the output bits as {w,x,y,z} . This resulted in 4 expressions for each S-Box totaling to 64 expressions for all the 16 S-Boxes. The Boolean expressions help us to further optimize the rigid S-box and can be pipelined easily to increase the throughput if need be. The expressions for one of the S-Box are given in Table III.

C. Factorization

Factorization is a method of bringing out the common factors in a Boolean expression which is given in a SOP (Sum-Of-Products). Due to the large number of expressions, a lot of common factors could be deduced. These common factors are operations that need to be calculated once and can be reused over the entire design. This helps us to further decrease the size of the S-Box. The expressions after factorization are given in Table IV.

D. Choosing the S-Box

The key is responsible for the S-Box chosen, the elements of the Key are XOR with themselves to give us the value between 0-15 and that corresponding S-Box is chosen. For instance, consider a 64-bit key which is of 16 digits and each digit is of 4-bit size. Hence, from the range 0 to 15, there are 16 such numbers that can be represented within 4-bit size. This is illustrated in Fig. 3.

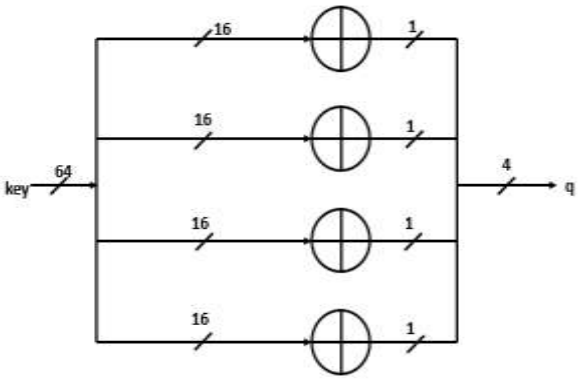

Figure 2. Choosing the S-Box

Table 3. Boolean SOP expressions for one of the 16 PRESENT S-Box generated from Karnaugh mapping

Output	Expression
w	$(a'c'd')+(a'cd)+(ab'd)+(ab'c)+(a'bc)$
x	$(b'cd')+(abc')+(b'c'd)+(a'b'c')+(a'bcd)$
y	$(abd)+(ab'c')+(a'cd')+(a'b'c)+(ab'd')$
z	$(a'cd)+(acd')+(ab'd')+(a'b'd)+(a'bc'd')+(abc'd)$

Table 4. Expressions for corresponding PRESENT S-Box after factorization.

Output	*Expression*
w	$(df_4)+(df_3)+(af_4)+(af_3)+(f_9f_2)$
x	$(cf_8)+(b'f_9)+(af_4)+(af_6)+(f_{11}f_{10})$
y	$(f_6)+(d'f_2)+(df_{11})$
z	$(f_{16})+(d'f_2)+(d'f_1)+(f_7f_{11})$
Factor	*Expression*
f_1	(ab)
f_2	$(a'b')$
f_3	(bc)
f_4	$(b'c')$
f_5	(cd)
f_6	$(c'd')$
f_7	(ad)
f_8	$(a'd')$
f_9	(cd')
f_{10}	$(c'd)$
f_{11}	$(a'b)$
f_{12}	(ab')
f_{13}	$(a'd)$
f_{14}	$(b'c)$
f_{15}	(bd')
f_{16}	$(a'c)$

III. IMPLEMENTATION RESULTS

Our proposed PRESENT cipher is implemented on the Xilinx Virtex-5 XC5VLX50 and requires 229 slices for the design. This uses the same platform as [8] and [10]. The results are shown in Table V. It should be noted that the large area of the proposed is mainly because of the large size of the S-Box. It is believed that area reduction cannot be done as S-Box is non-linear in nature.

By considering the operating frequency RFID sensors of 13.56 MHz [15] we can re-calculate the throughput as given in Table VI. From Table VI, it is observed the throughput far exceeds the upper bound of 100 kbps required for RFID [14]. Hence encryption takes place faster than the rate of data transfer for RFID.

Table 5. Results and Comparison with related works

Design	Flip flops	LUTs	Slices	Latency cycles	No of S-boxes used
[1]	200	300	202	32	1
Proposed	171	473	224	563	16

IV. CONCLUSION

In this paper, we present an implementation of the lightweight block cipher PRESENT on FPGA which utilizes dynamic S-Boxes. The number of S-Boxes was reduced to one by adopting a 4-bit datapath. In addition, we managed to reduce the area requirement of the S-Box through Karnaugh mapping and further factorization. Linear and differential cryptanalysis determines that these S-Boxes are more secure than the standard present algorithm S-Box. The use of dynamic key dependent S-Boxes helps in increasing the security by improving the complexity of S-Box in each round from 2^8 to 2^{12}. Hence the attacker needs more time to recover the key.

REFERENCES

[1] Author A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," in Cryptographic Hardware and Embedded Systems CHES 2007, ser. Lecture Notes in Computer Science, P. Paillier and I. Verbauwhede, Eds. Springer Berlin Heidelberg, 2007, vol. 4727, pp. 450–466.

[2] C. Rolfes, A Poschmann, G. Leander, and C. Paar, "Ultra-Lightweight Implementations for Smart Devices – Security for 100 Gate Equivalents", Smart Card Research and Advanced Applications, Springer, UK, 2008, pp. 89-103.

[3] M. Sbeiti, M. Silbermann, A. Poschmann, and C. Paar, "Design space exploration of PRESENT implementations for FPGAS," in 5th Southern Conference on Programmable Logic, 2009 – SPL, April 2009.

[4] P. Yalla and J. P. Kaps, "Lightweight Cryptography for FPGAs," in International Conference on Reconfigurable Computing and FPGAs, ReConFig '09, Dec 2009, pp. 225–230.

[5] Sufyan Salim Mahmood AlDabbagh and Imad Al Shaikhli," Lightweight Block Ciphers: a Comparative Study", in Journal of Advanced Computer Scienceand Technology Research Vol.2, No.4, November 2012, pp.159-165.

[6] E. Kavun and T. Yalcin, "RAM-Based Ultra-Lightweight FPGA Implementation of PRESENT," in Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on, Nov 2011, pp. 280–285.

[7] ISO/IEC 29192-2:2012, Information technology – Security techniques – Lightweight cryptography – Part 2: Block ciphers, ISO/IEC Std., Jan. 2012.

[8] N. Hanley and M. ONeill, "Hardware Comparison of the ISO/IEC 29192-2 Block Ciphers," in 2012 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Aug 2012, pp. 57–62.

[9] Sufyan Salim Mahmood AlDabbagh and Imad Fakhri Taha Al Shaikhli, "Improving PRESENT lightweight algorithm", International Conference on Advanced Computer Science Applications and Technologies,2013.

[10] J. J. Tay, M. L. D. Wong, M. M. Wong, C. Zhang, and I. Hijazin, "Compact FPGA implementation of PRESENT with Boolean S-Box," in Proc. 6th Asia Symp. Quality Electron. Design, Aug. 2015, pp. 144–148.

[11] C. A. Lara-Nino, M. Morales-Sandoval, and A. Diaz-Perez, "Novel FPGA-based low-cost hardware architecture for the PRESENT block cipher," in Proc. Euromicro Conf. Digit. Syst. Design, Aug./Sep. 2016, pp. 646–650.

[12] G. Leander and A. Poschmann, "On the Classification of 4 Bit S-Boxes Arithmetic of Finite Fields." vol. 4547, C. Carlet and B. Sunar, Eds., ed: Springer Berlin / Heidelberg, 2007, pp. 159-176.

[13] Carlos Andres Lara-Nino, Arturo Diaz-Perez, Miguel MoralesSandoval," Lightweight Hardware Architectures for the Present Cipher in FPGA", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, 2017.

[14] C. Alippi, A. Bogdanov, and F. Regazzoni, "Lightweight Cryptography for Constrained Devices", 2014 International Symposium on Integrated Circuits (ISIC), IEEE, Singapore, 2014, pp. 144-147.

[15] "Identification cards – Contactless integrated circuit cards – Proximity cards – Part 2: Radio frequency power and signal interface", ISO/IEC 14443-2:2010, ISO, 2010.

[16] Panasenko, S., Smagin, S., "Lightweight Cryptography: Underlying Principles and Approaches", International Journal of Computer Theory and Engineering, Vol 3 No.4, (2011).