Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 11, No. 2, June 2023, pp. 494~502
ISSN: 2089-3272, DOI: 10.52549/ijeei.v11i2.4264

Hardware Security Module Cryptosystem Using Petri Net

Billel Guechi[1], Mohammed Redjimi[2]
[1]Department of Computer Science, Badji Mokhtar University, Algeria
[2]Department of Computer Science, 20 August 1955 University, Algeria

Article Info

Article history:

Received Oct 28, 2022
Revised Mar 9, 2023
Accepted May 12, 2023

Keyword:

Secure SoC
HSM
Time to market
Petri net
Encryption/Decryption System

ABSTRACT

An embedded system is a combination of hardware and software designed to perform specific functions. It consists of SoCs (system on chip) that it relies on to do its computing work. A key feature of an embedded system is that it consumes less power and components occupy less space on the IC (integrated circuit) thus, the use of SoCs. Embedded system manufacturers get these SoCs from third-party companies to reduce their time to market. That would increase the possibility of the systems to be compromised. In this paper, we present a novel approach to securing such critical systems. For that, we made a Hardware Security Module (HSM), which consists of secure SoC with encrypt/decrypt engine that use Petri net for algorithm modulation to secure data flow. We ensure that the system uses genuine firmware and data is secured since we use encrypt/decrypt algorithms only known to manufacturers.

Corresponding Author:

Billel Guechi,
Department of Computer Science,
Badji Mokhtar University,
Badji Mokhtar University -Annaba- B.P.12, Annaba, 23000 Algeria.
Email: billel.guechi@univ-annaba.org

1. INTRODUCTION

Nowadays, embedded systems are used everywhere because of their lightweight and few hardware/software resources are required to design and implement such systems. They are the key feature of applications for sensitive fields such as military, aviation, automotive, secure communications, and the internet of things (IoT). The huge demand for these systems requires more chips to be produced, which forced companies to no longer produce hardware components on their own to meet time to market, so they outsource these chips from third-party companies. Thus, security becomes a high-priority factor to be satisfied to ensure the system's integrity and performance in achieving its design purpose [1].

The approach of outsourcing chips opens the door for significant security issues that cannot be underestimated [2]. Reverse engineering and hardware intrusion may result in encryption/decryption secret key disclosure, which leads to devastating consequences such as random or wrong results, and sensitive data leakage.

Our method consists of making a Hardware Security Module with an encrypt/decrypt engine based on a Petri net for algorithm modulation. The system components integrity is ensured by using Secure SoC architecture so that we provide physical protection. Data flow is completely secured by using encrypt/decrypt engine. The whole system implementation and study case have been presented.

Existing solutions using HSMs to deploy security services have been the subject of much research in many different fields, such as network equipment [4], Agriculture sensors security [5], and biomedical data [6]. We structure our paper like so. Section 2 presents a set of related works that used HSM as a security module to encrypt sensitive data. In section 3 we present our technique to create HSM dotted with a powerful encrypt/decrypt engine that uses Petri net for encryption/decryption algorithm and secure SOC as architecture. Section 4 provides obtained result with a full study case demonstration and a precise discussion of system strengths/weaknesses. Section 5 concludes this paper alongside future work highlights.

2. BACKGROUND

Due to the globalization of the Integrated Circuit (IC) manufacturing industry, hardware Trojans constitute a frightening threat to a great number of embedded system applications. Traditional testing methods are insufficient for finding hardware Trojans; several specialized detection methods have emerged. For a better understading of how hardware Trojans work and what damage they would cause to an IC, knowing their threat levels and taxonomy is a must. Our approach to handling the impact of such taxonomies is to make a hardware security module (HSM) in order to ensure the integrity of data by using the encryption system that relies on petri nets.

2.1. The threat levels of the hardware Trojan

We notice two major levels at which hardware Trojans would actively operate on it. That would be on the design level alongside the foundry level [3].

At the design level, the specifications of the entire circuit need to be translated into a behavioral description, which is usually a hardware description language such as VHDL or Verilog. Until this process has been achieved, the next step is to perform synthesis in order to turn the behavioral description into a design implementation as a layout design, and then the digital files are passed to the foundry for fabrication. Alongside outsourcing the production of the integrated circuits, companies are also using third-party intellectual property (IP) cores in their applications, so the probability of hardware being compromised is too high. Hardware Trojan insertion, IP cloning, and IP full control are the most dangerous attacks to be aware of.

Concerning Foundry level, a supposed golden prototype of the IC clear of any suspicious alteration is required to provide excellent verification. Testing methods require destructive de-packaging and reverse engineering of the IC; after this process, each layer of the IC is compared to its counterpart in the golden prototype; however, the stealthy nature of the hardware Trojan would ensure that these testing routines are futile.

2.2. The different hardware Trojan taxonomies

The general taxonomy related to the hardware Trojan consists of the physical representation, the behavioral phase or trigger, and the action phase in which the Trojan will execute its payload.

For the physical representation, there are several characteristics to be considered. First is the type of hardware Trojan that can be either functional or parametric: adding extra components such as logic gates to the original design or the deletion of components in order to cause a malicious function or damage would place it under the functional type. The attacker can make a few modifications, such as thinning wires or changing transistor values and flip-flops to be weak, that will be referred to as parametric. Besides, the size of the hardware Trojan is an important factor to be considered since the attacker would use only a few components required to execute their payload in accordance with the functions to be provided, as they will occupy only a small part of the layout of the IC, which will make the hardware Trojan stealthier.

The activation of the hardware Trojan is also to be considered; it can be triggered both internally and externally. An externally triggered hardware Trojan will consist of malicious logic within the IC that utilizes an external sensor such as a radio antenna. For the internally triggered hardware Trojan, the attacker will set a set of conditions that will cause it to activate, such as a countdown or countup logic. Hardware Trojans are one of the two categories: implicit, which will not alter the circuitry of the IC; they will perform their malicious function side by side with the supposed normal function of the board. In contrast, explicit hardware Trojans will change the function of the circuitry on activation, such as signal alteration or even the leak of sensitive data.

2.3. Mathematical background of the cryptosystem

Petri nets are graphical and mathematical framework used in many applications including the modeling and analysis of discrete-event systems, concurrent systems, fault tolerant systems and many others. As a mathematical formalism, it is possible to set up state equations, algebric equations, and other mathematical models governing the behavior of systems.

Petri net is a weighted directed graphs, it consists two types of nodes, the place (represented by circle) and transition (represented by bar), place and transition connected by directed arcs either from place to transition or transition to place, each arc are labeled with weight. A marking of net is labeled by M, the number of tokens in place p is denoted by M(p), the transition t is enabled when each input place has number of tokens greater of equal to the weight of the the input arc of that transition which is labeled w. Following a certain path on the net requires the transition to be enabled; that would delete a token from each input place p and add a token to each output place according to the weight of the associative arc. In our case, we have chosen the weight of the whole graph to be 1 in order to facilitate the task of implementing the algorithm in its hardware form.

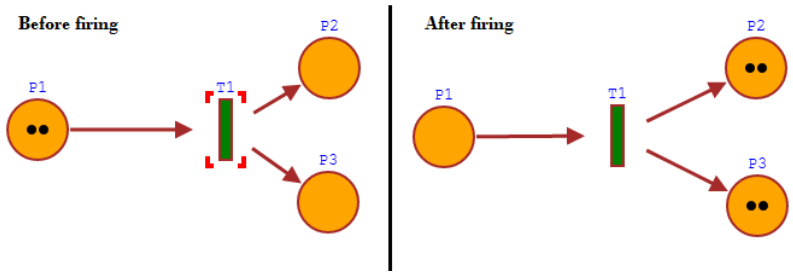

Figure 1. Firing of transition.

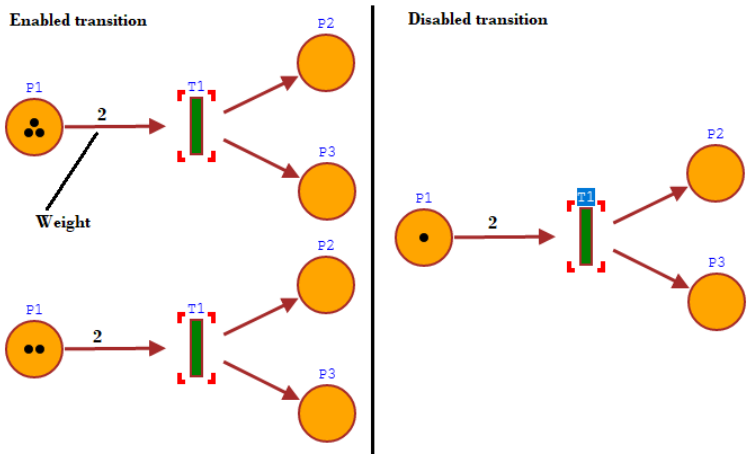

Figure 2. Enabled VS Disabled transition

The formal representation of Petri net is as follows:

PN=(P,T,F,W,M_0), P={p_1,p_2,p_3,…,p_n} is a finite set of places, T={t_1,t_2,t_3,…,t_n} is a finite set of transitions, $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs, W: F→{1,2,3,…,n} is a weight function, M_0: P→{0,1,2,3,…,n} is the initial marking. The structure N = (P,T,F,W) is the petri net structure without initial marking, and when there is an initial marking it would be denoted by (N,M_0).

Table 1. The formal definition of Petri net

A Petri net is a 5-tuple PN=< P,T,F,W,M_0> where:
P={p1,p2,p3,…,pn} is a finite set of places,
T={t1,t2,t3,…,tn} is a finite set of transitions,
$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs,
W: F→{1,2,3,…,n} is a weight function,
M0: P→{0,1,2,3,…,n} is the initial marking,
$P \cap T = \emptyset$ *and* $T \cap P = \emptyset$.

3. RELATED WORKS

This section is devoted to citing some approaches that used HSMs to secure system services and critical data using encryption; others used Petri Net as a tool to generate long, complex secret keys to be used in encryption and decryption operations.

In [4], use HSM to secure the acquisition of equipment status information; it uses a security controller attached to a customer device that generates secret keys and random initial values in order to encrypt the inquired information. However, the encryption and authentication processes are performed on an externally connected board running the Linux operating system, using a library and private keys provided by the HSM. That would compromise the connection between the HSM and the board as well as potential operating system vulnerabilities.

In [5], presented the research work of the EU AFarCloud project. It introduces the important LoRaWAN data communication technology for the transmission of sensor data used in agriculture. A HSM called Zymkey 4i, produced by Zymbit Corporation, has been integrated into the Raspberry Pi. In this approach, the key for encryption of the file system will be stored directly inside the module, which still poses a potential threat if physical access to the HSM is gained.

In [6], biomedical data on the server has been encrypted using the AES algorithm, and an IBM 4764 PCI-X cryptographic coprocessor has been used to store symmetric keys used in the encryption process. In this approach, secret keys have been stored directly in ROM, which gives a malicious user the possibility to reveal such keys when gaining physical access to the coprocessor.

In [7], use HSM to secure an online repository for public key infrastructures, to store more than 800 2048-bit RSA key pairs on the secure storage of an IBM 4758 cryptographic coprocessor. The coprocessor takes care of the entire RSA operation through the library on an external processor running the Linux operating system. Despite the flexibility of performing the RSA operation in hardware, the connection between the device and the external processor is still vulnerable.

In [8], a method to build a private key cryptosystem has been presented. This method is based on the dynamic priority colored petri net to produce complex long key sequence. It is divided into two parts: the first one produces random key based on four colored petri nets that have the same design and one ordinary petri net. The second part, the plaintext, has been converted to gray code and permutated in such a way to increase the complexity of the generated secret key. This approach still needs more work in order to be concrete in hardware form, since there are multiple confusions in colored petri nets and in each state different values transfer from one transition to another, which would increase the complexity of implementing such model in IC form.

In [9], in this paper a secured password has been provided using colored petri nets. Assigning time for each alphabet and character, the hacker has to decode the character and the assigned time of each character since the transitions are assigned with interval time and the user has to enter the password before this time elapses. This approach still needs more processes to be launched, and it is time-consuming, which is an important factor that must be considered in embedded systems applications.

4. METHODOLOGY

This section presents hardware and software countermeasures to enforce security within SOCs and defeat possible attacks that threaten their security [10].Our work consists of making secure SOC [11], and encrypt/decrypt engine that use Petri net for algorithm modulation.

The secure SOC block ensures component integrity, thus providing physical protection for secret keys and sensitive firmware. The use of buffers inside the internal RAM will prevent secret keys and intermediate values of cryptographic operations from being revealed, and that is performed by the use of a secure boot loader to ensure booting from a genuine OS or firmware with the right system privileges. This OS will configure the memory management unit (MMU) to permit access to buffers in the internal RAM to only secure system processes with the right system privileges. The secure ROM used is pre-programmed so that the user has the total freedom to initialize it with the master key, which will be used to generate the secret key during the execution time, and that would complicate the task for an unauthorized user to get the key since it is not directly stored in the ROM. In the internal RAM, buffers are allocated to ensure the secrecy of secret keys and intermediate values of cryptographic operations; for that, the OS during boot up configures the MMU with the right access privileges to the right system processes allowed to commit operations on buffers.

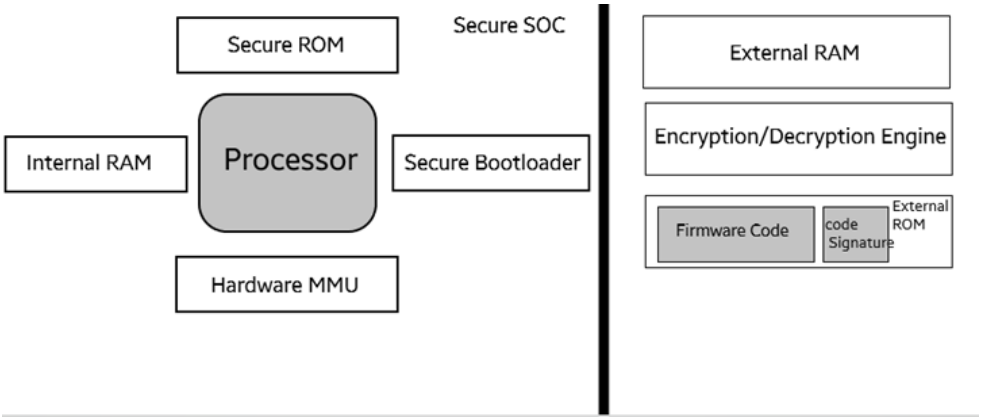

Figure 3. Chip components are protected inside secure SOC

On start-up, the secure boot loader checks the firmware to make sure that we boot from the genuine version since it contains critical code that handles the configuration to specify access permissions to the internal RAM. To perform this action, the secure boot loader uses the code signature module of the firmware code and the code verification public key.

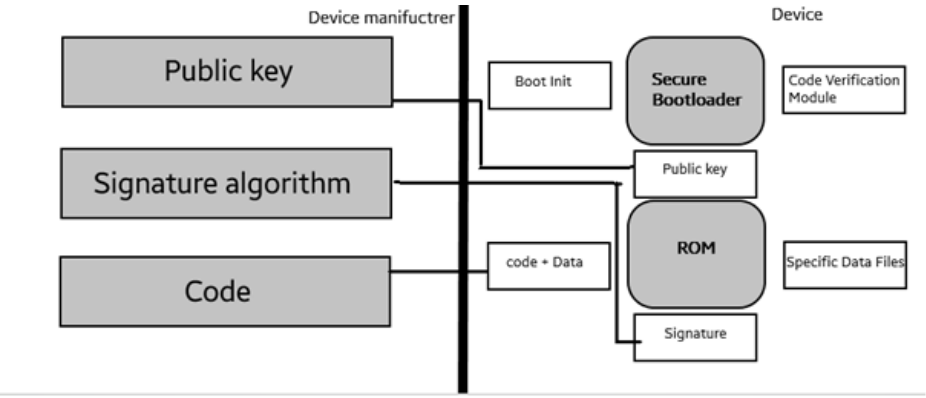

Figure 4. The code signing operation.

Concerning the Encrypt/decrypt engine that is used to encrypt/decrypt sensitive data sent to/from the device. Our cryptosystem is based on Petri net to generate a complex random number, which will be used as a private key by our engine. For that, we use the marking of the Petri net after firing certain transitions known only to us as SOC manufacturers. Thus, we ensure that only encrypted data flows over the bus to and from the engine, which makes the bus monitoring operation by an unauthorized user useless. Besides, our proposed Petri net consists of six transitions and six places, and it is subject to modification depending on user needs or security enforcement policy.

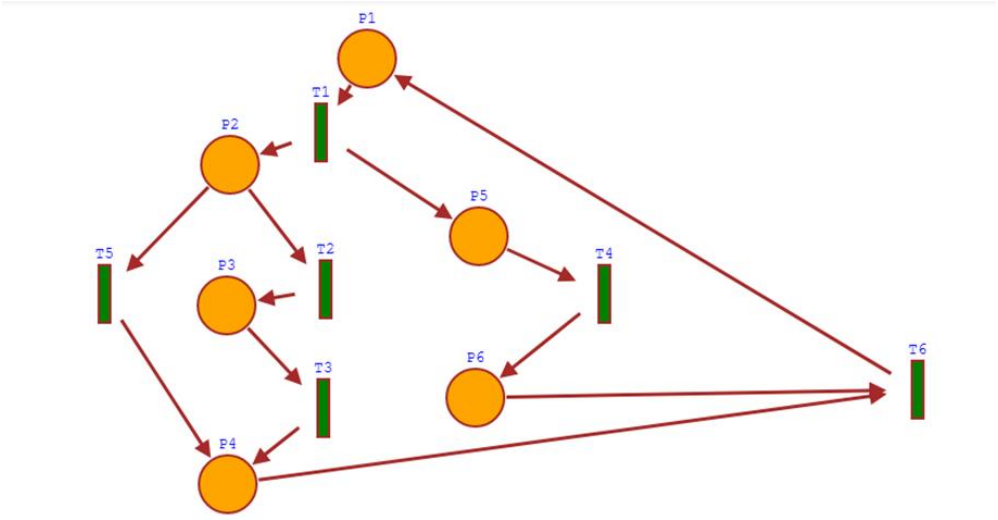

Figure 5. The suggested Petri net used to generate a private key.

The algorithm used to generate the private key for Encrypt/Decrypt operations is as follows:

Algorithm to generate the private key

Input: The master key
Output: The private key

BEGIN

turn the master key into its ASCII code then convert it to binary code and divide it into a sequence of 8 bits each, we get this sequence S= {S0, S1, S2,……}

Initialize the marking of the Petri net with S: place P1 with S0, place P2 with S1 …, while the remaining places are set to zero.

Fire the enabled transitions N time.

Finally take the resulting marking of the Petri net as a private key.

END

The algorithm used to Encrypt/Decrypt sensitive data is as follows:

Algorithm for the data encryption

Input: data
Output: Encrypted data

BEGIN

Turn data into its ASCII code then convert it to binary code.
Make the XOR operation of the result with the previously generated private key.
Divide the resulting sequence into 8 bits each and get its corresponding ASCII code which will be the cypher text

END

For decryption operations, we substitute only the plain data with the encrypted data, and we perform the same steps to get back the plain data. Finally, all this work has been wired into hardware, and to demonstrate the circuit in action, LOGISIM is used for simulation purposes.

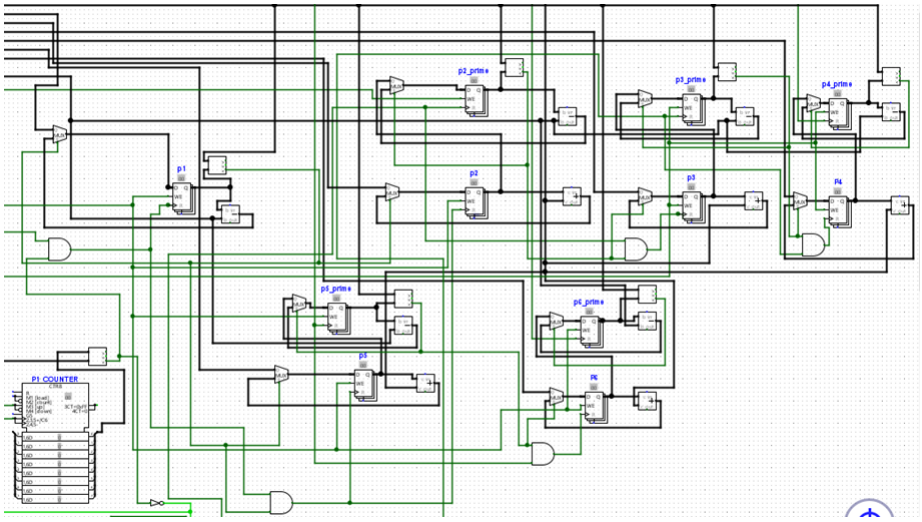

Figure 6. Hardware wired Petri net that generates a private key.

The user will feed the inputs of our circuit during SOC setup so that he can initialize the secure ROM with the desired master key value and the number of times N that will be used in the counter to fire the circuit transitions N times. After execution completes, registers value is the resulting private key. Furthermore, we used 8-bit registers for simulation purposes only; otherwise, the user can use larger ones depending on its security policy and the system storage requirements.

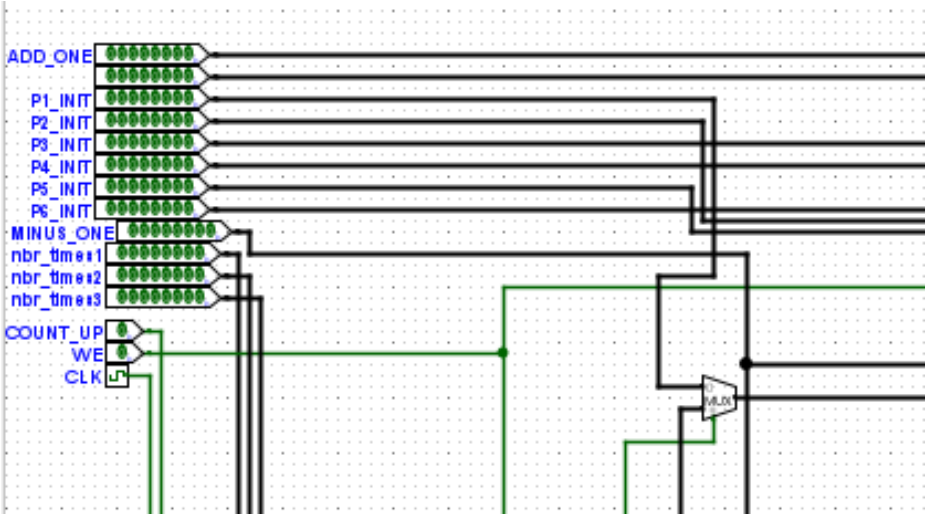

Figure 7. System inputs are to be fed during SOC setup.

5. RESULTS AND DISCUSSION

To demonstrate the results, strengths, and weaknesses of our secure SOC, we present the following study case: We use the word "alpha" as a master key to generate the private key and the plain text "hit the target" as sensitive data to be sent to an advanced army command post.

After using the master key "alpha" as an input to the algorithm that generates the private key we get the following 64-bit key with N=10 (we can use a larger value for N depending on security enforcement and system execution time) :

PK = 0000000000000000010000000101111101110000011101100110000100000001

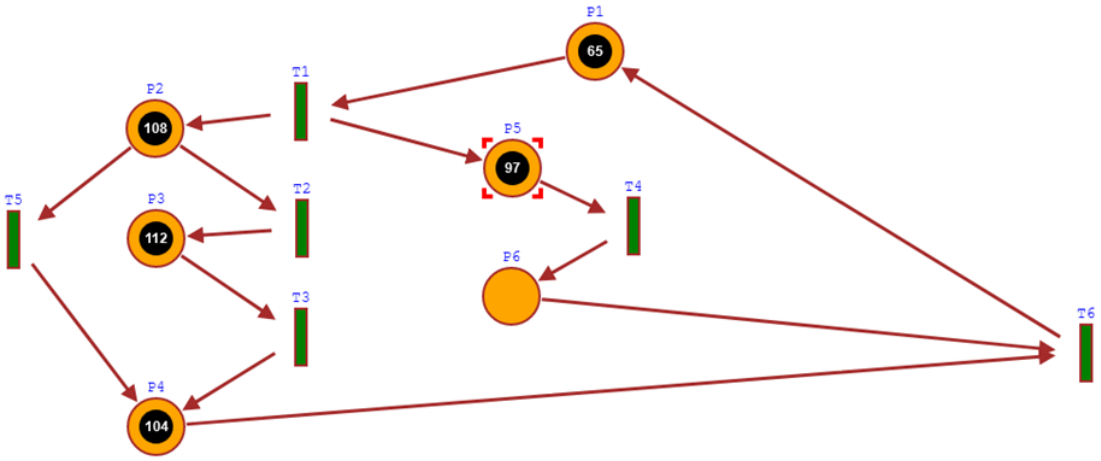

Figure 8. Petri net initialized with the master key value

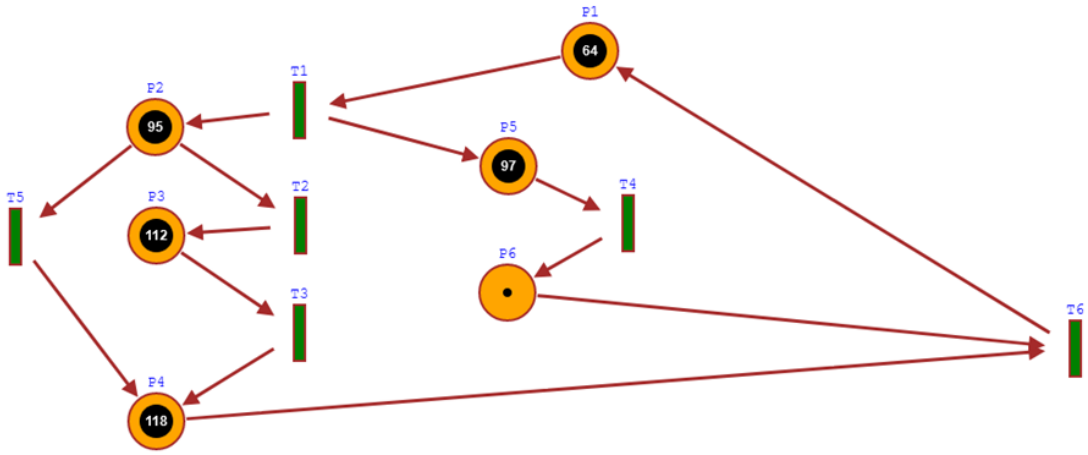

Figure 9. The Petri net after private key generation

After turning the message to be sent (hit the target) into its ASCII code and then to binary code we get the following sequence:

Message:
01101000011010010111010001110100011010000110010101110100011000010111001001100111011001010111010 0

After XOR operation between the two previous results and getting the corresponding ASCII code, we get the following cypher text:
hitthe4> STX (DC1) (EOT) u

The system performance has been tested depending on the number of firing Petri net transitions (N) and the length of the master key.

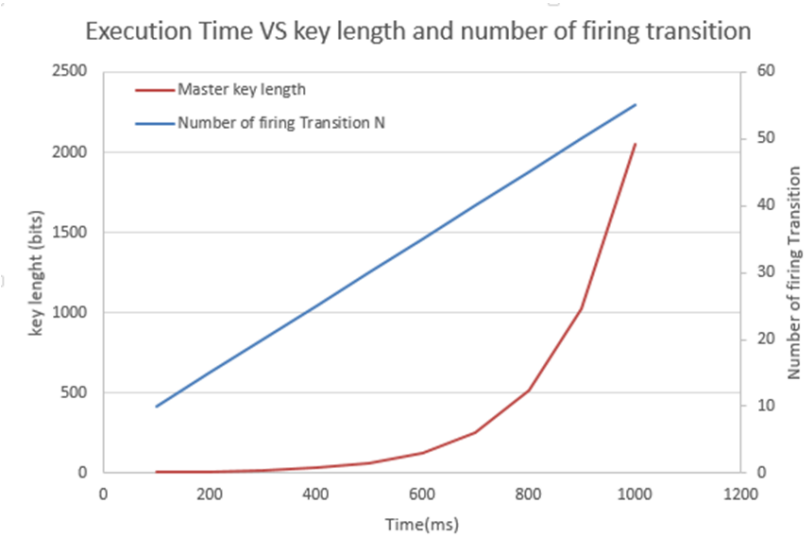

Figure 10. Execution Time VS key length and number of firing transition.

The system shows its power since we use a very powerful encrypt/decrypt system based on Petri net only known to SOC manufacturers; thus, it would be a very complex operation for an attacker to break the encryption. Furthermore, the secret keys are not directly stored on the chip so that we can prevent their disclosure; rather, they are generated during execution time. The SOC firmware is well protected since we use code signing to ensure that we boot from the genuine one. The only drawbacks are due mainly to system performance, such as execution time since we use a cyclic encryption algorithm that requires a few rounds or more to generate a complex encryption key, and key length, which requires more storage resources that are very critical in embedded systems.

6. CONCLUSION AND FUTURE WORK

In this paper, we discussed hardware and software countermeasures to enforce security within SOCs and embedded system security in general. We created an HSM with a secure SOC architecture to ensure the circuit's components' security, data integrity, encryption and decryption engine within the HSM ensures safety. We used the Petri net modulation technique for the engine's encrypt/decrypt algorithm.

Our method is quite secure in terms of protecting secret keys used in the encryption and decryption operations since we do not store these kinds of keys directly in HSM's storage; instead, we generate them during execution time using our engine, which uses the Petri net algorithm as a secret key generator. For future approaches, we will enhance our system performance and security; for that, we aim to adapt our technique with the SmartFusion2 SOC [12] and to use colored Petri nets for a more secure cryptosystem.

REFERENCES

[1] Y. Zhang, "A Systematic Security Design Approach for Heterogeneous Embedded Systems," 2021 IEEE 10th Global Conference on Consumer Electronics (GCCE), pp. 500-502, 2021.
[2] J. Rajendran, A. M. Dhandayuthapany, V. Vedula and R. Karri, "Formal Security Verification of Third Party Intellectual Property Cores for Information Leakage," 2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), pp. 547-552, 2016.
[3] B. Guechi, & M. Redjimi, "Hardware Trojan Detection in Heterogeneous Systems on Chip". In Innovations in Smart Cities Applications Volume 4: The Proceedings of the 5th International Conference on Smart City Applications (pp. 1105-1116). Springer International Publishing, 2021.
[4] C. Lesjak, H. Bock, D. Hein and M. Maritsch, "Hardware-secured and transparent multi-stakeholder data exchange for industrial IoT," 2016 IEEE 14th International Conference on Industrial Informatics (INDIN), pp. 706-713, 2016.
[5] R. Kloibhofer, E. kristen and L. Davoli "LoRaWAN with HSM as a security improvement for agriculture applications." In: International Conference on Computer Safety, Reliability, and Security. Springer, Cham, pp. 176-188, 2020.
[6] M. Canim, M. Kantarcioglu and B. Malin, "Secure Management of Biomedical Data With Cryptographic Hardware," in IEEE Transactions on Information Technology in Biomedicine, vol. 16, no.1,pp. 166-175,2012.
[7] M. Lorch, J. Basney and D. Kafura, "A hardware-secured credential repository for Grid PKIs," IEEE International Symposium on Cluster Computing and the Grid, pp. 640-647, 2004.

[8] R. Abdulwahid Albeer, H. A. Lafta, and H. Karim, "Key Stream Cipher Based on Coloured Petri Nets" in The 9th International Conference on Applied Science and Technology (ICAST 2021),pp. 1-8,2022.
[9] M.I. Mary Metilda, D. Lalitha, S. Vaithyasubramanian,"Password generation using array generating interval timed colored Petri net (AGITCPN) for effective security" Theoretical Computer Science,Vol. 929,pp. 114-123,2022.
[10] M. Malenko, L. B. Ribeiro, and M. Baunach. "Improving security and maintainability in modular embedded systems with hardware support: work-in-progress". In Proceedings of the 2021 International Conference on Hardware/Software Codesign and System Synthesis (CODES/ISSS '21). Association for Computing Machinery, New York, NY, USA, pp. 35–36, 2021.
[11] J. Haj-Yahya, M. M. Wong, V. Pudi, S. Bhasin and A. Chattopadhyay, "Lightweight Secure-Boot Architecture for RISC-V System-on-Chip," 20th International Symposium on Quality Electronic Design (ISQED), pp. 216-223, 2019.
[12] A. Rodrigues, J. C. Resende and R. chaves. "SmartFusion2 SoC as a Security Module for the IoT world". In 19th ACM International Conference on Computing Frontiers (CF'22), Torino, Italy, pp.1-9, 2022.
[13] N. Zupan, P. Kasinathan, J. Cuellar, and M. Sauer, "Secure Smart Contract Generation Based on Petri Nets," in Blockchain Technology for Industry 4.0,Springer,2020.

BIOGRAPHY OF AUTHORS

Billel Guechi received Masters degree in Embedded Systems from Badji Mokhtar University, Algeria in 2018. Currently Phd student in embedded systems and Software Developer. Research interests include: power management in embedded systems, security in embedded systems, embedded system design…

Mohammed Redjimi is full professor of computer science at Université 20 Août 1955, Skikda – Algeria. He received his PhD degree from the Université des sciences et Techniques de Lille1 – France in 1984. He received the Habilitation Universitaire degree from The University Badji Mokhtar of Annaba – Algeria In 2007. He was Head of computer science department and Dean of Engineering and Sciences Faculty at Université 20 Août 1955, Skikda – Algeria. He is reviewer in several international journals and conferences. He is currently the head of the team: modeling and simulation of complex systems at the "Computer science and Communication" Laboratoiry (LICUS) and director of this research laboratory at Université 20 Août 1955, Skikda – Algeria. His present research domains include modelling and simulation (SMA concepts and platforms and DEVS formalisms) and Wireless Sensors Networks (WSNs).