Enhanced Device-Specific Encryption for IoT: Leveraging Microcontroller UIDs and Dedicated Cryptographic Hardware

EDWAR JACINTO, FREDY MARTINEZ, FERNANDO MARTINEZ
Facultad Tecnológica
Universidad Distrital Francisco José de Caldas
Carrera 7 No. 40b-53, Bogotá
COLOMBIA

Abstract: - In this research, we introduce an advanced encryption system that aligns with global cryptographic standards, emphasizing device identification and adherence to the PKCS#5 standard. Instead of relying on pseudo-random numbers, our innovative key expansion approach capitalizes on the microcontroller's UID, merged with the session key, and subjected to a repetitive Digest algorithm, thus achieving a dimension apt for asymmetric encryption protocols. Leveraging the dedicated hardware support for the SHA-256 algorithm, we compute a distinctive digest through varying iteration counts, gauging computational prowess. We employ AES128 for data encryption, given its ubiquity and its hardware inclusion in the NXP FDRM-K82F device. This device boasts a Cryptographic Acceleration Unit (CAU), optimizing processing durations and memory consumption, paving the way for autonomous cryptographic systems with viable throughput rates tailored for IoT scenarios. The microcontroller's specialized LP Trusted Cryptography (LTC) hardware champions diverse algorithms, enriched with atomic directives. We integrate Physical Unclonable Functions (PUFs) into our design, harnessing inherent manufacturing disparities to spawn unique, hard-to-replicate keys. The key expansion is rooted in the PKI infrastructure, sourcing a distinct number per device from the FRDM-K82F's UID and culminating in a 256-bit Digest, employed as the AES-CBC key and Initialization Vector (I.V.). Our empirical assessment underscores the superior efficiency of the CAU against counterparts lacking such modules, showcasing remarkable boosts in performance and diminished encryption/decryption intervals. Consequently, our pioneering approach emerges as a prime candidate for fortifying IoT applications.

Key-Words: - AES encryption, cryptographic hardware, Internet of Things (IoT), Physical Unclonable Function (PUF), Unique Identification (UID), secure communication

Received: May 11, 2023. Revised: April 14, 2024. Accepted: May 13, 2024. Published: June 17, 2024.

1 Introduction

The ubiquitous boom of the Internet of Things (IoT) has exponentially expanded the landscape of interconnected devices, [1], catapulting the urgency for fortified and impenetrable communication channels, [2]. As IoT penetrates homes and businesses, ensuring essential levels of security in line with contemporary standards becomes something non-negotiable, [3]. This is accentuated by the spread of sensitive data, [4], and the possibility of generating vulnerabilities at multiple levels, [5]. More importantly, establishing an irrefutable identity of the data authors, making cloning and falsification of the data unlikely, is emerging as an essential requirement, [6].

The challenge of implementing encryption in IoT applications lies in balancing three critical aspects: performance, energy efficiency, and robust security, [7]. While established encryption methods provide strong security, [8], they often struggle to meet the processing and energy constraints of IoT devices, [9]. This situation highlights the need for innovative cryptographic solutions specifically designed for the unique requirements of IoT, focusing on cost-effectiveness, widespread applicability, and strong security, [10].

To tackle the complex issues of security, numerous cryptographic strategies have emerged, [11]. Among these, the *salt complement* method, [12], which combines a pseudo-random number with a user key, has become popular. However, this method does not inherently provide a clear way to verify the origin, [13], which is crucial for confirming the legitimacy of the sender.

Revolutionizing embedded systems, manufacturers have augmented microcontroller architectures, embedding specialized physical modules for discrete functionalities, [14], [15]. Notably, cryptographic-centric modules offer direct, CPU-independent operations, encompassing block ciphering, abstract-type unique number generation, and unique identification codes, [16]. This metamorphosis unlocks sophisticated cryptographic capacities within cost-effective microcontrollers, aligning with benchmarked standards like those set by NIST and delivering performance congruent with prevalent application bandwidths, [17].

In light of this evolving landscape, and building upon prior works examining cipher implementations using salt complements in IoT, [18], [19], our research introduces an innovative approach to sender authentication. We propose the substitution of the traditional salt complement with a microcontroller's intrinsic Unique Identification number (UID), [20]. Derived from uncontrolled manufacturing variables, [21], this UID resonates with the ethos of Physical Unclonable Function (PUF) technology, [22], [23]. Our exploration is grounded in two microcontrollers, the FDRM K82F and the STM32-F401RE, culminating in the generation of secure, verifiable keys optimal for encryption, and benchmarking their performances. Further, we harness the AES standard block cipher, [24], [25], leveraging dedicated encryption hardware, all orchestrated by an energy-efficient alternate processor ensuring autonomous operation.

The subsequent sections delineate our research journey: Section 2 elucidates our methodological framework, spotlighting three pivotal components. Section 3 chronicles our solution's implementation, demystifying the underpinning source code and emphasizing the synergy of dedicated cryptographic hardware with intricate low-level programming. Section 4 presents a comparative analysis between the cryptographic-equipped FDRM K82F and the more rudimentary STM32-F401RE. We conclude in Section 5, synthesizing our research insights and charting potential trajectories for future exploration.

2 Methodology

In this work, we advocate for a fortified encryption system, anchoring its foundation on an innovative key generation method. This methodology aligns rigorously with standards delineated by premier international cryptographic entities. The linchpin of our cryptographic architecture is its meticulous key expansion procedure, crafted to unmistakably identify the device orchestrating the encryption. This intricate design is underscored by adherence to the PKCS#5 standard, eschewing conventional pseudo-random numbers in favor of the microcontroller's UID. This UID, when seamlessly amalgamated with the session key, lays the groundwork for subsequent iterative applications of a Digest algorithm. This meticulous process culminates in a key, congruent in width to its asymmetric encryption algorithm counterpart.

To further bolster our system's robustness, we engaged the SHA-256 algorithm to compute a distinctive digest number. This algorithm underwent multiple iterations, serving as a litmus test to ascertain the requisite computational prowess, ensuring that our methodology not only embodies precision but also operational efficiency.

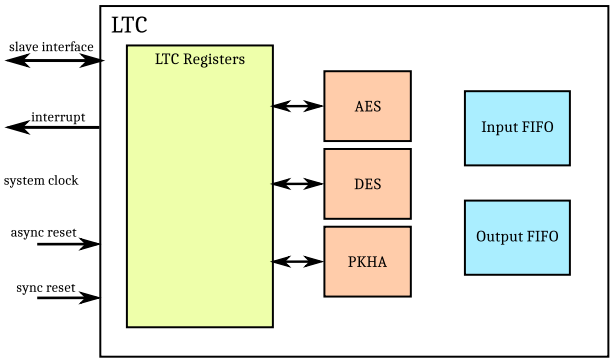

Figure 1: LTC architecture, [26]

After ensuring that the user session key generates a value with the characteristics required for the process, the information was encrypted using AES128 since it is the most widely used encryption algorithm today. For this reason, it has been included as a hardware block within the architecture of the device called Cryptographic Acceleration Unit (CAU) FDRM-K82F of the company NXP, reducing processing times and the amount of memory used; all this makes that, over time, the embedded cryptographic applications are stand-alone and have usable throughput rates in real applications such as IoT.

The microcontroller has dedicated memory-mapped hardware, shares a standard set of registers, and has its atomic instruction set to perform the cryptographic tasks required in this application. This custom-made hardware by the manufacturer NXP is called LP Trusted cryptography (LTC). A general schematic of the architecture that makes up LTC and the algorithms it supports directly by dedicated hardware is shown in Fig. 1, [26]:

2.1 PUF Function, [27]

Physical Unclonable Functions (PUF) are physical systems of response to stimuli that are easy to measure but not easy to clone, unlike other random key generation methods. They are generated from uncontrollable variations in the manufacturing process, in this case, the number of doped atoms due to the intrinsic process of semiconductor manufacturing, transistor print size, and changes in the physical structure of the material due to the natural behavior of the atoms, et cetera, such randomness, resulting from unpredictable physical phenomena, are complicated to attack. A series of literature talks about this characteristic. Depending on the manufacturer or the context in which this data is being discussed, this technology can be referred to as UID, UUID, or Device Identification (DIE).

Features of PUF functions

PUFs are considered electronic bio-metric measurement systems due to the way they are generated. Since they are based on non-controllable physical parameters, just like, for example, fingerprints, these systems are characterized by the following parameters, [27]:

- Unique: It is based on an analogous real-world physical system.
- Easy to evaluate: using the specific physical system chosen, it is possible to check it easily.
- Randomness: because its generation is based on real-world physical systems, its outcome is considered thoroughly random.
- Non-predictable: as a truly random system, there is no easy way to attempt to predict its value based on variables common to attack pseudo-random systems.
- Unclonable: as its generation is based on characteristics intrinsic to the manufacturing process, and because no two hardware units are identical even if they are the same, it is not possible to clone it.

2.2 PKCS#5 Key Expansion Process: Variation with PUF

A key expansion process is performed using the credentials dictated by the PKI public key infrastructure. However, in this case, instead of performing the process of mixing the user session key with a pseudo-random number typically called salt, it was sought that the process of generating this key in the expansion process will be performed with a unique number for each device. For this, it was chosen to read a physical address within the internal memory structure of the FRDM-K82F device, which corresponds to the UID number housed in the memory address 0x1FFF7A10 and has a length of 12 bytes. This number is concatenated with the session key entered by the user, and calculation of the unique digest number is performed using the SHA256 algorithm, which is supported by the LTC hardware managed by a co-processor for such tasks of ColdFire(R) architecture, which is one of the architectures handled by the manufacturer Motorola, now FreeScale-NXP.

2.3 Data encryption: Block Cipher AES

Over the past decade, the paradigm governing embedded applications in microcontroller-type devices has undergone a profound transformation. No longer confined to simplistic data interpretations, contemporary systems treat readings-whether emanating from sensors, actuators, or specialized hardware tailored for diverse applications-as intricate character strings. Against this backdrop, our research entailed comprehensive experimentation with plaintext strings of varying lengths. In each instance, meticulous care was exercised to adhere to the memory map architecture delineated by the LTC. Furthermore, to ensure precision and consistency in our data manipulation, we leveraged atomic instructions, a potent toolset provided within the manufacturer's Integrated Development Environment, MCUXpresso.

For such programming, it is necessary to provide the AES function with the following parameters: the session key after the key expansion process under the PKI standard and the mixing type of the key with the initial round. Such modes of operation of AES are shown in previous works [28]; in this case, it was chosen to use the AES-CBC mode since the internal architecture of the device supports it. It is of vital importance to emphasize the correct use of the key expansion since two numbers are required to perform the encryption process in the different AES modes, one is the expanded session key, and the other is the Initialization Vector (IV) of the same length, in this case, the key was expanded to 256 bits, and 128 bits were taken for the expanded session key and 128 bits for the IV.

This application seeks to leave a trace of the device that performed the information encryption process; therefore, the pseudo-random number is replaced in the PKCS #5 scheme by the PUF of the device, but it should be clarified that to complete the process, the number resulting from the key expansion process must be stored locally or in the cloud along with the unique number of the UID device, all this must be done securely, using the available resources and following the guidelines of international cryptographic entities.

2.4 Development: Block Diagram

Once the element to perform the complete encryption process is clear, it is necessary to understand the complete flow of the application; Fig. 2 is shown the general block diagram of the application. This diagram shows all the information flow of the key expansion on the left side and the block cipher part on the right side using AES-CBC.

The first process within the algorithm is to capture the unique identification number of the device; this is done by reading a specific memory location, then the session key must be obtained using the serial port of the device as a method of entry for the user, then the digested key is concatenated with the UID; for this work, this methodology is used, but in future work is expected to improve the Human Interface Device (HID) or method of entry in this way to make interactive applications, or that have the possibility of

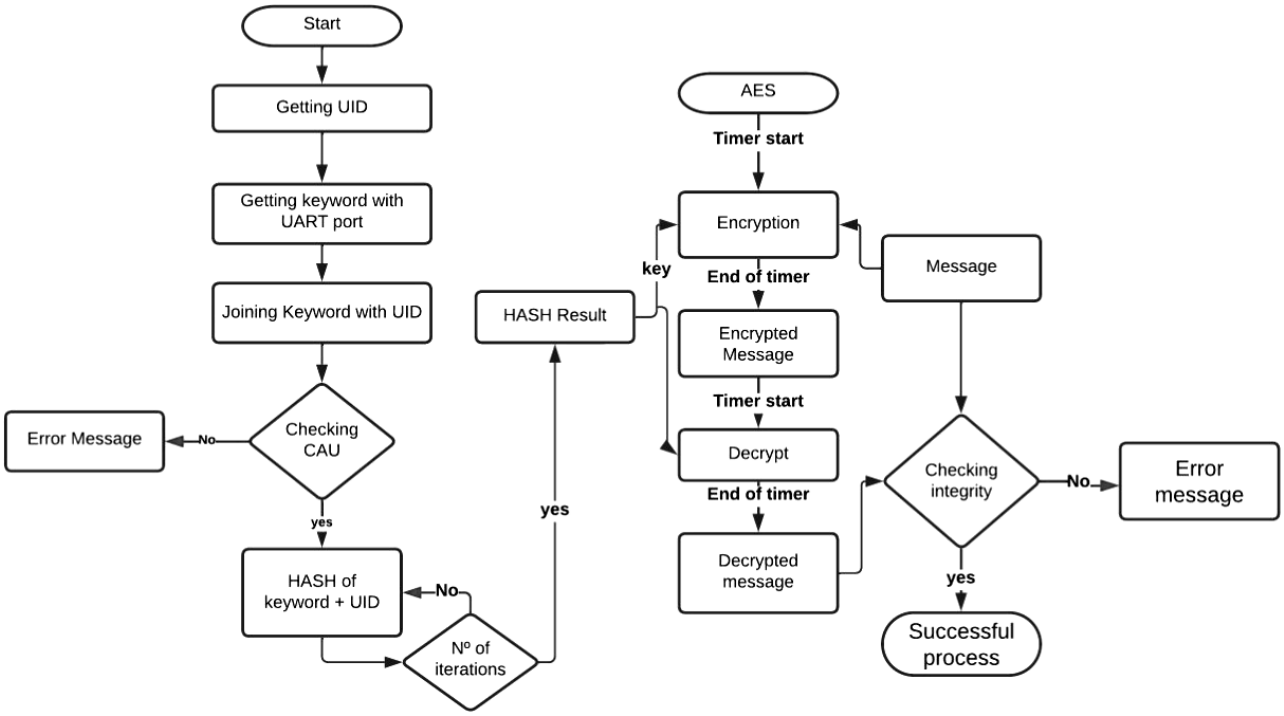

Figure 2: General block diagram of the application

remote access. Subsequently, the CAU, through the block dedicated to calculating unique numbers or digest, uses the SHA256 algorithm iteratively in order to give different levels of security, for which a series of tests were performed verifying the amount of time required to perform the above process, at the end the digest of 256 bits is obtained which consists of a number used as Key and I.V. for the AES-CBC algorithm, the above shown on the left side of the diagram.

The block diagram's right segment elucidates the intricate mechanics underpinning the encryption and decryption processes. Central to our empirical evaluation is the deployment of a microcontroller's intrinsic physical timer as a pivotal test parameter. This meticulous approach facilitates a granular assessment of the processing durations expended by the Cryptographic Acceleration Unit (CAU) across diverse information magnitudes, thereby equipping us with empirical metrics to gauge throughput for varying scenarios. To ensure data fidelity, each encryption or decryption operation is accompanied by a rigorous information integrity assessment. It's imperative to underscore that the encryption mechanism commences with a foundational key paired with an Initialization Vector (I.V.). However, the decryption paradigm mandates the nth I.V., given the encryption's inherent intricacies. Post-decryption, it is crucial that the deciphered data reverberate in its original plaintext form, conjoined with the inaugural I.V. birthed from the key expansion trajectory.

3 Implementation

The prototype was assembled using the NXP FDRM-K82F microcontroller, chosen for its integrated Cryptographic Acceleration Unit (CAU) and LP Trusted Cryptography (LTC) hardware, which significantly reduces processing times and memory usage. The selection criteria for this microcontroller were based on its robustness, accuracy, energy efficiency, cost, and availability. By utilizing the UID of the microcontroller, the proposed encryption system eliminates the need for pseudo-random numbers in the key generation process, resulting in a more secure and efficient solution.

For this case, an embedded software application was made in a microcontroller-type Stand-Alone device. This application is written in ANSI C language in the FreeScale-NXP MCUXpresso IDE. It is important to emphasize that this implementation must be done in a mandatory way in a Bare-metal programming mode since native and atomic functions of this IDE must be used to access the hardware resources of the CAU. On the other hand, previous work has been done using higher level IDE's, using Rtos and with C++ programming, which can be used as a reference to compare the performance of this type of solution using only the power of a CPU that does not have hardware blocks dedicated to cryptography. A

```
// main process
main(void)
    CPUClockCycleCheck();
    Peripherals init();
    ltc();

// LTC Process start
static void ltc(void);
    uid = UIDExtract();
    clave = KeywordRead();
    CAUinit();
    hash(uid+keyword);

// HASH process start
static void hash(uid+clave);
    n = rounds;
    key = hash(uid+keyword,n);
    AES(key);

// Encrypt and Decrypt
static void AES(key);
    messagge = UARTmessage;
    StartTime;
    encryptedMessage= Encrypt(key,UARTmessage);
    EndTime;
    t1 = EndTime - StartTime;
    StartTime;
    DecryptMessage = Decrypt(key,encryptMessage);
    EndTime;
    t2=EndTime-StartTime;
```

Figure 3: Pseudo-code

pseudo-code of the embedded software application programmed for the FRDM-K82F in bare-metal is shown in Fig. 3.

Like all applications for this device, a hardware verification must be performed, starting with the system clock, it is prescaling, and internal or external clock sources, depending on the application and the power consumption required. Then, all necessary peripherals are configured, such as communication ports for reading sensors or external modules. Many programmers make a series of APIs tailored for such configurations. In this case, a function that uses the ARM CMSIS layer facilitates application compatibility tasks.

A statement is also used to turn on the CAU unit using LTC. This hardware module is instantiated in a static function with no input or output parameters, thus generating the memory space and initializing the dedicated hardware.

The first parameter is the UID obtained by reading a position in the device's FLASH memory in a specific zone, which is mixed with the key entered by the user. Having both the UID and the session key, we perform the necessary iterations to expand the

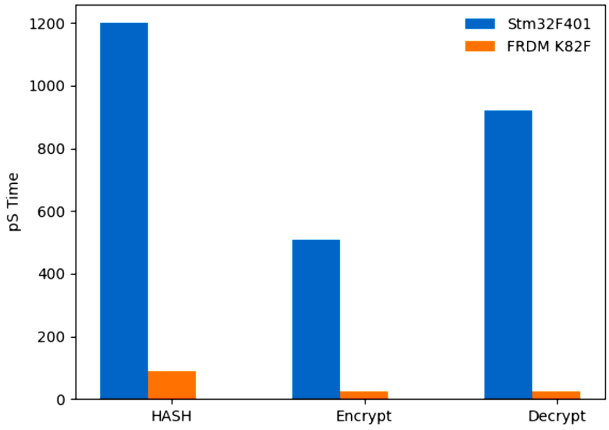

Figure 4: Performance comparison

key. Without first initializing the required hardware, the number of iterations within the key expansion process is dynamic depending on the degree of security and the time available for this task.

From the previous process, the AES block cipher requires a key of the same size, and the I.V. is returned to encrypt the plain text to be sent securely. Finally, some functions perform the process of encryption and decryption of the information, for which the UART makes transmission of the information of the microcontroller, and the times for each process of the information are measured, all this to perform the measurements of the metrics proposed and required.

4 Findings and results

To validate the efficiency of the system with hardware dedicated to cryptography, which for this manufacturer is called CAU, compared to a device with similar performance but without this type of dedicated modules, a comparison of the time of each of the cryptographic processes was made, which can be seen in Fig. 4, where the high performance of the microcontroller with cryptographic tools in dedicated hardware is clearly evidenced.

In the key expansion process, there is a significant difference between the two systems because an iterative process generates an exponentially increasing variation, going from a few nanoseconds to 120 microseconds.

As for the encryption process, tests were performed with character strings entered through the serial port more minor than the size of the cipher; in other words, AES256 was used, and the test strings were smaller than this size, so the algorithm automatically completed the information to the size of the block by filling with zeros to the right.

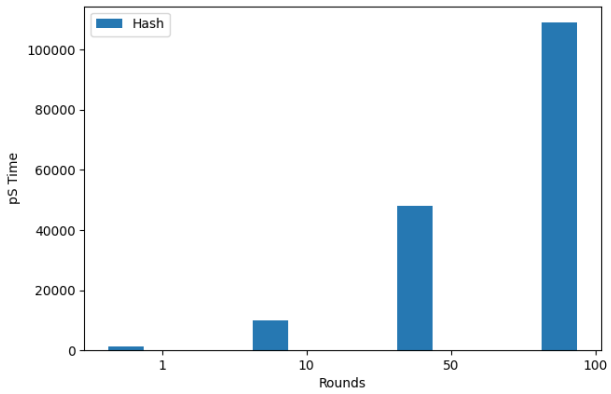

Figure 5: HASH rounds in key expansion algorithm

Typically the size of the readings of a sensor or a group of them is in Ascii format, and in the vast majority of cases, it is of small size (less than 32 characters) and therefore smaller than the 256-bit size, which is the block size of the cipher.

So, for any of the cases, the throughput does not exceed 100 microseconds, which generates an average of 256 bps for a microcontroller without hardware dedicated to cryptography, which is acceptable for some range of IoT or similar applications, but if the encryption and decryption process is performed using the CAU unit, the throughput of 38.4 Mbps is achieved, a speed comparable to the speed of the CPU. With 256-bit packet encryption, the bandwidth of possible encrypted information enters the audio spectrum and, generally, any application that can be performed on a microcontroller of this range.

On the other hand, a measurement of the execution time of the algorithm in the FRDM-K82F was performed using the CAU SHA-256, changing the number of iterations to give the algorithm a higher degree of security. Such a table is shown in Fig. 5, where the execution times are shown concerning the number of iterations of the key expansion process.

Even if a fair number of rounds are generated for this application, they always reach 100 nanoseconds, 1000 times faster than the average time it takes a user to enter each character through the serial port.

Our empirical findings unequivocally validate the prowess of the UID-centric key generation methodology in sculpting a key of optimal width, seamlessly harmonizing with asymmetric encryption paradigms. Further accentuating the system's robustness is the inclusion of dedicated cryptographic hardware, which not only delivers a pronounced uptick in operational efficiency but also dramatically truncates processing intervals associated with both encryption and decryption endeavors. This heightened performance dovetails with the integration of Physical Unclonable Functions (PUFs), which ingeniously exploit inherent, non-deterministic discrepancies present during the manufacturing phase to spawn singular, stochastic keys—rendering replication endeavors ostensibly insurmountable. Furthermore, our analytical assessment illuminated the system's commendable energy frugality paired with impeccable data precision. Collectively, these attributes underscore the system's preeminence as a quintessential cryptographic solution tailored for the intricate demands of secure IoT ecosystems.

5 Conclusion

This research unveils a groundbreaking alternative for fortifying IoT and sensor network applications. At the heart of our proposition is the judicious employment of Physical Unclonable Functions (PUFs) or the device's Unique Identification (UID) number. This strategic integration ensures that each encrypted information packet remains indelibly stamped with an unreplicable signature, thereby infusing the application with an unparalleled security layer whilst concurrently maintaining a discernible digital footprint of the data originator. To harness the full capabilities of the device, we leveraged the MCUxpresso development environment, sculpting our solution in the C programming language, underscored by a bare-metal programming paradigm.

Furthermore, we introduce an avant-garde adaptation to the quintessential *salt* usage in key expansion processes, supplanting the conventional pseudo-random number with the device-specific UID. This approach is meticulously aligned with the PKCS#5 standard delineated by NIST, enhancing the robust management of user session keys across a plethora of application domains. It's worth noting that this innovation builds upon and amplifies the foundational work of the Embedded Information Security (SIE) research group. Our meticulously crafted security framework not only assures data confidentiality across myriad applications deployable on a microcontroller of this computational prowess but also achieves staggering throughputs exceeding 150 Mbps. This is further complemented by an expedited key expansion process, executed with ample iterations to bolster security, yet remarkably swift—surpassing the time a user would typically expend in inputting a single key character.

A synthesis of our research insights reveals the pragmatic feasibility of UID-centric key generation, the pronounced efficiency dividends reaped from dedicated cryptographic hardware, [1], and the holistic robustness of our proposed encryption blueprint. Collectively, our findings champion our encryption paradigm as an optimal cryptographic solution for IoT

ecosystems, adeptly balancing performance metrics, energy frugality, and impregnable security.

Peering into the horizon, we envision augmenting our cryptographic architecture by architecting a comprehensive system adept at managing session keys and PUF or UID numbers. This envisioned system would seamlessly integrate asymmetric encryption algorithms like RSA, [14], or ECC, [15], [11]. Furthermore, we aim to harness cloud-based platforms, such as AWS, renowned for their prowess in secure data transit. This would invariably facilitate robust user access management for stand-alone applications, thereby further solidifying our cryptographic framework.

Acknowledgment:
The authors gratefully acknowledge the use of facilities and equipment provided by the Universidad Distrital Francisco José de Caldas, which enabled the development of this work. Special thanks are extended to the Facultad Tecnológica and the Oficina de Investigaciones for their support, and the evaluation process carried out by the research group SIE on prototypes of ideas and strategies. It is important to clarify that the project was developed with internal resources only and did not receive external funding. The views expressed herein do not necessarily represent those of the Universidad Distrital.

References:

[1] R. Bharathi and N. Parvatham, "Lea-siot: Hardware architecture of lightweight encryption algorithm for secure iot on fpga platform," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 11, no. 1, pp. 720–725, 2020, ISSN: 2156-5570. DOI: 10.14569/IJACSA.2020.0110189.

[2] K. Gurumanapalli and N. Muthuluru, "Feistel network assisted dynamic keying based spn lightweight encryption for iot security," 6, vol. 12, 2021.

[3] F. Maqsood, M. Ahmed, M. Mumtaz, and M. Sha, "Cryptography: A comparative analysis for modern techniques," 2017.

[4] M. Sawand and N. Khan, "Privacy and security mechanisms for ehealth monitoring systems," IEEE, 2012.

[5] H. Akram, D. Konstantas, and M. Mahyoub, "A comprehensive iot attacks survey based on a building-blocked reference model," 3, vol. 9, 2018.

[6] W. Yu and J. Chen, "Masked aes puf: A new puf against hybrid sca-mlas," *Electronics Letters*, vol. 54, no. 10, pp. 618–620, 2018. DOI: 10.1049/el.2018.0735.

[7] M. Fariss, H. Gafif, and A. Toumanari, "A lightweight ECC-based three-factor mutual authentication and key agreement protocol for WSNs in IoT," 6, vol. 13, 2022.

[8] V. V. Thavavel Vaiyapuri Adel Binbusayyis, "Security, privacy and trust in iomt enabled smarthealthcare system: A systematic review of currentand future trends," 2, vol. 12, 2021.

[9] D. Kumar, C. Labrado, R. Badhan, H. Thapliyal, and V. Singh, "Solar cell based physically unclonable function for cybersecurity in IoT devices," in *2018 IEEE Computer Society Annual Symposium on VLSI ISVLSI*, 2018. DOI: 10.1109/isvlsi.2018.00131.

[10] S. Banerjee, V. Odelu, A. K. Das, S. Chattopadhyay, J. J. P. C. Rodrigues, and Y. Park, "Physically secure lightweight anonymous user authentication protocol for internet of things using physically unclonable functions," *IEEE Access*, vol. 7, pp. 85 627–85 644, 2019. DOI: 10.1109/access.2019.2926578.

[11] R. Bhadada and A. Sharma, "Montgomery implantation of ecc over rsa onfpga for public key cryptography application," in *2014 2nd International Conference on Emerging Technology Trends in Electronics, Communication and Networking*, vol. 1, IEEE, 2014.

[12] S. Kumari, "Enhancing the quantum communication channel using a novel quantum binary salt blowfish strategy," *Wireless Personal Communications*, vol. 123, no. 2, pp. 1085–1102, 2021. DOI: 10.1007/s11277-021-09171-y.

[13] M. Laban and M. Drutarovsky, "Leakage free helper data storage in microcontroller based PUF implementation," *Microprocessors and Microsystems*, vol. 87, no. 1, p. 103 369, 2021. DOI: 10.1016/j.micpro.2020.103369.

[14] S. KOTEL and F. SBIAA, "A data security algorithm for the cloud computing based on elliptic curve functions and sha3 signature," 2022.

[15] A. Rahman, I. Ullah, M. Naeem, *et al.*, "A lightweight multi-message and multi-receiver heterogeneous hybrid signcryption scheme based on hyper elliptic curve," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 5, pp. 160–167, 2018.

[16] R. Roman, R. Arjona, J. Arcenegui, and I. Baturone, "Hardware security for eXtended merkle signature scheme using SRAM-based PUFs and TRNGs," in *2020 32nd International Conference on Microelectronics (ICM)*, IEEE,

2020. DOI: 10.1109/icm50269.2020.9331821.

[17] F. Syifaul, P. Aris, A. Trio, and M. Tengku, "Rtl design and testing methodology for uhf rfid passive tag baseband-processor enabling internet-of-things (iot) technology," Tech. Rep. effortless, 2022.

[18] H. Montiel, F. Martínez, and E. Jacinto, "Implementation of password hashing on embedded systems with cryptographic acceleration unit," *(IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 13, no. 2, pp. 171–175, 2022, ISSN: 2156-5570. DOI: 10.14569/IJACSA.2022.0130221.

[19] M. Mohammadinodoushan, D. Hely, B. Cambou, D. Booher, and C. Philabaum, "Implementation of password management system using ternary addressable puf generator," IEEE, 2019.

[20] M. Mohammadinodoushan, B. Cambou, C. R. Philabaum, and N. Duan, "Resilient password manager using physical unclonable functions," *IEEE Access*, vol. 9, pp. 17 060–17 070, 2021. DOI: 10.1109/access.2021.3053307.

[21] I. Karageorgos, M. Isgenc, S. Pagliarini, and L. Pileggi, "Chip-to-chip authentication method based on SRAM PUF and public key cryptography," *Journal of Hardware and Systems Security*, vol. 3, no. 4, pp. 382–396, 2019. DOI: 10.1007/s41635-019-00080-y.

[22] P. Urien, "Innovative atmega8 microcontroler static authentication based on sram puf," in *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, IEEE, 2020.

[23] W. Xiong, A. Schaller, S. Katzenbeisser, and J. Szefer, "Software protection using dynamic PUFs," *IEEE Transactions on Information Forensics and Security*, vol. 15, no. 1, pp. 2053–2068, 2020. DOI: 10.1109/tifs.2019.2955788.

[24] Y.-S. Won and S. Bhasin, "A systematic side-channel evaluation of black box AES in secure MCU: Architecture recovery and retrieval of PUF based secret key," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021. DOI: 10.1109/iscas51556.2021.9401404.

[25] Y. Weize and C. Jia, "Puf-aes-puf: A novel puf architecture against non-invasive attacks," *aXiv*, 2018.

[26] N. Semiconductors, "Kinetis kl82 microcontroller datasheet," 2016.

[27] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable and functions and applications: A and tutorial," 8, vol. 102, IEEE, 2014.

[28] N. Mouha, *Review of the advanced encryption standard*. 2021. DOI: 10.6028/nist.ir.8319.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflicts of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.