International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 19 (2017) pp. 8092-8096

Implementation of the cryptographic algorithm "Present" in different microcontroller type embedded software platforms

Edwar Jacinto G[1]., Holman Montiel A.[2] and Fernando Martínez S.[3]

Technological Faculty, District University Francisco José de Caldas, Bogotá D.C., Colombia.

[1]ORCID ID: 0000-0003-4038-8137, [2]ORCID ID: 0000-0002-6077-3510
[3]ORCID ID: 0000-0003-2895-3084

Abstract

Given the proliferation of portable applications using low-cost devices and energy consumption, they have generated the need to provide security solutions in embedded microcontroller-type software devices, which, in addition to processing and storing information locally, are capable of providing Security to the information transmitted, using "lightweight" algorithms, in this case, the implementation of the "Present" algorithm is shown, an algorithm that, up to date has not been breached, such algorithm has characteristics that make it suitable to be programmed devices with a small memory and reduced sizes; many tests have been performed in several microcontroller type embedded software platforms, using structured C language, applying a series of tests and performance measurements to verify their operation and some parameters of interest.

Keywords: Block Ciphers, Lightweight algorithms, Cryptology, Embedded software.

INTRODUCTION

Considering the need to connect different characteristics devices used in applications with customized needs, but always having the limitation of cost and low "*throughput"*[1] or data transfer, it is imperative to use microcontroller software embedded platforms[2],[3] that can perform "*lightweight"* cipher algorithms[4],[5],[6] but also having a low cost dedicated hardware device.

Many of the applications using microcontroller platforms are wireless in nature with low *throughput* [1],[7],[8], some of the most used applications are sensor networks[9], RfID, PAN networks, Xbee and some other wireless protocols [6],[10].

This work shows a security scheme in microcontroller embedded devices, based on block ciphers [11],[12], among them, there are many algorithms designed to be implemented in embedded software, specifically in microcontroller type platforms [2],[3],[13], this way, it is decided to carry out the implementation of the "*Present*" lightweight blocking algorithm [14],[15],[16],[17],[8], in addition to conduct the implementation, some tests are made to verify their operation and performance.

The algorithm "*Present*" is a block cipher algorithm of reduced size and execution, since it uses substitution and permutation blocks of only 4 bits, also, the key size is small compared to other algorithms of the same kind, this feature together with its reduced number of rounds makes it an algorithm that balances the use of the microcontroller internal memory with the communication "*throughput"* speed achieved [1],[18],[19].

METHODOLOGY

To give security to information in this type of applications, it is necessary to choose a scheme in which it is clear how the information will be cipher, what kind of algorithm, how the secret key is shared, and in general the type of cryptographic scheme to be used.

In this case a symmetric ciphering scheme is chosen, in which there is a single secret key, known by the sender and the receiver, this type of ciphering has as a characteristic and advantage, and it has a very short execution time, which makes it available to be used in high-speed telematic applications and/or applications in which devices of low computational power are used.

Types of Ciphers

A clear message m, can be stored as an octets variable number "bytes", at the time of performing the ciphering to send the secure message. There are two options of ciphers:

- *Flow ciphering*: Consists in taking a clear message as a continuous stream of bits from a clear text message and by means of a continuous process generating a continuous stream of coded bits.
- *Block ciphering*: In this case the message is divided into blocks of *n* bits each[8],[14], and in this case regardless of the message order, each block is encrypted in the same way; All bits of the message block take part in all operations (rounds), to "disconnect" the possible relationships they had in the original message.

Block ciphers basic operations

In this case a block cipher called *Present* was chosen, which complies with the basic methodology of block ciphers and has the structure shown in Figure 1. In all block ciphers a combination of ByteSub, ShiftRow, MixColum and AddRoundKey or their equivalents is generated [1], [11].

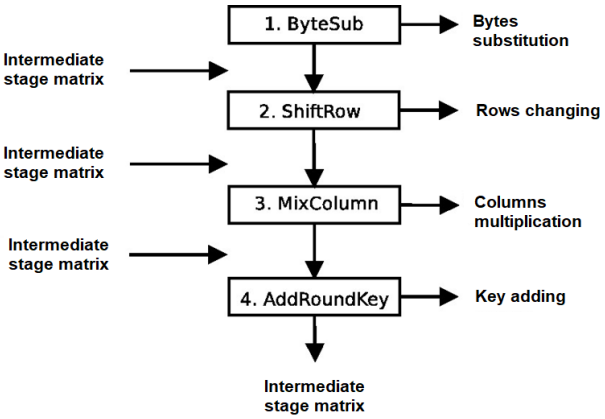

Figure 1: Basic round operations in a typical block cipher [1]

The combination of these operations is called round and must have operations based on uniform and invertible transformations, called layers, which have been designed to withstand linear and differential cryptanalysis, which are:

- *Non-linear layer*: Which consists on the application of S-boxes in parallel with non-linearity optimal properties.
- *Linear mixing layer*: It guarantees a high level of diffusion throughout the multiple rounds.
- *Key addition layer*: This is a unique OR operation between the intermediate stage and the unique key of each round.

PRESENT algorithm structure

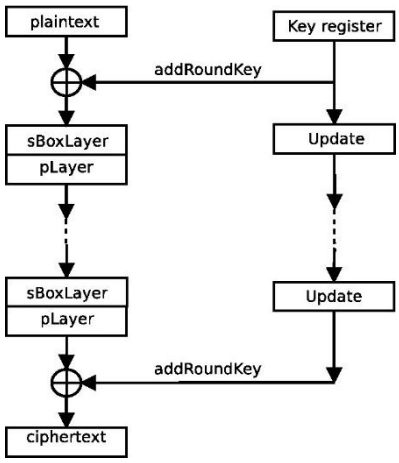

Figure 2: PRESENT algorithm round[1]

Figure 2 shows PRESENT algorithm basic structure, which is expressed in functional blocks, that can be easily taken to source code in software and hardware, 31 rounds must be made in the information encryption process [20].

***Byte substitution layer:* SboxLayer.** It consists of a nonlinear substitution block applied to each stage matrix nibble independently, it is called S-Box layer, shown in table 1. This layer generates new information not linearly connected with the original information[20],[17]. This substitution block is applied to 16 nibbles completing 64 bits of information, which is the standard size of the cipher blocks. This substitution table is made by searching in a table with 4 bits masks, all of this based on the information of a microcontroller with a higher size data bus.

Table 1: S-box from PRESENT algorithm [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

Bit Permutation: pLayer. It is a layer that mixes by bit to bit substitution, a 64 bits information block, where Bit i of the round is moved to the P position (i); The order of such substitution is shown in Table 2.

Table 2: Player substitution table of the PRESENT algorithm

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

Key expansion function: addRoundKey. It is one of the most important blocks, since it is the generation of new keys for each round, for this specific implementation, there was only an implementation for 80-bit keys, for that reason, a vector K of 80 positions was generated, as it can be seen below $K_{79}K_{78}...K_0$. In each round only the most significant 64 bits of the new K_i key of the next round will be mixed, the way in which the rotation must be made, is shown below:

$$K_i = K_{63}K_{62}\dots K_0 = K_{79}K_{78}\dots K_0 \qquad (1)$$

Then the following operations must be performed, in the order in which they are presented:

Bit rotation of the input key:

$$[K_{79}K_{78}\dots K_0] = [K_{18}K_{17}\dots K_{20}K_{19}] \qquad (2)$$

Substitution by using S-Box for the nibble (4 bits) K_{78} to K_{76} of the key:

$$[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}] \quad (3)$$

Nibble adding or mixing K_{19} to K_{15} of the key with the round counter, through the operation XOR:

$$[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \; XorRoundCounter \quad (4)$$

The addRoundKey operation must be made in each of the necessary rounds to cipher the information, otherwise, it is important to bare in mind that in the deciphering process, blocks are taken from the list of K_i keys of the final values until, until reaching the initial ones, which means, the last key used to cipher, will be the first one used to decipher, so, before starting the deciphering process all the rounds of addRoundKey must be made until getting to the last K_i by making an invers process back to the original key.

Algorithm implementation structure

All ciphering algorithms by blocks or by flow should show their implementation details and even though should not be susceptible to be breached, in this case a Pseudo code of the algorithm that describes it and makes possible its implementation in a microcontroller is shown:

```
generate RoundKeys()
for i=1 to 31 do
addRoundKEy(STATE,Ki)
SBoxLayer(STATE)
pLayer(STATE)
end for
addRoundKey(STATE,K32)
```

Figure 3: Structure of the PRESENT algorithm expressed in Pseudocode.

Each of the code lines shown in Figure 3 are functions or methods of the Present_Cypher class, for the substitutions of the pLayer and Sbox, a code that uses tables in the Flash memory of the microcontroller was used [21],[2], which reduces the algorithm execution time, but increases the amount of memory required for its implementation.

IMPLEMENTATION

Microcontroller type embedded platforms

Before implementing the algorithm, a review was made on the available platforms in the local market, considering their architecture, price, available compilers, programming languages, memory and performance [2].

After performing this analysis, it was decided to perform tests with several devices, for the 8 bits, it was used the family 16F and 18F from the company Microchip, due to its leadership in the world market; for 16 bits, it was used the family 24F from the same company, as it is having a retro compatibility with 8-bit families. The above-mentioned devices have compilers, simulators, and verification tools licensed by the Universidad Distrital. As for the 32-bit microcontrollers an ARM platform is selected in its smaller Cortex-M0 family, as it is the direct competition to 8-bit microcontrollers. Specifically, the work was made with the KL25Z platform, since it has an on-line compiler that supports C/C++, pointer management, dynamic memory allocation, version control with mercurial/git. ARM microcontrollers are entering into the market, through their on-line platform at www.mbed.org, they allow compilation regardless the hardware or the device manufacturer, which makes them very versatile and appropriate for many tasks, including those that related to ciphering algorithms.

Algorithm implementation

In block ciphers, there is a combination of operations called round, the security provided by the algorithm depends on the complexity of these operations and the number of rounds needed to cipher. In this case, for the *PRESENT*, it is summarized in a pseudocode shown in Figure 4, the functions shown in it are standard from the C language and can be compiled in a personal computer or in a microcontroller.

For each of the pseudocode lines a function was created, which performs the necessary actions to mix the information as required by the algorithm, below there is a list of the functions generated in C language, which make each of the Functions of the algorithm:

*bool read_bit_n_bits (unsigned int *datap, int index);*

*void write_bit_n_bits(unsigned int *datap, bool val, int index);*

void p_layer(void);

void s_box_layer(void);

void data_update(void);

void key_update(void);

void data_xor_key(void);

Read and write functions were designed to feed the functions that perform substitution and mixing operations; they were written differently for each width of the bus processor, in addition they were written differently, considering if the processor had a hardware for bit management, these functions perform the appropriate masking according to the CPU width.

To the code written in C, adapted to each of the platforms including a personal computer, was tested by using the test vectors as shown in Figure 4., which verify the algorithm operation, in this case on the PC.

Test vectors for PRESENT with an 80-bit key are shown in hexadecimal notation.

plaintext	*key*	*ciphertext*
00000000 00000000	00000000 00000000 0000	5579C138 7B228445
00000000 00000000	FFFFFFFF FFFFFFFF FFFF	E72C46C0 F5945049
FFFFFFFF FFFFFFFF	00000000 00000000 0000	A112FFC7 2F68417B
FFFFFFFF FFFFFFFF	FFFFFFFF FFFFFFFF FFFF	3333DCD3 213210D2

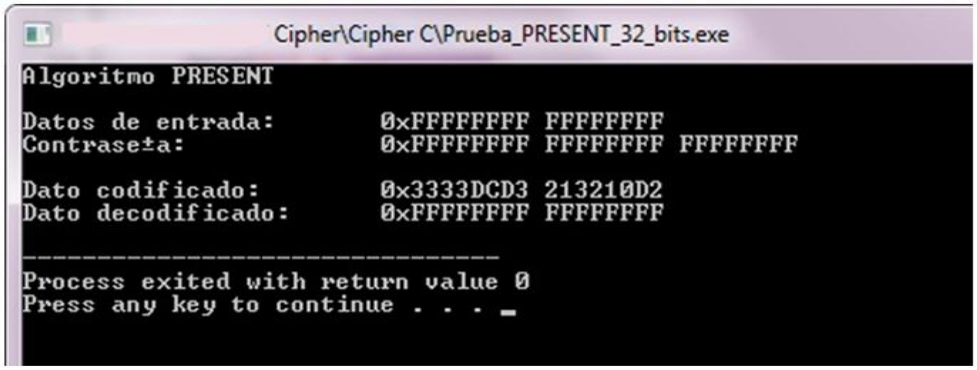

Figure 4: PRESENT algorithm Tests on a PC.

Table 3: Performance parameters for different platforms

Microcontroller	Program memory Flash (bytes)	Data memory Ram (bytes)	Throughput (m/s)	Pointers	MIPS	Binary Operators
8 bits	2210	73	12	no	5	yes
8 bits	2129	768	12,12	yes	5	yes
16 bits	3276	245	18,8	yes	10	no
32 bits	20,5k	600	234	yes	48	no

After verifying the correct operation of the algorithm, real tests were performed on each of the platforms, initially for microcontrollers of 8 bits (families 16F and 18F) without and with pointers, then for the 16bits microcontrollers (24FJ) and at the end in a 32-bit ARM architecture microcontroller. These tests were made by generating a change in an output pin each time the algorithm performed a new encryption.

RESULTS

When performing these tests, it was verified that the algorithm performed the encryption work regardless of the platform, after which some algorithm performance parameters were defined: program memory (FLASH), data memory (RAM) throughput measured in cipher packs per second and their equivalent in bits/second. Table 3 shows the results for the different devices:

A description of the algorithm in standard C language was made, which runs on any platform and it is compatible with the different architectures, especially with the 32-bits ARM architectures, which having pointers and bit oriented operations, achieved in almost 20 times, the throughput of 8-bit microcontrollers with a very similar cost.

CONCLUSIONS

Present is an ultra-lightweight algorithm with one of the most compact encryption methods. Due to these characteristics, it is used in applications of low power consumption. Its performance on different platforms was studied, with the intention of finding ideal conditions for high performance applications. A total of four custom-made implementations were achieved on four different fixed-hardware microcontrollers.

Such implementations require the study of previous arithmetic concepts in finite fields, basic operations of any cryptographic algorithm, including Present.

With a deep knowledge of the internal architecture of the devices, an analysis was made on how to perform suitable implementations in 8, 16, 32 and 64 bits with lower computational cost than the earlier reported in previous applications.

It is important to emphasize that in the research made previously to this project, regarding this subject, it was not possible to find any reported implementation of this algorithm on 32-bit microcontrollers, which makes this work unique.

ACKNOWLEDGMENTS

This work was supported by the District University Francisco José de Caldas, in part through CIDC, and partly by the Technological Faculty. The views expressed in this paper are not necessarily endorsed by District University. The authors thank the research group ARMOS for the evaluation carried out on prototypes of ideas and strategies.

REFERENCES

[1] J. Attridge, "An Overview of Hardware Security Modules," *SANS Institute, InfoSec Read. Room*, vol. 1, no. 1, pp. 1–10, 2002.

[2] R. Azuero, E. Jacinto, and J. Castano, "A low-memory implementation of 128 AES for 32 bits architectures.," in *En Congreso Argentino de Sistemas Embebidos CASE 2012*, 2012, pp. 67–73.

[3] M. Kumar and A. Singhal, "Efficient implementation of advanced encryption standard (AES) for ARM based platforms," *2012 1st Int. Conf. Recent Adv. Inf. Technol. RAIT-2012*, no. November 2001, pp. 23–27, 2012.

[4] H. A. Alkhzaimi and M. M. Lauridsen, "Cryptanalysis of the SIMON Family of Block Ciphers," *Tech. Univ. Denmark*, vol. 1, no. 1, pp. 1–26, 2013.

[5] Z. Gong, S. Nikova, and Y. W. Law, "KLEIN: A new family of lightweight block ciphers," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7055 LNCS, pp. 1–18, 2012.

[6] T. Eisenbarth and S. Kumar, "A Survey of Lightweight-Cryptography Implementations," *IEEE Des Test Comput*, vol. 24, no. 6, pp. 522–533, 2007.

[7] N. Sklavos, "Cryptographic hardware & embedded systems for communications," in *Satellite Telecommunications (ESTEL), 2012 IEEE First AESS European Conference on*, 2012, pp. 1–6.

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

[9] A. Delgadillo, N. Pena, and M. Guerrero, "Diseno de un criptosistema para redes de sensores inalambricos WSN basado en MPSOC," Universidad de los Andes, 2008.

[10] D. Huang and H. Kapoor, "Towards Lightweight Secure Communication Protocols for Passive {RFID}s," in *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, 2009, pp. 1–9.

[11] A. Aysu, E. Gulcan, and P. Schaumont, "{SIMO}N Says: Break Area Records of Block Ciphers on {FPGA}s," *IEEE Embed. Syst. Lett.*, vol. 6, no. 2, pp. 37–40, 2014.

[12] S. Feizi, A. Ahmadi, and A. Nemati, "A hardware implementation of Simon cryptography algorithm," in *Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on*, 2014, pp. 245–250.

[13] S. Engels, E. B. Kavun, C. Paar, T. Yalcin, and H. Mihajloska, "A Non-Linear/Linear Instruction Set Extension for Lightweight Ciphers," in *Computer Arithmetic (ARITH), 2013 21st IEEE Symposium on*, 2013, pp. 67–75.

[14] R. Beaulieu, D. Shors, and J. Smith, "The Simon and Speck Block Ciphers on AVR 8-bit Microcontrollers," *Light. 2014 Proc.*, vol. 1, no. 1, pp. 1–18, 2014.

[15] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK Families of Lightweight Block Ciphers." 2013.

[16] A. Bogdanov, D. Khovratovich, and C. Rechberger, "Biclique cryptanalysis of the full AES," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7073 LNCS, pp. 344–371, 2011.

[17] J. Pospiil and M. Novotny, "Evaluating Cryptanalytical Strength of Lightweight Cipher PRESENT on Reconfigurable Hardware," in *Digital System Design (DSD), 2012 15th Euromicro Conference on*, 2012, pp. 560–567.

[18] F. M. Qatan and I. W. Damaj, "High-speed KATAN ciphers on-a-chip," in *Computer Systems and Industrial Informatics (ICCSII), 2012 International Conference on*, 2012, pp. 1–6.

[19] S. Mane, M. Taha, and P. Schaumont, "Efficient and side-channel-secure block cipher implementation with custom instructions on {FPGA}," in *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, 2012, pp. 20–25.

[20] A. Bogdanov *et al.*, "PRESENT : An Ultra-Lightweight Block Cipher," *Springer Berlin Heidelb.*, pp. 450–466, 2007.

[21] E. B. Kavun, G. Leander, and T. Yalcind, "A reconfigurable architecture for searching optimal software code to implement block cipher permutation matrices," in *Reconfigurable Computing and FPGAs (ReConFig), 2013 International Conference on*, 2013, pp. 1–8.