Lightweight Stream Ciphers for IoT Applications

Adway Girish
Dept. of Electrical Engineering
IIT Bombay
Mumbai, India
adwaygirish@iitb.ac.in

Fathima Zarin Faizal
Dept. of Electrical Engineering
IIT Bombay
Mumbai, India
180070018@iitb.ac.in

Sai Anirudh M
Dept. of Electrical Engineering
IIT Bombay
Mumbai, India
180070048@iitb.ac.in

***Abstract*—This is the final project report for EE 706: Communication Networks, offered in Spring 2022 at IIT Bombay. We propose Chaotic Espresso, which is a combination of two schemes, one of which is well-established, and the other is a recent phenomenon. Our goal for the project is broadly to study and improve on algorithms for cryptography in the Internet of Things (IoT). We have performed an extensive literature survey for the same, to identify flaws and shortcomings in the existing schemes. We then provide a description of our proposed design and elaborate on its improvements over its predecessors.**

***Index Terms*—Espresso, Chaotic cryptography, Lightweight cryptography, Internet of Things (IoT), Stream cipher**

I. Introduction

The Internet of Things (IoT) refers to an up-and-coming technology that enables a system of interconnected devices, machines or living things to communicate with each other and/or the Internet without the need for human interaction. A typical IoT setup would include end devices fitted with sensors that collect (mostly real-time) data to be shared or analyzed for various purposes such as monitoring, control etc. Such a technology would be privy to vast volumes of data and hence ensuring authentication, confidentiality, data integrity and access control while data is being shared in these systems is essential [1].

Some of the challenges involved in ensuring secure communication in IoT are the following [1]:

1) Complex algorithms are not suitable for the CPU in IoT devices as they have limited capabilities.
2) As some of the devices in the IoT network are likely to be battery-powered, the power consumption of the security algorithm should be minimized.
3) Some IoT applications involve a large number of rudimentary sensors connected to cover a large physical network.
4) The implementation of the security algorithm must be cost-effective and should minimize the number of devices to be deployed.

Conventional cybersecurity cryptography such as AES (Advanced Encryption Standard), RSA (Rivest–Shamir–Adleman), DES (Data Encryption Standard), Blowfish, and RC6 cannot be used here exactly as these algorithms consume a large amount of energy while operating, the IoT network is heterogeneous and dynamic, and due to the need for scalability to larger networks [1]. This calls for the development of Lightweight Cryptography techniques for applications such as IoT and low power, lossy systems in general.

The IoT architecture contains four distinct critical layers: (i) perception layer, (ii) network layer, (iii) middleware layer, and (iv) application layer [2]. The latter two layers use resource-rich devices that can use traditional cryptography for security [1] and hence the focus is on developing lightweight cryptographic techniques for the former two layers.

The aim of the project is to propose a new lightweight cryptography scheme that can be used for IoT applications. This paper is organized into five sections. Section I introduces IoT and motivates the need for lightweight cryptographic techniques. Section II goes into further detail on the broad area of lightweight cryptography. In Section III, we focus on lightweight stream ciphers, which are a particular type of symmetric cipher, which look promising and are widely studied for resource-constrained applications. We explain the concepts of Espresso and a chaos-based cryptographic scheme, Logic, in Sections IV and V respectively. We then describe our proposed 'Chaotic Espresso' in Section VI. We end by outlining the contributions of each team member in Section VII and then providing some concluding remarks in Section VIII.

II. Lightweight Cryptography for IoT

IoT networks generally involve a number of devices, each using different operating systems and communication protocols. This heterogeneity leads to considerable security and privacy threats. In particular, we discuss security in the network and perception layers due to resource constraints mentioned previously.

The network layer involves secure routing and transmission of data across the IoT infrastructure. It generally uses protocols such as Zigbee, Bluetooth and IR for transmission. It is susceptible to attacks such as eavesdropping, device cloning, spoofing, DDoS and key-related attacks. The perception layer consists of devices such as sensors and actuators, that gather, process or modify data. It consists of two parts: perception nodes and the perception network. It is susceptible to physical capture of nodes, code injection, jamming, replay and battery draining attacks.

Due to limitations of processing power, memory, power consumption, etc. present in IoT devices, lightweight cryptography is necessary to operate on the given network. A network's effectiveness is determined by its design complexity (gate value/GE), power consumption, throughput and the underlying CMOS technology. Securing communication consists of two components: cipher algorithms, used to encrypt and decrypt data, and key management, used to generate, distribute, store and destroy secret keys. Cipher algorithms may be symmetric, which use smaller key lengths at the cost of more vulnerability to attack, or asymmetric, which use greater complexity at the cost of being slower. There also exist lightweight hashing functions such as LNHASH, PHOTON, HVH and SPONGENT, and lightweight message authentication codes (MACs) such as LightMAC and CHASKEY.

Some examples of recent lightweight cryptographic protocols in IoT security along with their features are given below:

- Lightweight hybrid cryptography (LWHC): uses a combination of LED and PRESENT ciphers (64 bit block cipher), however uses a 128 bit SPECK key scheduling algorithm, which makes the protocol both lightweight and secure from key attacks. It is still vulnerable to other attacks
- Lightweight stream cipher (LSC): uses cryptographic primitives, which change dynamically and use less operations/processing power from block to block. This protocol is highly secure and resistant to statistical, algebraic and brute force attacks, but not to disclosure and de-synchronisation attacks
- SAT_Jo system: based on substitution-permutation network of a block cipher with DES and PRESENT with 31 rounds. It offers adequate security for tag-based applications, and a good balance between performance/resource requirements and security.

The following parameters are used to evaluate performance of a lightweight cryptography protocol:

Key size: Keys of greater length provide more robustness but require more power and complexity. Typical 128 bit keys such as AES, LED, RECTANGLE are unsuitable for resource-constrained devices. However, the modified QARMA (64 bit), SAT_Jo, modified PRESENT, Piccolo, KTANTAN (80 bit) are all suitable for IoT devices.

Block size: Larger block sizes (such as in AES - 128 bit blocks) require more computation and battery power. Hence, smaller block sizes (typically 64 bits) are suitable for IoT devices. SIMON applies one of the lowest block lengths of 46 bits

Gate area: Gate Equivalent (GE) measures the physical area required to execute an algorithm. Power consumption can be determined accordingly using GE value and CMOS technology. ISO/IEC standard specifies that lightweight cryptography should have GE value 1000-2000. For example, GE value of AES is 2400, whereas that of SAT_Jo us 1167.

Number of rounds: Round based execution is used in encryption using the key, with more number of rounds giving greater security. Cryptographic designing aims to decrease the number of rounds necessary in the algorithm. AES typically uses less rounds (10-14) but has large GE. Piccolo is a suitably lightweight algorithm that uses less number of rounds (25) compared to others like SAT_Jo (31) and SFN (32).

Latency: It is the time taken to generate encrypted output after initial approach of encryption. It is critical for real time applications. Low computational complexity is necessary to ensure low latency. PRINCE, QARMA and modified QARMA have some of the lowest latencies (12, 1, 27 cycles respectively).

Throughput: Measure of number of bits transformed per unit time at specified frequency. IoT applications require moderate throughput, lower than that in traditional cryptography. SAT_Jo cipher allows high throughput (14.9Mbps) at low latency

Confusion/diffusion properties: Confusion is the property arising from substitution in a cipher (through S-box), distorting the relationship between plaintext, key and ciphertext. Diffusion is the property arising from permutation (through P-box), scattering the statistical structure of plaintext over ciphertext.

Broadly, by varying the key complexity, architecture and computational iterations, there is a trade-off between the cost, performance and security of the IoT network, as demonstrated in Figure 1.

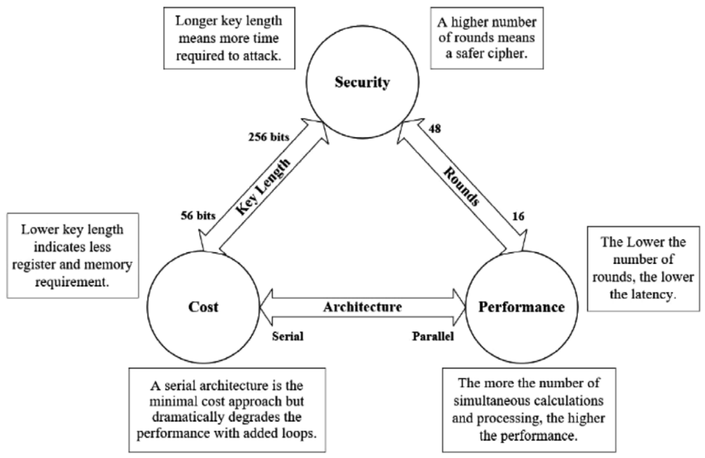
Longer key length means more time required to attack.
A higher number of rounds means a safer cipher.
Security
256 bits
Key Length
56 bits
48
Rounds
16
Lower key length indicates less register and memory requirement.
The Lower the number of rounds, the lower the latency.
Cost
Architecture
Performance
Serial
Parallel
A serial architecture is the minimal cost approach but dramatically degrades the performance with added loops.
The more the number of simultaneous calculations and processing, the higher the performance.

Fig. 1. The trade-off between cost, performance and security needed while designed cryptography schemes for IoT [1]

III. Prior Work on Stream Ciphers

Stream ciphers encrypt and decrypt data one bit at a time. The ciphertext is obtained by simply XORing the plaintext with the keystream, bit-by-bit. They generally make use of linear and nonlinear feedback shift registers, hence are faster, of a low complexity, and consume less power [3]. One drawback, however, is that the setup prior to the first use requires a lot more effort. Nonetheless, they are widely applied in wireless sensor networks and cell phones thanks to their simplicity.

Around the early 2000s, the interest in stream ciphers had actually begun to decrease, as seen by the replacement of the stream cipher RC4 by the block cipher AES in the improved wireless standard 802.11i. Their simplicity of implementation and design, however, meant they remained enticing prospects, and a large amount of work on stream ciphers began during that period thanks to the European Network of Excellence in Cryptology (ECRYPT) introducing the eSTREAM project [4], encouraging hardware and software implementations of stream ciphers. The pioneering works in this direction include Trivium [5], Grain [6], Fruit [7], MICKEY [8], Espresso [9], LIZARD [10] and Sprout [11], among others. Each of them introduced what was, at the time, a radically different idea, that gives us different dimensions to pursue while designing cryptographic systems.

A. *Predecessors*

Grain [6] makes use of a nonlinear filter function, which allows it to perform better than purely hardware focussed schemes. **Trivium** [5] helped revive the interest in stream ciphers by providing a reliable strategy, as opposed to the previously structure-less schemes that were all very different from each other, also providing flexibility for hardware implementation. In typical stream ciphers, we require that the size of the internal state must be at least twice that of its security level, to protect against time-memory-data trade-off (TMDTO) attack [12]. However, the design key can be exploited to reduce the internal state size, as done in **Fruit** [7]. **LIZARD** [10] obtains a similar improvement by modifying the initialization, making it harder to recover the secret key. Even at the time of introduction, **MICKEY** [8] was not intended to be particularly fast, but it was designed specifically for resource-constrained devices.

Unfortunately, while stream ciphers offer simpler implementations than block ciphers, they have also been discovered to be much weaker in the face of attacks. Several vulnerabilities have been exposed in recent years thanks to the advancements in decoding technology. Additionally, to apply these schemes in protecting resource-constrained IoT devices, we require that they be as efficient and low-power as possible. For example, Grain suffers from having large propagation delays and Trivium requires too many flip-flops to achieve the same security level.

B. *Newer Schemes*

Most of the above schemes have been improved with minor variations, for example, Grain-128a [13] offers additional authentication. Another cipher that has achieved widespread attention is **Espresso** [9], designed primarily for application in 5G communication. It belongs to the Grain family, but manages to obtain a significantly shorter propagation delay, while also providing a formal analysis and roubstness to various attacks, such as linear approximation, algebraic, TDMTO, and chosen IV attacks. It has, however, been shown to be vulnerable to differential fault attacks [14].

A Lightweight Stream Cipher (**LSC**) scheme [15] has been introduced, which manages to outperform the traditional AES scheme, by using a dynamic key approach. However, using dynamic keys requires large memory to store the parameters and computational power to encrypt and decrypt. While LSC uses less resources than previous dynamic key approaches, it would save a lot of precious resources that could be used elsewhere on the IoT device if dynamic keys are avoided.

Finally, we also present the idea of chaotic cryptography [16]. They were first applied to stream ciphers by combining them with nonlinear feedback shift registers in **Logic** [17]. Entropy analysis techniques can be used to mathematically analyze them. They have been shown to be secure against algebraic, TMDTO, fault, linear approximation and correlation attacks. However, they suffer from floating-point computations and finite periodicity.

We now go into Espresso and Chaos-based cryptography in some detail, to set up the basis to introduce our work.

IV. Espresso

A. *Preliminaries*

An n-bit Feedback Shift Register (FSR) consists of n stages that store a binary value. Each stage $i \in \{0, \dots, n-1\}$ is associated with a state variable x_i and a feedback function $f_i : \mathrm{GF}(2^n) \to \mathrm{GF}(2)$, which is used to update the value of the state variable at the end of each clock cycle. If all the feedback functions are linear, then the FSR is called Linear (LFSR), else Non-Linear (NFSR). A *state* of an FSR is a vector of values of its state variables. The next state is computed by updating the value of each stage to that of its corresponding feedback function. The *period* of an FSR is the length of the longest cyclic output sequence it produces. In the Galois configuration, the feedback can potentially be applied to every stage while in the Fibonacci configuration, the feedback is applied to the input stage only.

B. *Design*

Espresso [9] uses FSRs in the Galois configuration, which makes the feedback functions smaller compared to the Fibonacci configuration (which is the case in Grain) and hence reduces the propagation delay. For cryptographic analysis to be possible, the Galois NFSR is transformed to an NFSR which resembles a Fibonacci configuration.

There is a 256 bit NFSR G in the Galois configuration and a 20-variable nonlinear output function. The feedback function for stage i is g_i. Nine of these are non-trivial and the exact formulations of these functions can be found in [9]. The output function $z(x)$ is:

$$\begin{aligned} z(x) = {} & x_{80} \oplus x_{99} \oplus x_{137} \oplus x_{227} \oplus x_{222} \oplus x_{187} \oplus x_{243}x_{217} \\ & \oplus x_{247}x_{231} \oplus x_{213}x_{235} \oplus x_{255}x_{251} \oplus x_{181}x_{239} \\ & \oplus x_{174}x_{44} \oplus x_{164}x_{29} \oplus x_{255}x_{247}x_{243}x_{213}x_{181}x_{174}. \end{aligned}$$

NFSRs in the Fibonacci configuration have been studied extensively and hence the need for transforming from Galois to an equivalent configuration F that resembles the Fibonacci configuration. This transformation is shown in Fig. 2. F has only two non-trivial feedback functions g_{255} and g_{217}. k_i, $0 \leq i \leq 127$ denotes the key and IV_i, $0 \leq i \leq 95$ denotes the initialization vector. The shift register is loaded as:

$$\begin{aligned} x_i &= k_i, && 0 \leq i \leq 127 \\ x_i &= \mathrm{IV}_{i-128}, && 128 \leq i \leq 223 \\ x_i &= 1, && 224 \leq i \leq 254 \\ x_i &= 0, && i = 255 \end{aligned}$$

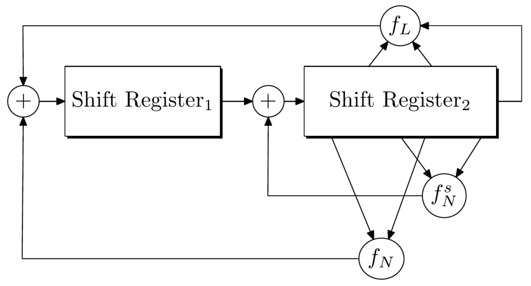

Fig. 2. NFSR for Espresso

C. Takeaways

Here are the main points that are relevant to us, from the above design scheme as well the analysis from [9]:

1) A cryptographic analysis is made possible by the use of a Fibonacci configuration NFSR.
2) A linear approximation attack as was possible against Grain [6] is no longer possible as it requires 2^{127} samples, making it inefficient.
3) Time-Memory-Data Trade-Off (TMDTO) attacks require a precomputation time of 2^{168}, which is very large compared to a key size of 128 bits.
4) There is also protection from the chosen IV attacks thanks to the large number of initialization steps which adds sufficient randomness.

V. Logic

Logic [17] is a lightweight chaos-based stream cipher which has desirable properties such as extreme sensitivity to initial conditions, pseudo-random behavior, and long-period instability. These arise due to the use of the chaos equation, for some constant μ,

$$x(n+1) = \mu x(n)\,[1 - x(n)], \quad \mu \in [0,4],\ x(n) \in (0,1).$$

The key point is that small differences in the initial condition can lead to large differences in the keystream generated. By restricting our sequences to be represented by L-bit precision, taking $\mu = 4$ and $L = 32$, our equation becomes

$$X_{n+1} = 4X_n \frac{(2^L - X_n)}{2^L}.$$

A. Design

The flow diagram of the stream cipher is shown in Fig. 3. The main constituents are the following:

- the feedback polynomial of NFSR-1 (Nonlinear Shift Feedback Register), $g(x)$,
- the feedback polynomial of NFSR-2, $f(x)$,
- the filtering function $h(x)$,
- the chaotic unit, Logistic,
- the multiplexers M1, M2, M3,
- the output $Z(x)$.

The NFSRs drive the key generation. The feedback polynomials are fixed and their expressions can be found in [17]. We omit them here since they are long and unnecessary for our discussion. The task of the chaotic module is to introduce the confusion in the NFSRs. The other components are simply a combination of several Grain [6] series ciphers. The final output is a combination of the outputs of the filter, NFSR-1 and the chaotic module, which adds more confusion.

B. Takeaways

From the above design and its analysis from the paper [17], we make the following observations that are crucial to our work.

1) The main part is the logistic chaos module.
2) The role of the other components is just to make it work as a stream cipher. Hence the other components may be suitably replaced by a different stream cipher as needed.
3) The use of two NFSRs provides protection from algebraic attacks by increasing the level of nonlinearity.
4) Since the NFSRs can be assumed to be independent of each other, we also have protection against correlation attacks.
5) This is no longer vulnerable to fault attacks (where a single bit error is introduced into the NFSR), as a mirror image can be used in the two-stage NFSR setup that we have here.
6) This scheme also passes the NIST statistical tests.

VI. Our Contributions

In this section, we propose a new lightweight cryptographic stream ciphers that combines the best parts of Espresso with those of Logic, named ‘Chaotic Espresso’.

A rough flow diagram of the same is shown in Fig. 4. The key idea is that we replace the NFSRs inspired by Grain with the Espresso. The various inputs and outputs from the ‘Espresso’ block in the figure are described later.

A. Design

Here we provide details of the how the Espresso is incorporated with the Logistic stream cipher setup.

- We require that the Logistic module interacts with all NFSRs in Espresso, so we first add another input at the left end of Fig. 2, which arises from the chaotic Logistic module.
- Further, we require a recursive relation to complete the chaos loop, which is done by connecting the output of the

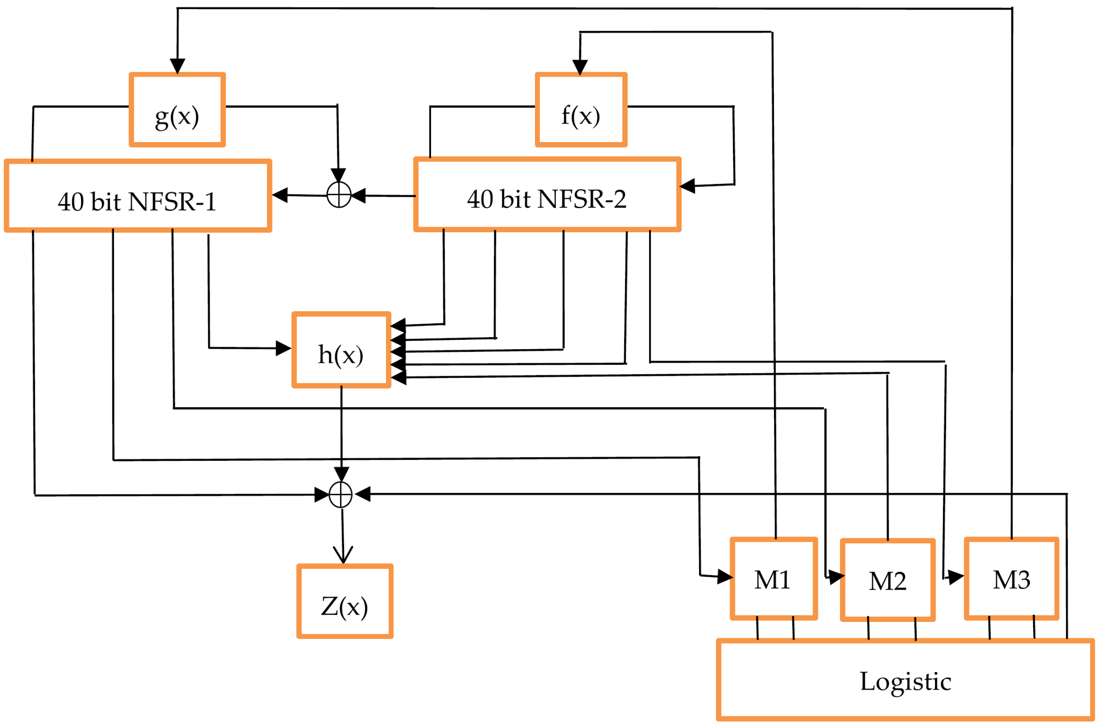

Fig. 3. The logic stream cipher

register at the right end of Fig. 2 back to the multiplexers connected to the chaotic module, after passing them through an appropriate filter function.

The exact functions and indices will need to be tweaked following numerical simulations.

B. *Strengths*

The benefits this provides are as follows:

1) Since we are combining the two schemes without removing the effect of either one, we have that the securities provided by each scheme remains intact.
2) In addition, this has the benefit that unlike Logic, a linear approximation attack is not possible.
3) Further, unlike standard Espresso, differential fault attacks will not work on Chaotic Espresso as we use a two-stage NFSR, which allows for mirroring or masking in the hardware.
4) Espresso has a complicated key initialization stage, which ensures protection from chosen key attacks, but that can be made simpler as the chaotic nature of the Logistic module provides this protection directly.
5) Espresso itself uses a smaller chip area than a standard two-NFSR implementation.

VII. Individual Contributions

Our work strategy was as follows: each member of the group first proposes some ideas and literature to study, following which we all go through them and discuss together to come up with a plan and identify the next step. Here are the contributions of each member in terms of introducing possible directions to the discussion.

- Adway Girish (180070002): suggested lightweight cryptography as the primary topic and identified the application of chaotic cryptography to stream ciphers; identified the key takeaways from Espresso and proposed the design for combining Espresso with Logic.
- Fathima Zarin Faizal (180070018): suggested security vulnerabilities in Wi-Fi as the primary topic and reviewed Espresso; identified the key takeaways from Espresso and Chaos, and the improvements in Chaotic Espresso.
- Sai Anirudh M (180070048): suggested zero trust architecture as the primary topic and identified stream ciphers as the area of focus; proposed the design of and identified the improvements in Chaotic Espresso.

VIII. Conclusion

We have proposed ‘Chaotic Espresso’, a new lightweight stream cipher that combines the principles of the recently growing chaotic cryptography with Espresso, which is well-established. We claim that our proposed design is capable of handling and dealing with all well-known, standard attacks. A weakness of our proposal is that we do not have simulation results to support our claims. This is due to the lack of time and computational resources at our disposal, and we leave it as potential future work. Further future work is possible in the direction of entropy analyses of the proposed scheme, which provides theoretical support to our claims.

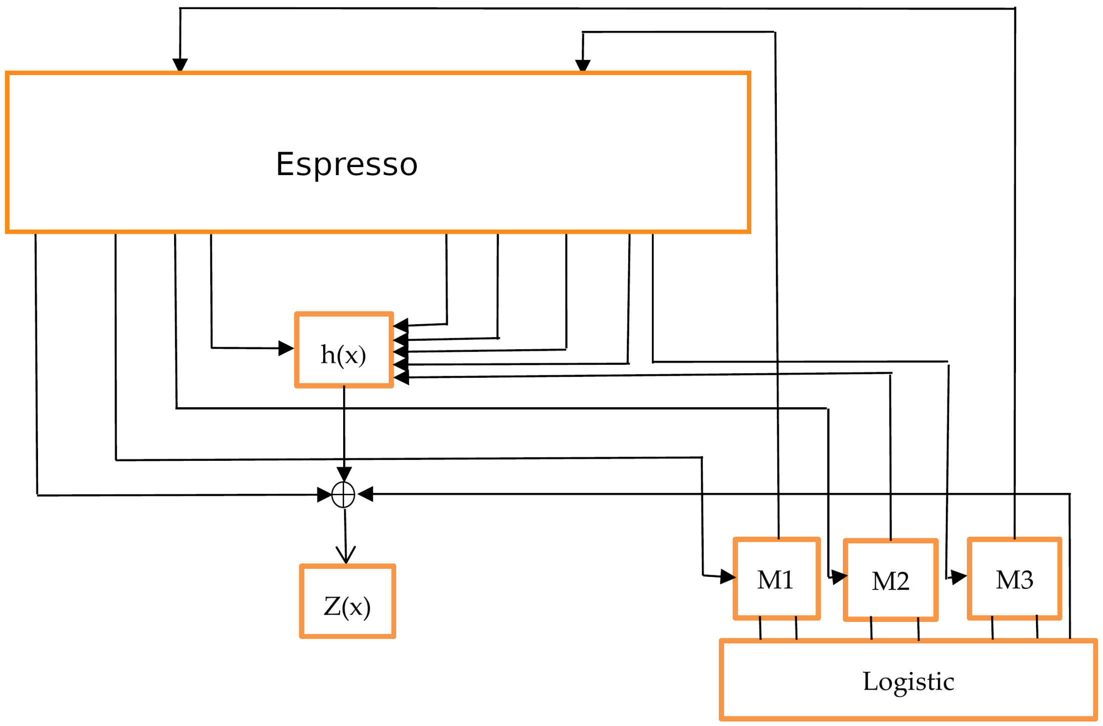

Fig. 4. Proposed Chaotic Espresso

References

[1] M. Rana, Q. Mamun, and R. Islam, "Lightweight cryptography in IoT networks: A survey," vol. 129, pp. 77–89.

[2] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in internet-of-things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.

[3] S. A. Jassim and A. K. Farhan, "A survey on stream ciphers for constrained environments," in *2021 1st Babylon International Conference on Information Technology and Science (BICITS)*, pp. 228–233.

[4] "Estream: The ecrypt stream cipher project."

[5] C. De Cannière, "Trivium: A stream cipher construction inspired by block cipher design principles," in *Information Security* (S. K. Katsikas, J. López, M. Backes, S. Gritzalis, and B. Preneel, eds.), (Berlin, Heidelberg), pp. 171–186, Springer Berlin Heidelberg, 2006.

[6] M. Hell, T. Johansson, and W. Meier, "Grain: A stream cipher for constrained environments," *Int. J. Wire. Mob. Comput.*, vol. 2, p. 86–93, may 2007.

[7] H. H. Vahid Amin Ghafari and Y. Chen, "Fruit-v2: Ultra-lightweight stream cipher with shorter internal state." Cryptology ePrint Archive, Report 2016/355, 2016. https://ia.cr/2016/355.

[8] M. D. Steve Babbage, "The stream cipher mickey 2.0," 2006. https://www.ecrypt.eu.org/stream/mickeypf.html.

[9] E. Dubrova and M. Hell, "Espresso: A stream cipher for 5g wireless communication systems," *Cryptography and Communications*, vol. 9, 03 2017.

[10] M. Hamann, M. Krause, and W. Meier, "Lizard - a lightweight stream cipher for power-constrained devices." Cryptology ePrint Archive, Report 2016/926, 2016. https://ia.cr/2016/926.

[11] F. Armknecht and V. Mikhalev, "On lightweight stream ciphers with shorter internal states," vol. 9054, pp. 451–470, 03 2015.

[12] M. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Transactions on Information Theory*, vol. 26, no. 4, pp. 401–406, 1980.

[13] M. Ågren, M. Hell, T. Johansson, and W. Meier, "Grain-128a: A new version of grain-128 with optional authentication," *IJWMC*, vol. 5, pp. 48–59, 12 2011.

[14] B. Bathe, S. Tiwari, R. Anand, D. Roy, and S. Maitra, "Differential fault attack onÂ espresso," in *Progress in Cryptology – INDOCRYPT 2021* (A. Adhikari, R. Küsters, and B. Preneel, eds.), (Cham), pp. 271–286, Springer International Publishing, 2021.

[15] H. Noura, R. Couturier, C. Pham, and A. Chehab, "Lightweight stream cipher scheme for resource-constrained iot devices," in *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1–8, 2019.

[16] J. AmigÃ³, L. Kocarev, and J. Szczepanski, "Theory and practice of chaotic cryptography," *Physics Letters A*, vol. 366, no. 3, pp. 211–216, 2007.

[17] L. Ding, C. Liu, Y. Zhang, and Q. Ding, "A new lightweight stream cipher based on chaos," vol. 11, no. 7, p. 853. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute.