10.53477/2668-2028-21-33

LIGHTWEIGHT CRYPTOGRAPHIC ALGORITHM IMPLEMENTATION IN A MICROCONTROLLER SYSTEM

Alexandru VULPE, PhD
Research staff, Faculty of Electronics, Telecommunications and Information Technology, University Politehnica of Bucharest
alex.vulpe@radio.pub.ro

Raluca ANDREI
Research staff, Faculty of Electronics, Telecommunications and Information Technology, University Politehnica of Bucharest
raluca.andrei@upb.ro

Alexandru BRUMARU
INVITE Systems
alexandru.brumaru@invite-sys.ro

Octavian FRATU, PhD
Research staff, Faculty of Electronics, Telecommunications and Information Technology, University Politehnica of Bucharest
ofratu@elcom.pub.ro

Abstract: *With the development of mobile devices and the advent of smartphones, the Internet has become part of everyday life. Any category of information about weather, flight schedule, etc. it is just a click away from the keyboard. This availability of data has led to a continuous increase in connectivity between devices, from any corner of the world. Combining device connectivity with systems automation allows the collection of information, its analysis and implicitly decision-making on the basis of information. Their introduction and continued expansion of devices that communicate in networks (including the Internet) have made security issues very important devices as well as for users. One of the main methodologies that ensures data confidentiality is encryption, which protects data from unauthorized access, but at the cost of using extensive mathematical models. Due to the nature of IoT devices, the resources allocated to a device can be constrained by certain factors, some of which are related to costs and others to the physical limitations of the device. Ensuring the confidentiality of data requires the use of encryption algorithms for these interconnected devices, which provide protection while maintaining the operation of that device. The need for these types of algorithms has created conditions for the growth and development of the concept of lightweight encryption, which aim to find encryption systems that can be implemented on these categories of devices, with limited hardware and software requirements. The paper proposes a lightweight cryptographic algorithm implemented on a microcontroller system, comparing its performances with those of the already existing system (based on x86).*

Keywords: *Lightweight cryptography; security; Internet-of-Things; microcontroller; implementation.*

Introduction

The novelty of lightweight cryptographic algorithms is that they address an extremely dynamic area of IoT devices that use very low power microcontrollers, which allow them to dedicate only a small part of their capacity to security.

An example of such a field of use is that of sensor networks. These networks aim to connect a very large number of simple sensors to a central hub, and lightweight algorithms must ensure in the communication channels: security, authenticity and integrity of messages. Within this scope, encryption algorithms must be "light" to limit the power consumption in addition to that required by the basic functionality[1].

Another novelty example for the successful use of lightweight algorithms is in the field of mobile applications, being a constantly increasing trend of loading sensitive data on smartphones, such as banking, messaging accounts, access to online social networks, monitoring systems distance, personal data of users.

[1] MCKAY et al., NISTIR 8114, *Report on Lightweight Cryptography*, DOI: 10.6028/NIST.IR.8114

The paper is organized as follows. Section 1 presents on overview of lightweight cryptography, while Section 2 presents the hardware development and implementation of a lightweight cryptography algorithm, namely PRESENT algorithm. Section 3 presents results of the performance evaluation of the implemented algorithm, while Section 4 provides a discussion of our findings.

Lightweight cryptography

Lightweight cryptographic algorithms, as we showed in the introduction, find their utility in the context of the developing and expanding range of IoT devices, whose limited resources do not allow the use of reasonable symmetric cryptographic algorithms under reasonable conditions.

The concept of lightweight cryptographic algorithms is the creation of algorithms that require fewer hardware and software resources. At the same time, these algorithms must ensure the security and performance of the devices without increasing the costs of research and development.

The current standard of lightweight cryptography contains two algorithms called PRESENT and CLEFIA, with, theoretically faster results compared to the generic symmetric AES algorithm.

The PRESENT Algorithm

The PRESENT lightweight cryptographic algorithm was introduced starting with 2007 with the objective of operating in restricted environments. When PRESENT is implemented on a device, it can include either the encryption and decryption function, or just the encryption function, the decryption being performed in another location. The PRESENT algorithm, similar to the AES, uses substitutions and permutations, generating a network, known as the Substitution and Permutation Network (SPN).

Encryption in the PRESENT algorithm

The encryption within the PRESENT lightweight algorithm is performed through a process with 31 rounds. The initial input, as well as the input of each round, is a 64-bit state array and a 64-bit round key, with two acceptable key lengths: 80 and 128 bits.

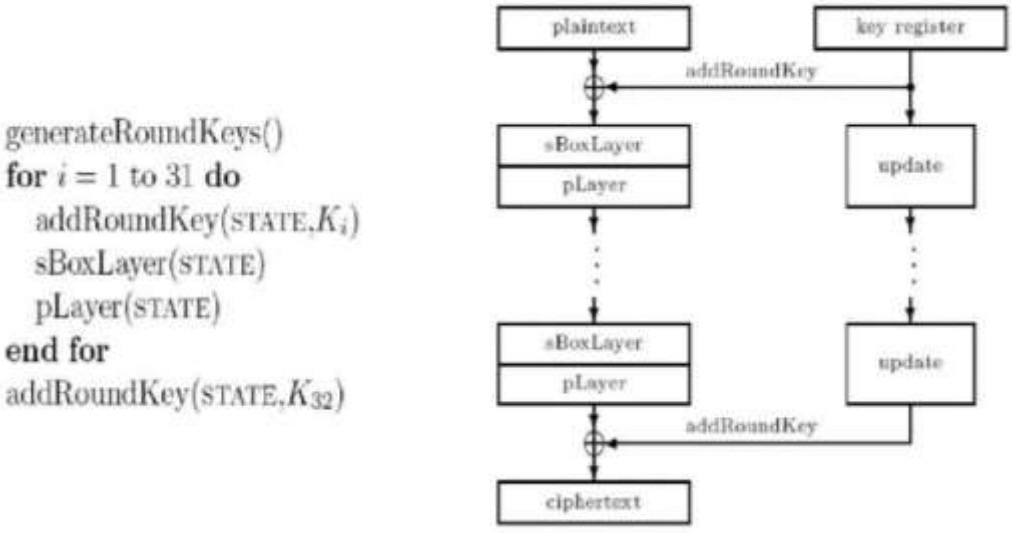

Figure 1. PRESENT Algorithmic Description[2]

[2] *Ibidem*.

The first step is *addRoundKey* which realizes an XOR between the bits of the state matrix and those of the round key, following that the new state is introduced in the repetitive cycle.

The next step of the encryption process is in the *sBoxLayer* function, in which the state matrix is broken into 4-bit groups, and by using 16 Sboxes the 4-bit groups are processed in parallel.

Table 1. S-Box PRESENT

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

Subsequently, the *pLayer* step performs the bit permutation, according to a table, in which the bit of the state matrix i is moved to position $P(i)$.

Each of the 31 rounds consists of an XOR operation to enter the key for the current K_i round for $1 \leq i \leq 32$. The *SBoxLayer* box is applied 16 times in parallel in each round.

Implementation

The Tiva TM4C129X development kit provides a feature-rich development platform that allows users to evaluate Tiva TM4C129x series microcontrollers (MCUs).

The development board has a 10/100-bit Ethernet interface, QVGA color display with touch screen, In-Circuit Debugging Interface (ICDI), microSD card slot, USB Host capability, OTG capability and several connectors for extensibility[3].

The PRESENT lightweight cryptographic algorithm was implemented in a C source file, in Header format (*PRESENT.h*). The implementation has a modular aspect, and is made based on the concept of dynamic programming, the problem of encryption / decryption being broken into elementary procedures. Emphasis was placed on optimization as much as possible, so data structures as compact as possible and operations as simple as possible were used. The PRESENT.h file contains both working memory allocations and substitution / permutation tables, basic level procedures for byte / bit operations, and high-level procedures for encryption and decryption. In the case of the PRESENT algorithm, they consist of repetitive substitution and permutation functions as per the procedure described in Section 1.

For substitution, PRESENT assigns a different value to each nibble (half byte; 4 bits) when processed through the S-Box, the allocation of substitutions being corresponding to the table in Figure 2.

Each of the 16 possible 4-bit combinations that make up a nibble is replaced with a different value. Since a nibble is represented by a single hexadecimal character, the implementation of the substitution vector was chosen in the form of an integer of 64 bits, where every 4 consecutive bits (counted from right to left, the least significant number being written by convention on the right) are noted as the value it substitutes.

For permutations, byte unpacking and wrapping procedures were used to move the bits from one position to another.

The third and last basic element of the cipher is the key schedule. It defines how the key changes at each round (iteration) of the algorithm, and is performed in the form of successive bitwise operations on the encryption/decryption key, as defined in the documentation of the PRESENT cipher.

Results

To obtain information about the performance of an algorithm, it must be run in a controlled benchmark environment. A benchmark is a type of synthetic test, in which a hardware and/or software component is subjected to repeated tests, recording relevant performance indices, such as memory usage or runtime, in most cases.

[3] A. Bogdanov, *PRESENT: An Ultra-Lightweight Block Cipher*, http://www.lightweightcrypto.org/present/present_ches2007.pdf [Accesed 29.01.2021].

In the case of testing the PRESENT algorithm, it is desired to observe the execution time. This requires a timer mechanism that runs independently of the processor to measure the speed of the algorithm.

The Tiva Tiva-C development platform used in this project has an ARM Cortex-M4 processor. The SysTick component, or system timer, a function encountered in Cortex-M series ARM processors was used. Since the Tiva-C development board used in this project has a 16MHz oscillator, it resulted in an interrupt frequency of 1KHz, so 1,000 times per second, thus measuring milliseconds by calling the *SysTick_Handler* function.

The benchmark program verifies the performance of the cipher implemented in the PRESENT software, comparing it with that of the AES-ECB cipher implemented in the hardware on the Tiva-C platform. To compare performance analysis, both algorithms go through the same number of blocks to encrypt and decrypt. Here, PRESENT uses blocks of 64 bits in length, while AES-ECB uses blocks of 512 bits, 8 times larger. Basically, we can say that for blocks encrypted/decrypted by the PRESENT, only one block passed through the AES-ECB must be tested.

Knowing the differences in block lengths and wanting a test that runs the same amount of data through both algorithms, the 64-bit block will be used as a unit of measure for the data size. To get the most accurate results, both algorithms were tested by encrypting and decrypting a number of 1,000 64-bit blocks

With all the necessary functionalities, the performance test was performed for the two algorithms. It went through the following steps:

For each algorithm:

1. Reset *g_millis* to 0 to start counting milliseconds within the algorithm
2. Iterate the algorithm to process *g_blocksToTest* 64-bit blocks
3. Check the encrypted text to ensure that the algorithm ran correctly
4. Display total time, in milliseconds

To measure the execution time for the two algorithms, a file with a length of 7,813 KB was used, representing a number of 1,000 64-bit blocks. The program successfully tests the processing times in terms of data encryption / decryption using the AES-ECB cipher and PRESENT, but it is important to consider that AES-ECB is natively supported in microcontroller hardware, and PRESENT is a cipher implemented in software.

The results of the measurement are presented in the table below and Figure 2:

AES	PRESENT
391 ms	6548 ms

Figure 2. Result of execution time measurement for each algorithm and GUI display

Discussion

The main program compares the AES-ECB encryption algorithm supported by the microcontroller on the TI Tiva-C development board, with the PRESENT algorithm, implemented in this paper.[4]

Thus, while AES-ECB takes advantage of bit-level logical configurations for executed operations, PRESENT must be run on the microcontroller processor, with operations defined as instructions compiled in assembly language more complex than the case of AES-ECB, which has dedicated instructions already present at the transistor level.

Moreover, PRESENT has to do its bit operations using mathematical manipulations when it comes to bit permutations and bit block operations that are not perfectly aligned as bytes.

We can thus say that where the AES-ECB is executed by a dedicated hardware unit in the microcontroller used, PRESENT is implemented as an "emulation" in the software of what could be a cipher supported in the hardware on another chip.

Thus, comparing the execution speed of AES-ECB with that of the PRESENT encryption algorithm in this situation, is not likely to show how fast AES-ECB is compared to PRESENT, but is a test that demonstrates the feasibility of the lightweight cryptographic algorithm PRESENT, in the form of a software implementation with execution speeds that might be acceptable for applications that are not delay-constrained for an encryption process that can run on a microprocessor without dedicated instructions and with very low memory usage.

Conclusions

This paper presented a lightweight cryptographic algorithm (PRESENT) implemented on a microcontroller system. The system mimics a constrained device, with limited hardware and software, which could support such an encryption mechanism.

The PRESENT algorithm was tested in comparison with the traditional AES algorithm with the objective of measuring and observing the execution time as an indicator of the delay that might affect the encryption process.

The results have shown that AES actually fares better than PRESENT, due to the fact that it is implemented in an optimized manner, taking advantage of the hardware encryption provided by the microcontroller system.

Acknowledgement

This work was supported by a grant of the Ministry of Innovation and Research, UEFISCDI, project number 8Sol/2018 within PNCDI III and in part under the research subjects "e-Government" during the sustainability timeframe of the project "Endowing investment at Ad Net Market Media laboratories for R&D in future mobile communications" – FUTURE-NET-LAB, funded by the EU under the Operational Program Competitivity (POC), priority axis 1, action 1.1.1:. Large R&D infrastructures.

BIBLIOGRAPHY

1. MCKAY et al., NISTIR 8114, *Report on Lightweight Cryptography*, DOI: 10.6028/ NIST.IR.8114
2. BOGDANOV A., *PRESENT: An Ultra-Lightweight Block Cipher*, http://www.light weightcrypto.org/present/present_ches2007.pdf [Accesed 29.01.2021]
3. *TivaTM TM4C129X Development Board Datasheet*, https://www.ti.com/lit/pdf/spms 444 [Accesed 24.01.2021]

[4] TivaTM TM4C129X *Development Board Datasheet*, https://www.ti.com/lit/ pdf/spms444, Accesed of 24.01.2021.