Hardware Implementation of Lightweight Chaos-Based Stream Cipher

Guillaume Gautier*, Maguy Le Glatin*, Safwan El Assad†, Wassim Hamidouche*,
Olivier Deforges*, Sylvain Guilley‡, Adrien Facon‡
* Univ Rennes, INSA Rennes, CNRS, IETR - UMR 6164, F-35000 Rennes, France
Email: guillaume.gautier@insa-rennes.fr, maguy.le-glatin@insa-rennes.fr,
wassim.hamidouche@insa-rennes.fr, olivier.deforges@insa-rennes.fr
†Polytech Nantes, CNRS, IETR - UMR 6164, F-44000 Nantes, France
Email: safwan.elassad@univ-nantes.fr
‡ Secure-IC SAS, F-35510 Cesson-Sévigné, France
Email: sylvain.guilley@secure-ic.com, adrien.facon@secure-ic.com

***Abstract*—Due to the proliferation of connected devices, the development of secured and low-resource cryptographic systems has become a real challenge. In fact, ciphering algorithms have not been thought to be implemented on embedded platforms with limited computing, memory and energy resources. This paper addresses a hardware implementation of a chaos-based stream cipher is initially optimized for software. First, the structure of the cipher is investigated. Then, its hardware implementation on a Zynq7000 platform is proposed considering both throughput performance and logic resources usage. The proposed design is compared to hardware implementations of existing stream ciphers including chaos-based ones, Advanced Encryption Standard (AES), Rabbit, Salsa20 and Trivium.**

Keywords–Chaos-based stream ciphers; Hardware implementation; Lightweight stream cipher; Computational performance.

I. INTRODUCTION

The need of encryption methods has nearly always existed to protect sensitive information. The number of connected devices is constantly and rapidly increasing. Those devices are communicating between each other through multiple channels exchanging information, such as confidential messages. In this context, the protection of sensitive data exchanged over networks is necessary. This can be done thanks to cryptography. It consists in making the information unintelligible for a person from outside the application by coding it with a secret key. Therefore, it becomes necessary to develop new lightweight cryptographic systems that can be embedded on these connected devices with low energy and computing resources.

In the literature, multiple ciphers are defined and implemented on hardware devices. We especially studied the stream ciphers in which the output is obtained by performing an eXclusive OR (XOR) between the input of the cipher and the output of a random generator. For example, Advanced Encryption Standard (AES), the state of the art encryption standard, is available in multiple versions [1], [2], each optimizing a trade-off between surface and speed. Moreover, some ciphers, like Trivium [3], are designed to be hardware friendly and minimize logic resources. Some ciphers, based on chaos theory, have also hardware implementations [4], [5]. Those ciphers usually require a high logic resources to digitize chaotic systems.

Based on works in [6], a LightWeight Chaos-Based Stream Cipher (LWCB SC) is proposed in this paper. This design is implemented on Field-Programmable Gate Array (FPGA) hardware platform. The system includes some counter-measures against Side Channel Attacks (SCA) [7], such as Correlation Power Analysis (CPA) and Differential Power Analysis (DPA). The proposed hardware implementation achieves a throughput of 565 Mbps at an operating frequency of 18.5 MHz .

The rest of this paper is organised as follows. The existing stream ciphers are first presented in Section II. Section III investigates the design and details of the hardware implementation of the chaos-based stream cipher. The performance of the proposed system is assessed in Section IV in terms of both throughput and used logic resources. Section IV accesses the performance of the proposed implementation in terms of throughput and used logic resources. Finally, Section V concludes this paper.

II. RELATED WORK

A. Existing stream ciphers

There are several hardware implementation of the state of the art stream ciphers. This section gives a brief review of the existing implementations.

1) AES: AES [8] developed in 2001 is the most widely used system. It is considered as a stream cipher only in its CounTeR (CTR) mode. It takes 128 bits as input and can use a key of 128, 192 or 256 bits. Its round function consists of three layers including key addition layer, byte substitution layer called S-Box and diffusion layer.

2) Rabbit: The Rabbit stream cipher [9] is one of the most effective algorithms of the eSTREAM project. This project was launched in 2004 to create new stream ciphers for dedicated designs. The cipher is not based on S-Boxes but on 8-32 bits state variables and counters.

3) Salsa20: Salsa20 [10] is based on a hash function. This latter is implemented with simple operations as additions, XOR and rotation. This cipher is very fast but presents some security weaknesses. Indeed, several attacks have been concluded against it.

4) Trivium: Trivium [3] is also a cipher from the eSTREAM Project. It is particularly well suited for applications requiring a flexible hardware implementation. The 288-bit internal state is stored in three shift registers which are the heart of the cipher and can be viewed as a circular register.

 ISBN: 978-1-61208-743-6

B. Existing chaotic systems

Several chaotic systems have been developed and used for designing chaotic hardware key generation for secure cryptography systems. Lorenz's [4] and Lü's [5] systems are the famous ones. A chaotic system can be considered as discrete or continuous. The previously cited systems are continuous and defined by a differential equations.

In [4], a cipher to encrypt images is developed and implemented on an FPGA platform. It uses a key generator based on the Lorenz's chaotic system. Another example of such system is presented in [5] where a Lü's-system-based-chaotic key generator is used.

III. Proposed hardware chaos-based stream cipher

The proposed solution is based on the generator previously presented in [6]. This generator consists of two cells that use different chaotic maps called Skew Tent map and PieceWise Linear Chaotic (PWLC) map. These maps are encapsulated into an Infinite Impulse Response (IIR) filter to form the two cells. Finally, the outputs of the two cells are XORed to generate the key stream [6].

The solution described in this paper aims to improve the security of the chaos-based system introduced in [6] by increasing its resilience against SCA. Figure 1 illustrates the block diagram of the enhanced chaos-based generator where the new blocks are highlighted in red. The Skew Tent map is replaced by 4D map defined by a discrete Chebyshev polynomial T_4 of degree 4. A second Linear Feedback Shift Register (LFSR) block is added to overcome some inconsistencies of the 4D map. The outputs of the two cells are weakly coupled before being fed back to the recursive cells illustrated in green in Figure 1.

The cipher text C is created with a simple XOR operation between the plain text P and a key stream X_G generated by the proposed system

$$C = P \oplus X_G.$$

A. Hardware-friendly architecture of the generator

The block diagram of the proposed generator is depicted in Figure 1. We have defined a hardware high level module that instantiates the two cells of the generator, the weak coupling and the key stream's output. The generator module takes as input a secret key and an Initial Vector (IV).

The output key stream X_G corresponds to the sum of the outputs of the two chaotic maps X_{4D} and X_P after a number of iteration tr, defined in the secret key

$$X_G(n) = \begin{cases} X_{4D}(n) + X_P(n) & \text{if } n > tr, \\ 0 & \text{otherwise,} \end{cases}$$

with n is the iteration number, $X_{4D}(n)$ and $X_P(n)$ are the outputs of the 4D and PWLC cells at iteration n, respectively.

To introduce more resilience against algebraic attacks and SCA, we define a weak coupling block $[B]$

$$\begin{bmatrix} XC_{4D}(n) \\ XC_P(n) \end{bmatrix} = \begin{bmatrix} 2^N - B_{11} & B_{12} \\ B_{21} & 2^N - B_{22} \end{bmatrix} \begin{bmatrix} X_{4D}(n-1) \\ X_P(n-1) \end{bmatrix},$$

with N is bit depth of the system, B_{ij} are coefficients defined in the secret key, $XC_{4D}(n)$ and $XC_P(n)$ are the outputs of weak coupling block at iteration n.

The block diagrams of the two cells are delimited in orange in Figure 1. Both cells are composed of a recursive cell illustrated in green, a map and LFSRs. The next two paragraphs present the hardware implementations of the chaotic maps: 4D and PWLC.

1) 4D map: The 4D map is defined by a discrete version of the Chebyshev polynomial of degree 4

$$4D_{MAP}(X) = (X - 2^{2N-1})^4 - 2^{2N-2} \times (X - 2^{2N-1})^2, \quad (1)$$

where X corresponds to Xin_{4D} in Figure 1.

To implement the new 4D map, (1) is expressed to minimize the logic resources of the hardware implementation

$$\begin{aligned} 4D_{MAP}(X) &= (X - 2^{2N-1})^4 - 2^{2N-2} \times (X - 2^{2N-1})^2 \\ &= Y^2 - 2^{2N-2} \times Y \\ &= [Y \times Y - Y << (N-2)] >> (3N-6), \\ \text{with } Y &= (X - 2^{N-1})^2 = X \times X + 2^{N-2} - X << 2^{N-1}. \end{aligned} \quad (2)$$

Implementation of (2) requires only two multipliers of 32 and 64 bits inputs, respectively.

2) PWLC map: The PWLC map is defined by (3)

$$PLWCmap(X, P_P) = \begin{cases} C_1 \times X & \text{if } 0 < X < P_P, \\ C_2 \times (X - P_P) & \text{if } P_P < X < 2^{N-1}, \\ C_2 \times (2^N - P_P - X) & \text{if } 2^{N-1} < X < 2^N - P_P, \\ C_1 \times (2^N - X) & \text{if } 2^N - P_P < X < 2^N, \\ 2^N - 1 & \text{otherwise,} \end{cases} \quad (3)$$

where X is the input of the PWLC map Xin_P and P_P is the parameter of the PWLC map defined in the secret key, C_1 and C_2 are two constants derived from P_P.

In the proposed solution, the ratios C_1 and C_2 are precomputed by the key generator to avoid the implementation a resource-intensive divider. Then, to use the minimum number of multipliers, without reducing the throughput of the generator, the inputs of the multiplier are selected by multiplexers.

B. A counter-measure against SCA

To protect the generator against CPA and DPA attacks [7], masking operations are added to the recursive cells. The aim of masking operations is to randomize intermediate results for the same couple (secret key, IV). The masking is performed by adding a random value to the outputs of the weak coupling XC_{4D} and XC_P

$$XM_{4D}(n) = XC_{4D}(n) + mask_{4D}(n),$$

where XM_{4D} is the output of the 4D-cell mask operation at iteration n, $mask_{4D}(n)$ is a random value, generated by XOR Shift Random Number Generator of integer values in the interval $[0, 2^N - 1]$. The same calculation is performed for the output of the PWLC-cell mask $XM_P(n)$ with $XC_P(n)$ and $mask_P(n)$.

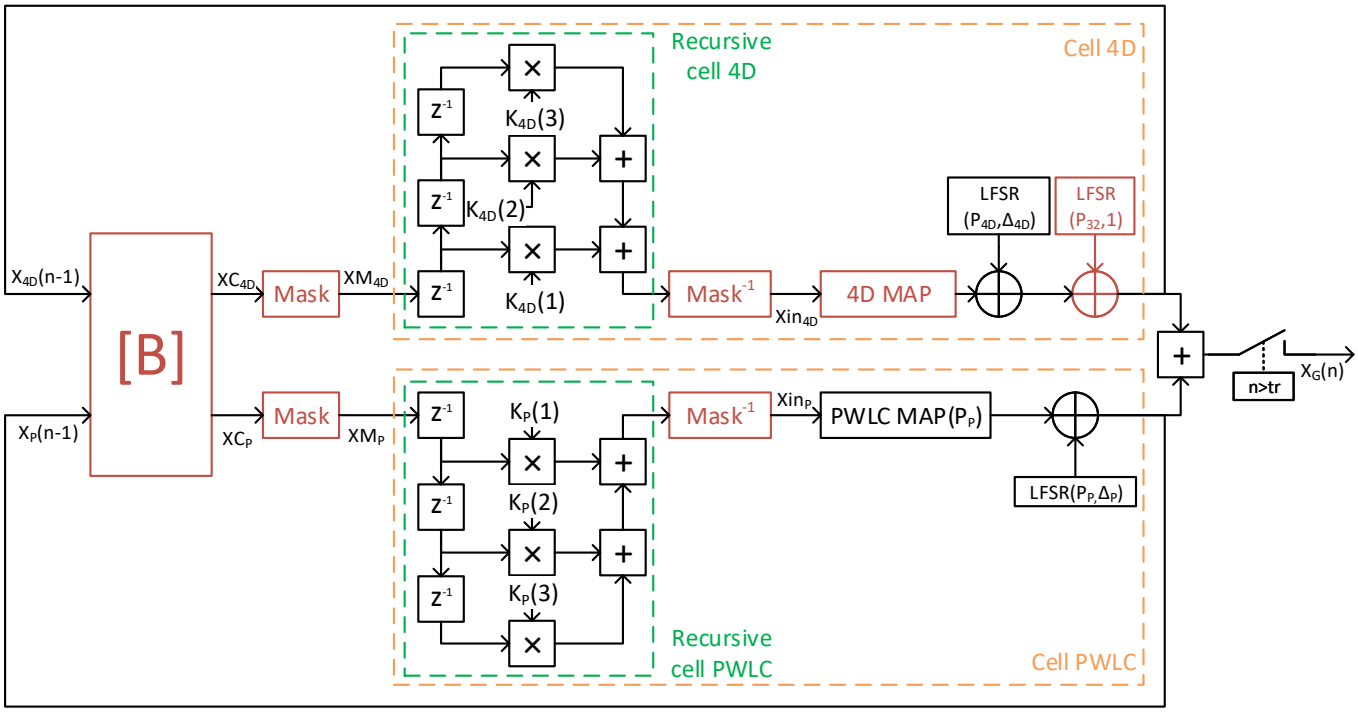

Figure 1. Improved PCNG diagram. In red are shown the differences with [6].
In green are highlighted the recursive cell. Orange dotted line delimits the two cells.

To obtain the same key stream X_G for the same couple (key, IV), the mask operations are reverted before the chaotic maps. The inverse mask operation is performed after the recursive cells by subtracting the transform of the mask value from the output value of the recursive cell.

$$Xin_{4D}(n) = \sum_{i=1}^{D} XM_{4D}(n-i) \times K_{4D}(i) - \sum_{i=1}^{D} mask_{4D}(n-i) \times K_{4D}(i),$$

where $K_{4D}(i)$ is the i^{th} coefficient of the recursive cells given in the secret key and D is the number of delays in the recursive cells. Identically, $Xin_{4D}(n)$ is a function of $XM_{4D}(n-i)$, $K_{4D}(i)$ and $mask_{4D}(n-i)$.

IV. Results and Discussion

In this section, an internal module of the cipher is implemented and tested. Some comparisons with the state-of-the-art are also provided. To evaluate the performance in terms the resources usage and speed, Xilinx Vivado set of tools is used. In these experiments, the bit depth N is set to 32 bits and the number of delay D inside the recursive cells is set to 3.

A. Implementation of the LWCB SC

Table I gives the hardware resources used for the PCNG. It uses in total 2,363 Look-Up Tables (LUTs) and 96 DSP blocks. Most of the resources are used by the recursive cells with 1,087 LUTs and 42 DSP blocks for the PWLC cell and 1,001 LUTs and 44 DSP blocks for the 4D one.

TABLE I. Hardware resources usage of the proposed implementation

Component	LUTs	DSP Blocks
Output Adder	102	0
PWLC Cell	1,087	42
4D Cell	1,001	44
4D Mask	24	0
PWLC Mask	39	0
Coupling Matrix $[B]$	110	10
Total	2,363	96

The proposed implementation produces one sample of the PCNG at each clock cycle and can operate at 18.5 MHz with a throughput of 565 Mbps.

B. Comparison with other existing stream ciphers

Table III and II present a comparison with existing implementations of the ciphers presented in Section II.

TABLE II. Speed performance comparison of several systems

Cipher	Max Freq (MHz)	Throughput (Mbps)
LWCB SC	**18.5**	**565**
Lorenz's chaotic system [4]	15	124
Lü's chaotic system [5]	23	183
AES [1]	644	84,449
	886	11,776
AES [2]	290	232
Rabbit [11]	NC	9,380
Trivium [12]	240	254
Trivium [13]	201	201
Salsa20 [14] (Spartan 3)	19	911
Salsa20 [14] (Spartan 6)	48	2,519

 ISBN: 978-1-61208-743-6

TABLE III. HARDWARE RESOURCES USAGE COMPARISON OF SEVERAL SYSTEMS

Cipher	Device	Area (nb of LUTs)	DSP Blocks
LWCB SC	**Zynq7000**	**2,363 (4.44%)**	**96 (43.64%)**
Lorenz's chaotic system [4]	Virtex II	2,718	40
Lü's chaotic system [5]	Virtex II	1,926	40
AES [1]	Virtex VI	9,276	NC
	Spartan 6	9,375	NC
AES [2]	Spartan 2	444	NC
Rabbit [11]	Virtex V	2,272	24
Trivium [12]	Spartan 3	100	NC
Trivium [13]	Spartan 3	376	NC
Salsa20 [14]	Spartan 3	3,374	NC
	Spartan 6	2955	NC

It can be noted that the fastest cipher is AES [1]. The AES implementations presented in [1], [2] operate at 644 MHz and 886 MHz and the throughput is equal to 84,448 Mbps and 11,776 Mbps, respectively. Rabbit [11] implementation reaches AES performance with a throughput of 9,380 Mbps. Even though, Trivium [12], [13] reaches a high frequency the throughput is not better than other slow frequency ciphers. For example, Salsa20 [14] implementations present a throughput of 911 Mbps and 2,519 Mbps with an operating frequency of 19 MHz and 48 MHz, respectively. These frequencies are the same as those achieved by the implementations of ciphers based on chaotic key generators [4], [5]. However, these ciphers are still, in term of throughput less efficient than all the others implementations, as they have at 124 Mbps and 183 Mbps throughput.

A correlation between the speed performance and the hardware resources usage can be noted. In fact, the implementation of AES [1] which is the fastest is also the one using the most resources. This is also the case with the implementations of Rabbit [11] and Salsa20 [14], which use approximately 3,000 LUTs for a quite important throughput. Moreover, the implementations using the less resources are the Trivium's [12], [13] with only 100 and 376 used LUTs. The compact implementation of AES [2] has nearly the same results with 444 LUTs for the same speed performance.

For the implementations of the ciphers based on a chaotic generator [4], [5], this relation is not satisfied. Indeed, they use 2,718 and 1,926 LUTs, it achieves the same than Rabbit and Salsa20 implementations which have better speed performance.

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a hardware implementation of a new version of the LWCB SC on a FPGA platform. The new design can operate at a maximum frequency of 18.5MHz with a throughput of 565 Mbps. In addition it uses a reduced area of the platform of 2,363 LUTs.

The comparison with the state-of-the-start shows that the speed performance reached by the LWCB SC is below those of AES and the existing ciphers from the eSTREAM project. However, it is the same frequency and a better throughput than the ciphers based on a chaotic generator. The results in terms of resources usage reveal that the LWCB SC uses the same area as the state-of-the-art.

In future work, we will investigate some enhancements in order to improve the performance of the LWCB SC in terms of resources usage and speed performance. The synchronous pipeline of the system and a new design of the cell could be considered to improve its performance. Furthermore, tests against attacks such as CPA and DPA will be carried-out to validate the security of this implementation.

ACKNOWLEDGMENT

This work was funded by the Research Pole of the "Pôle d'Excellence Cyber" with the support of the French Ministry of the Armed Forces and the Brittany Region.

REFERENCES

[1] U. Farooq and M. F. Aslam, "Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA," Journal of King Saud University - Computer and Information Sciences, vol. 29, no. 3, Jul. 2017, pp. 295–302.

[2] P. Chodowiec and K. Gaj, "Very Compact FPGA Implementation of the AES Algorithm," in Cryptographic Hardware and Embedded Systems - CHES 2003. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, vol. 2779, pp. 319–333.

[3] C. De Canniere and B. Preneel, Trivium. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 244–266.

[4] C. Tanougast, "Hardware Implementation of Chaos Based Cipher: Design of Embedded Systems for Security Applications," in Chaos-Based Cryptography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, vol. 354, pp. 297–330. [Online]. Available: http://link.springer.com/10.1007/978-3-642-20542-2_9

[5] S. Sadoudi, C. Tanougast, M. S. Azzaz, A. Dandache, and A. Bouridane, "Real-time FPGA implementation of Lorenz's chaotic generator for cipher embedded systems," in 2009 International Symposium on Signals, Circuits and Systems. Iasi, Romania: IEEE, Jul. 2009, pp. 1–4.

[6] G. Gautier et al., "Enhanced Software Implementation of a Chaos-Based Stream Cipher," SECURWARE 2018.

[7] J. Fan et al., "State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures," in 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST). Anaheim, CA, USA: IEEE, Jun. 2010, pp. 76–87.

[8] C. Paar and J. Pelzl, Understanding cryptography: a textbook for students and practitioners. Heidelberg ; New York: Springer, 2010.

[9] M. Boesgaard, M. Vesterager, and E. Zenner, "The Rabbit Stream Cipher," in New Stream Cipher Designs. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 4986, pp. 69–83.

[10] D. J. Bernstein, "The Salsa20 Family of Stream Ciphers," in New Stream Cipher Designs. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 4986, pp. 84–97. [Online]. Available: http://link.springer.com/10.1007/978-3-540-68351-3_8

[11] D. Stefan, "Hardware Framework for the Rabbit Stream Cipher," in Information Security and Cryptology. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 6151, pp. 230–247.

[12] D. Hwang, M. Chaney, S. Karanam, N. Ton, and K. Gaj, "Comparison of FPGA-Targeted Hardware Implementations of eSTREAM Stream Cipher Candidates," p. 12.

[13] K. Gaj, G. Southern, R. Bachimanchi, and E. Department, "Comparison of hardware performance of selected Phase II eSTREAM candidates," p. 11.

[14] J. Sugier, "Implementing Salsa20 vs. AES and Serpent Ciphers in Popular-Grade FPGA Devices," in New Results in Dependability and Computer Systems. Heidelberg: Springer International Publishing, 2013, vol. 224, pp. 431–438.

 ISBN: 978-1-61208-743-6