Lightweight Hardware Design of a Chaos-Based Stream Cipher for Secure Video Applications

Guillaume Gautier

DOCTORAT BRETAGNE LOIRE / MATHSTIC

INSA RENNES

THÈSE DE DOCTORAT DE

L'INSTITUT NATIONAL DES SCIENCES APPLIQUEES DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Signal, Images, Vision*

Par

Guillaume GAUTIER

Lightweight Hardware Design of a Chaos-Based Stream Cipher for Secure Video Applications

Thèse présentée et soutenue à Rennes, le 09 décembre 2020
Unité de recherche : Institut d'Électronique et des Technologies du numéRique (IETR)
Thèse N° : 20ISAR 27 / D20 - 27

Rapporteurs avant soutenance :

Régis Leveugle	Professeur des Universités, Grenoble INP, Université Grenoble Alpes
René Lozi	Professeur des Universités, Université Nice Sophia Antipolis

Composition du Jury :

Président :	William Puech	Professeur des Universités, Université de Montpellier
Examinateurs :	Nathalie Bochard	Ingénieure de recherche, CNRS, Université de Saint-Etienne
	Wassim Hamidouche	Maître de conférences, INSA Rennes
	Régis Leveugle	Professeur des Universités, Grenoble INP, Université Grenoble Alpes
	René Lozi	Professeur des Universités, Université Nice Sophia Antipolis
Dir. de thèse :	Olivier Déforges	Professeur des Universités, INSA Rennes
Co-Dir. de thèse :	Safwan El Assad	Maître de conférences, HDR, Polytech Nantes

Table of Contents

Acknowledgements

Je tiens tout d'abord à remercier le Pole d'Excellence Cyber, la region Bretagne ainsi que la Direction Général de l'Armement pour avoir financé cette thèse.

Merci à l'ensemble des membres du jury d'avoir pris le temps d'évaluer mon travail de ses trois dernières années.

Je tiens à remercier Olivier Deforges, Safwan El Assad et Wassim Hamidouche de m'avoir fait confiance et encadré cette thèse. Merci de m'avoir accompagné au quotidien, et m'avoir permis de découvrir vos domaines de recherche respectifs.

Maguy, ton investissement lors de ton parcours recherche, puis de ton stage n'est pas passé inaperçu. Merci pour ton aide.

Je tiens également à remercier l'ensemble de mes collègues de l'équipe VAADER d'avoir rendu ce cadre de travail agréable et convivial. Secure IC, et plus particulièrement Sylvain Guilley, Robert Nguyen et Adrien Facon, vos conseils et votre aide ont contribué à la réalisation de ce document.

It is time to have some English to thank Naofumi Homma, Ville Yli-Mäyry and the rest of the team for welcoming me in Sendai to study Side Channel Attacks.

Ville, thank you for everything, your help was precious. Thanks for the making me discover Japan and its food and beverages. Hoping to see you again very soon.

Je tiens à remercier mes parents de m'avoir soutenu et encouragé pendant l'ensemble de mes études, elles arrivent enfin à leur terme.

Quentin, merci pour toutes ces discussions qui ne devaient pas durer longtemps et qui 1h30 après continuent toujours.

Arthur, tu vois, c'est faisable, il "suffit" juste de s'y mettre.

Justine, pour avoir réussi à me supporter pendant cette phase de rédaction.

CHAPTER 1

Introduction

1.1 Context

The need for secure communication between multiple parties practically always existed. At first, cryptography was only used by military and governments. Nowadays, this need became even more important with the apparition of more modern means of communication and the transition to an all-numeric environment. Secure communication channels are required even for civil applications such as bank applications, telecommuting, Video On Demand (VOD)... This diversification of use coupled with the multiplication of devices, architectures, the limited resources of embedded devices are adding more and more constraints to cipher design. Indeed, ciphers and its implementations need to be more robust as the computation capacity becomes cheaper, to adapt to the plurality of devices and its variety of architectures, and the limited resources such as energy, memory or computation power limitations.

Furthermore, during the past decade, the use of video applications over the network increased exponentially. VOD services such as Netflix or Amazon Prime for example, pay television networks need to protect its content to be watchable only to its subscribers. However, this protection should remain low-delay and low-complexity to keep achieving real-time decoding.

In this context, this thesis focuses to create a link between this two different fields of research, first by investigating the implementation of secure encryption scheme and then, by investigating how to protect video applications without creating a discomfort for the end user.

1.2 Objectives and Contributions of the Thesis

The main objectives of this thesis are the following:

1. Investigate the resilience against Side-Channel Attacks (SCAs) an existing chaos-based stream cipher.
2. Improve its resilience through modifications of the design and implementation of countermeasures like masking.
3. Propose a hardware design of the improved stream cipher.
4. Propose selective encryption method for video content without affecting the compression rate.

These four objectives are regrouped in three contributions. The first contribution is focusing on improving the chaos-based stream cipher proposed by Taha [1]. A method of attack, using 25 consecutive SCAs, is proposed to completely recover the secret key. From the proposed method, a new design of stream cipher is presented. This new design aims to counter, or at least increase the complexity of the attacks performed on the previous design.

The second contribution is to propose a secure and efficient lightweight hardware design of the newly proposed stream cipher. Several hardware implementations are proposed, the first is an optimized implementation of new stream cipher without masking or pipeline. This implementation is the reference design. Then, the second implementation is taking advantage of a pipeline to increase the throughput. The third and fourth implementations embed a masking operation on a part of the reference design as a countermeasure to SCAs.

The third contribution is a selective encryption scheme within the newest video compression standard Versatile Video Coding (VVC). This selective encryption scheme is developed respecting two major constraints: the solution must be Constant Bit Rate (CBR), the efficiency of the compression scheme must remain unchanged, and standard compliant, the encrypted video must be readable using any standard VVC decoder. For

comparison purposes with other selective encryption schemes, Advanced Encryption Standard (AES) is used instead of the stream cipher developed in the first two contributions. It is important to note that proposed solution is compatible with any stream ciphers.

1.3 Outline

This thesis is organized as follows. Part I is dedicated to the background and a short review of the state-of-the-art concerning three main topics:

1. cryptography in Section 2.1, with a presentation of the properties of **confidentiality**, **data integrity** and **authentication**. The difference between an **asymmetric** and a **symmetric** encryption scheme, its applications. Finally, a presentation of a few symmetric ciphers is proposed;
2. chaos theory in Section 2.2, the definition of a chaotic map and its applications to cryptography;
3. finally, in Section 2.3, a brief overview of exiting SCAs and countermeasures.

Part II focuses on the contributions of this thesis with:

1. first, after assessing the security of the stream cipher proposed by Taha [1], a new version of this cipher is proposed in Chapter 3;
2. Chapter 4 presents hardware designs of the proposed stream cipher with results in terms of speed and hardware resource usage. This chapter also includes a short study on the resilience of the designs against SCAs;
3. finally, in Chapter 5, a selective encryption scheme on the newest standard of video encoding VVC is presented.

This thesis is then concluded in Chapter 6 followed with some perspectives and future works.

Part I

Background

CHAPTER 2

Introduction to Chaos-Based Cryptography and Side-Channel Attacks (SCAs)

2.1 Introduction to Cryptography

Cryptography is used to provide a secure communication link between multiple parties. A secure communication link is established if the following three properties are met.
Confidentiality: This property ensures that the content of the communication is only intelligible for authorized parties. Confidentiality is usually achieved by encrypting the to-be-send data called *plaintext* using an encryption scheme, the output of this scheme is called *ciphertext*. The cipher text is then over the communication link and only authorized receivers should be able to recover the data using a decryption scheme.
Data integrity: This property ensures that received and decrypted cipher text remains unaltered through the communication process between the authorized parties. Data Integrity is usually performed using hash functions. A hash function is a function that generates a unique hash string for a given data. The hash function should also produce a completely different hash string for similar data. If two sets of data produce the same hash string then it can be assumed that the two sets are identical.
Authentication: This property ensures that received data is coming from the expected party.

2.1.1 Asymmetric Cryptography

Asymmetric or Public-Key cryptography is an encryption scheme based on an encryption key, also called a public key, and a decryption key called a private key. Anyone can encrypt data using a public key, however, only the associated private key can be used to decrypt the encrypted data. This type encryption scheme cannot be used to transfer huge payloads and require a good computational power. It is usually used establish for small-payload communication such as emails or to set up a secure communication channel using symmetric cryptography to transfer bigger payloads.

2.1.2 Symmetric Cryptography

In opposition to asymmetric cryptography where two different keys are used for encryption and decryption, symmetric cryptography uses the same key to perform encryption and decryption. The main advantage is the possibility to transfer big payloads using relatively lower computational power than asymmetric encryption scheme. However, the transfer of the secret key is the main drawback. The secret key being used to encrypt and decrypt the plaintext, needs to be sent to the other parties using secure channels and asymmetric cryptography can be used to exchange the keys. Among symmetric encryption schemes, two main categories can be distinguished: stream ciphers and block ciphers.

2.1.2.1 Stream cipher

A stream cipher is an encryption scheme where the plaintext P is XORed with the output of a Pseudo-Random Number Generator (PRNG) X_G to produce ciphertext C.

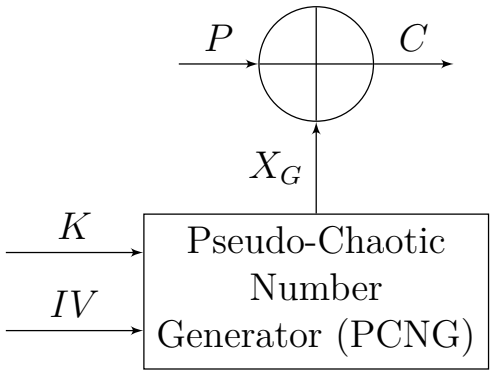


Figure 2.1 – Block diagram of a stream cipher.

Figure 2.1 represents the block diagram of a stream cipher where the plaintext P is ciphered using an PRNG initialized with a secret key K and an Initial Vector (IV). The main advantage of this kind of cipher is a possibility to cipher without delays, and the absence of minimal length of the plaintext. However only the value of the data is changed, the position remains unchanged.

2.1.2.2 Block cipher

A block cipher encrypts fixed-length blocks of plaintext one by one by performing a confusion and a diffusion operation.

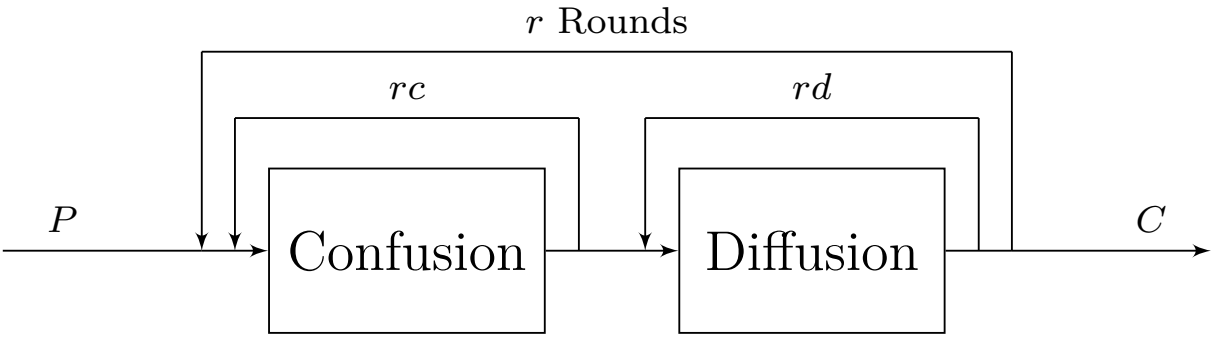


Figure 2.2 – Block diagram of a block cipher.

Figure 2.2 represents a general block diagram of a block cipher. The block of plaintext goes through the confusion and the diffusion, this operation can be repeated on the same block for a given number of rounds. It is important to note that either the confusion or the diffusion is non-linear.

Confusion The confusion is the operation that changes the value of the data inside the block. This operation can be performed using Substitution Boxes (S-Boxes).

Diffusion The diffusion is the operation that changes the position of the value inside the block. Shifting and shuffling rows of the block, for example, is a method of diffusion.

When encrypting more than one block, the block cipher can be used in multiple modes of operation. The most famous ones are Electronic Code Book (ECB) and Cipher Block Chaining (CBC) to perform a block-based encryption and Cipher FeedBack (CFB), Output FeedBack (OFB) and CounTeR (CTR) to use the block cipher as PRNG and create a stream cipher.

2.1.2.2.1 Electronic Code Book (ECB) mode Figure 2.3 represents how the ECB mode of operation is working. In this mode, the plaintext is organized in blocks and each block is encrypted independently. It is the simplest mode, fully parallelizable , and in case of transmission errors, the following blocks are still decryptable. However, ECB is sensitive to chosen-plaintext attack and attack by repetition. The ciphertext is the same for a given key and plaintext, therefore, the produced cipher can be reused as is to cheat the receiver.

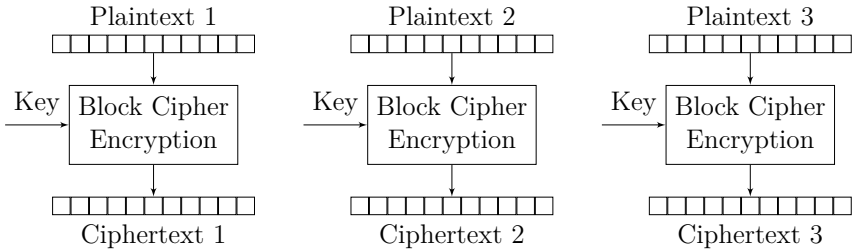


Figure 2.3 – ECB block cipher mode of operation

2.1.2.2.2 Cipher Block Chaining (CBC) mode Figure 2.4 depicts the CBC mode of operation. The first block of plaintext is XORed with a IV block. Then the output of the XOR operation is put through the encryption block producing the first cipher block. This cipher block is then reused and XORed to the next block of plaintext and the process repeat itself. This is the most used mode and is more resilient to attacks compared to ECB. However, the previous ciphered block is required to encrypt the current block, making this mode, not parallelizable and undecryptable if a block is lost in the middle of a transmission.

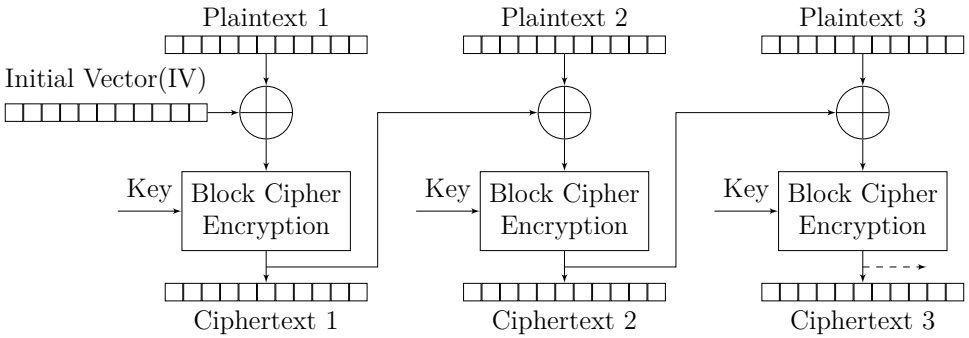


Figure 2.4 – CBC block cipher mode of operation

2.1.2.2.3 Cipher FeedBack (CFB) mode Figure 2.5 shows the first mode to use a block cipher as a stream cipher, the CFB. In these kinds of modes, the plaintext is XORed after the encryption block. The first block used is an IV. The output of the encryption block is then XORed to the plaintext to produce the ciphertext. The cipher text is then reused IV for encrypting the next block of plaintext. However, this mode of operation is sensitive to chosen-plaintext attacks.

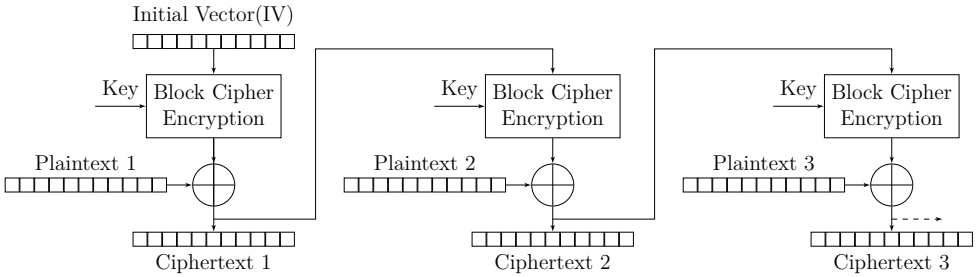


Figure 2.5 – CFB block cipher mode of operation

2.1.2.2.4 Output FeedBack (OFB) mode Figure 2.6 illustrates the OFB mode. This one is similar to the CFB mode, the only change the reused block used IV is the block at the output of the encryption block. This mode of operation is also sensitive to chosen-plaintext attacks.

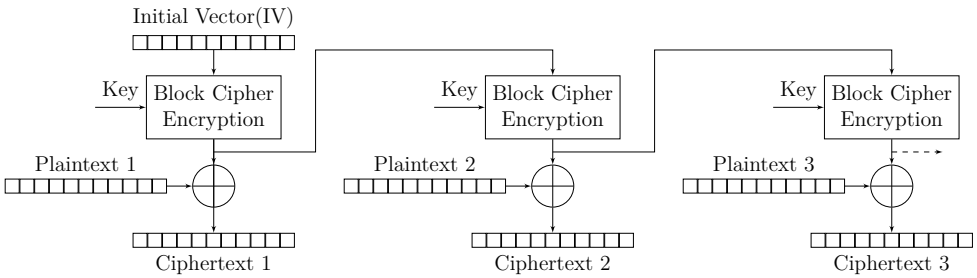


Figure 2.6 – OFB block cipher mode of operation

2.1.2.2.5 CounTeR (CTR) mode Figure 2.7 represents CTR mode, the most commonly used block cipher mode to create a stream cipher. The block used by the encryption is counter value. The result of the encryption of the counter value is then XORed to the plaintext block. For the next plaintext block, the value of the counter is incremented, and the process is repeated.

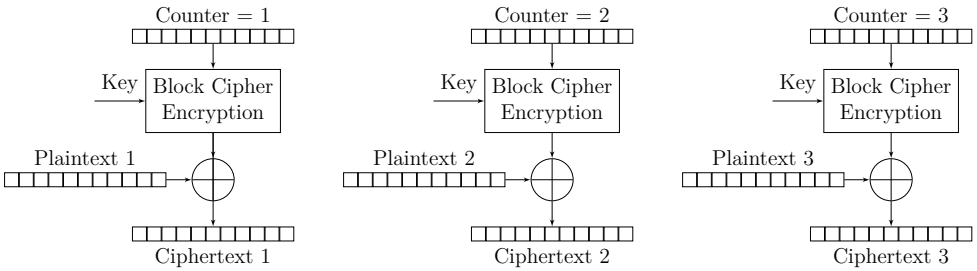


Figure 2.7 – CTR block cipher mode of operation

2.1.2.3 Existing non-chaotic cipher

Data Encryption Standard (DES) [2] is a block cipher which encrypts block of length of 64 bits with a 64-bit key. It has been by far the most popular block cipher for the last 30 years but has been now replaced by Advanced Encryption Standard (AES) because it is considered not secure against a determined attacker due to the fact that the DES key space of 64 bits is too small. The DES has a structure called Feistel network which is described as: it firstly does a bitwise permutation for the 64-bit plaintext x with an initial 64 bits Initial Permutation (IP) and each round performs an identical operation.

Advanced Encryption Standard (AES) [2] is the most widely used symmetric cipher today, which processes data in 128-bit blocks. AES is almost identical to the block cipher Rijndael. The key size may take values such as 128, 192 and 256 bits executing between 10 and 14 rounds depending on the key size. Its encryption round function consists of three layers including key addition layer, byte substitution layer called S-Box and diffusion layer. Its decryption round inverts the iterations resulting in a partially different data path.

Rabbit [3] stream cipher is one of the most effective algorithms of the eSTREAM Project [4]. It was designed for software and hardware applications. Its aim is to be faster than commonly used ciphers. Rabbit does not have an S-Box. Instead, it has an internal state that consists of 8 32-bit state variables, 8 32-bit counters and one counter carry bit. The eight counters are updated every time before each iteration of the internal system. In addition, the eight state variables are updated in the iteration of eight coupled nonlinear functions. All state variables only depend on their corresponding counters and the previous state variables. The 128-bit keystream is generated using the state variables. The encryption is done by XOR operation between the keystream and the plaintext.

HC-128 [5] is an efficient software stream cipher. It is constructed on two secret tables of 512 32-bit elements each. At each step, it updates one element of a table with a non-linear feedback function. All the elements of each table is updated every 1024 steps.

In addition, it is suitable for modern and future super-scalar microprocessors. Indeed, three consecutive steps can be computed in parallel, as feedback and output functions. We can say that this cipher can be implemented with a high degree of parallelism, which is a good feature to be run efficiently on modern processors.

Trivium [6] is particularly well suited for applications requiring a flexible hardware implementation. It is a synchronous stream cipher. It was designed to explore how far a stream cipher can be simplified without sacrificing its security, speed or flexibility.

It generates 2^{64} key stream from an 80-bit secret key and an 80-bit IV. The internal structure of Trivium is composed of a 288-bit internal state. It is stored in three shift registers of different lengths (93, 84 and 111) and is the heart of Trivium. The XOR-sum of all three registers outputs forms the key stream. In addition, the output of each register is connected to the input of another register. Trivium can be viewed as one circular register with a length of 288 bits.

2.2 Chaos Theory and its Application to Cryptography

Chaos refers in mathematics to non-linear dynamic systems with a behaviour that appears completely random but is deterministic. A chaotic system is also highly sensitive to initial conditions, topologically transitive and has a dense periodic orbit [7].

2.2.1 Chaotic Maps

A chaotic map is a discrete-time chaotic system. It is usually defined by a function from a domain to the same domain appearing completely unpredictable [8]. This chaotic map can be multidimensional. However 1-D maps are the one commonly used in cryptography. The attractor is a representation in which the samples are displayed function of the previous one. The dynamic system tends to evolve into this state. A fixed point is a state in which the dynamic system is no longer evolving. For a dynamic system $x_{n+1} = f(x_n)$, a fixed point exists when $x_i = f(x_i)$. In literature multiple chaotic maps were defined such

as the 2D-Baker map [9], the Gauss map [10], the Henon map [11], the Lorenz map [12], the Lozi map [13] or the Tent map [14].

2.2.1.1 Skew Tent (ST) Map

The ST map [15] is defined in the literature as $f : [0,1] \to [0,1]$ given by

$$f(x,p) = \begin{cases} \frac{x}{p} & \text{if } 0 \leq x \leq p, \\ \frac{1-x}{1-p} & \text{if } p < x \leq 1, \end{cases} \tag{2.1}$$

where $p \in]0,1[$ is the parameter of the ST map. Figure 2.8 represents the mapping of the map with parameter p set to 0.75. The ST map is used in multiple encryption scheme [16, 17, 18].

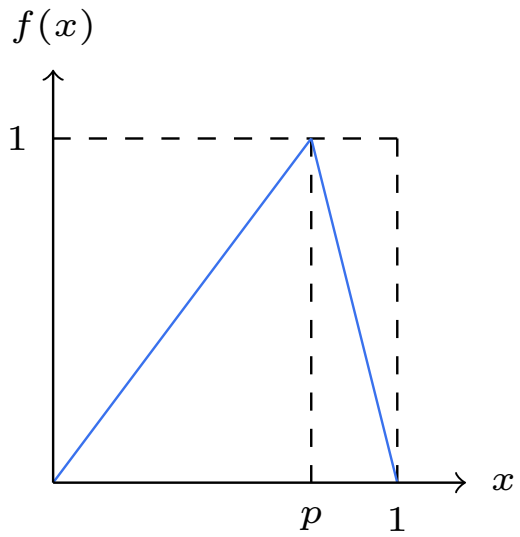


Figure 2.8 – Mapping of the ST map, with $p = 0.75$

2.2.1.2 PieceWise Linear Chaotic (PWLC) Map

The PWLC map [19] is defined in the literature as $f : [0,1] \to [0,1]$ given by

$$f(x,p) = \begin{cases} \frac{x}{p} & \text{if } 0 \leq x \leq p, \\ \frac{1-x}{1-p} & \text{if } p < x \leq \frac{1}{2}, \\ f(1-x,p) & \text{if } \frac{1}{2} < x \leq 1, \end{cases} \tag{2.2}$$

where $p \in]0, \frac{1}{2}[$ is the parameter of the PWLC. The mapping of the PWLC with the parameter $p = 0.2$ is represented in Figure 2.9. This map is also used in several chaos-based encryption schemes [20, 21]

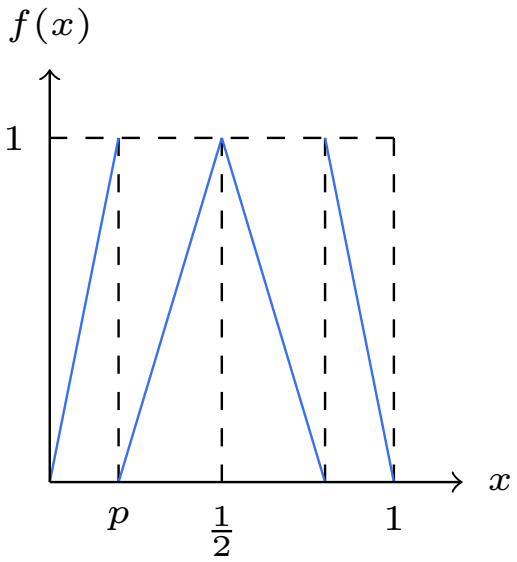


Figure 2.9 – Mapping of the PWLC map, with $p = 0.2$

2.2.2 Existing Chaos-Based Ciphers

In the past decade, many chaos-based cryptographic techniques have been proposed. In 1997, Fridrich [22] introduced a chaos-based block encryption scheme and a structure [23] used as the core structure of most chaos-based block cryptosystems. The Fridrich structure in Figure 2.10 is a traditional confusion-diffusion architecture. It encrypts the plaintext P by a round operation consisted of a 2-D Baker chaotic map confusion layer and a fixed random permutation diffusion layer. The cipher will repeat the same operation for r

rounds and outputs the ciphertext C. However in [24], the authors analysed the security of the Fridrich scheme and highlighted some weaknesses and proposed improvements to overcome these security failures. Thus, Yang *et al.* [25] propose a new version of Fridrich scheme where the designer adds a Pseudo-Chaotic Number Generator (PCNG) to generate key for the encryption operations. The confusion layer and diffusion layer in the previous system can be replaced by other efficient mathematics functions.

For example, Barakat *et al.* [26] developed a block cipher for image encryption based on Lorenz Chaotic Generator. The Lorenz's Chaotic Generator [27], based on Lorenz's differential equation system, can be used as a hard-key generator in a chaotic self-synchronizing cipher encryption. Bensikaddour *et al.* [28] also proposed a chaos-based block cipher with the new Fridrich structure for image encryption. The designer takes a PWLC map as key generator map, an Arnold's Cat map as confusion layer and the enhanced function in [24] as the diffusion layer.

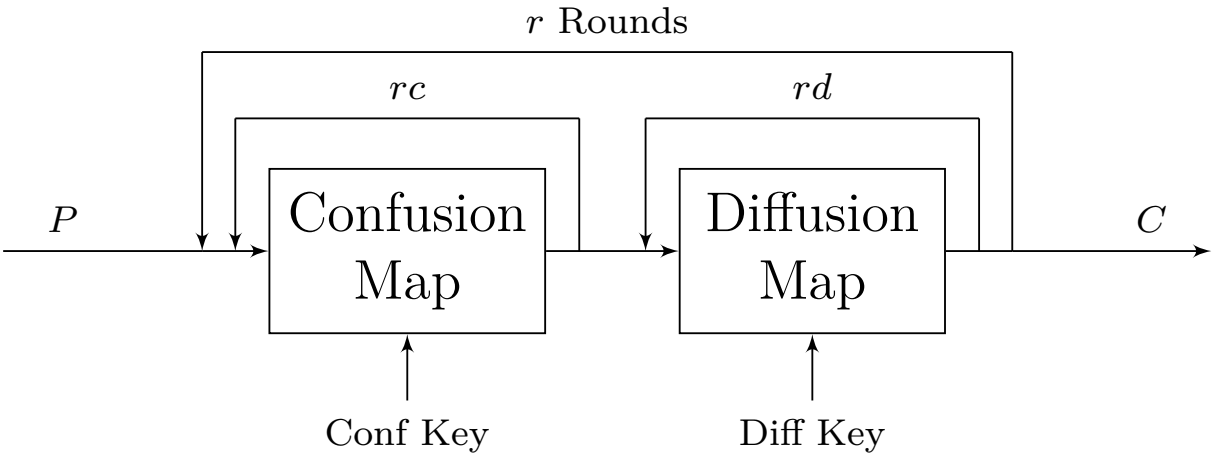


Figure 2.10 – Fridrich image encryption structure

2.3 Physical Cryptanalysis

Cryptanalysis is the study of cryptographic systems in order to find weaknesses to either find the secret key, retrieve the plaintext from the ciphertext or reduce the space for brute-force attacks. Cryptanalysis can be divided into two main categories.

1. Mathematical cryptanalysis studies the algorithm to find vulnerabilities. Most recent encryption algorithms are generally secure against that type of cryptanalysis.

2. Physical cryptanalysis, on the other hand, studies the implementation of the algorithm. An algorithm can be mathematically secure but its implementation is physically leaking information.

Among physical cryptanalysis, attacks are classified into two types, Fault Attacks (FAs) or active attacks and SCAs or passive attacks.

2.3.1 Fault Attacks (FAs)

FAs are attacks aiming to create errors into the system by changing the execution environment. Bar-El *et al.* [29] present multiple fault injection methods. For example, creating a variation in the power supply of the device [30]. By changing the voltage outside the requirement for a short amount of time, the system may enter in a faulty behaviour and skip or misinterpret instructions. This type of fault injection is cheap. However it is not a precise fault injection. In the same way, glitching the clock signal [30] is one of the simplest and cheapest to operate. Glitching the clock will allow to create a shorter clock cycle, the computation of the device will not be complete and error will be created.

Electromagnetic fault injection [31, 32] is another common method of injection. In this method, a probe is used to perturb the external electromagnetic field of the attack device and possibly impact the transistors and the memory blocks.

Laser fault injection [33] is one of the most expensive techniques, however, it is one of the most precise. With laser injection it is possible to change the state of the system temporarily or permanently.

The fault injection technique needed depends on the fault model required. Depending on whether the attack requires to precisely flip one bit a time then laser injection should be used. However, to produce random faults, a simple power variation or clock glitches can be enough.

2.3.2 Side-Channel Attacks (SCAs)

Oppositely to FAs, SCAs are passive physical attacks. The attacker is not tampering the system while performing an attack. The SCAs are based on exploiting leakage from external physical measurements of the system such as execution time, power consumption, or electromagnetic emissions, for example.

2.3.2.1 Channels

2.3.2.1.1 Time Attacks introduced by Kocher [34], exploits execution time differences in algorithms in order to recover the secret. The most famous example of timing attacks is on RSA implementations.

Algorithm 2.1: RSA square and multiply algorithm.

```
x ← C
for j = 1 to n do
  x ← x^2 mod N
  if d_j == 1 then
    x ← xC mod N
  end if
end for
return x
```

In Algorithm 2.1, the RSA square-and-multiply algorithm is presented. Depending on the secret key bit d_j, it will execute just a square operation when the key bit equal 0 and execute a square operation followed multiply operation when the key bit equals 1. A square operation takes significantly less time than a square-multiply operation. Therefore, when a short iteration execution time occurs, the key bit d_j is equal to 0, and 1 when for a longer execution time.

2.3.2.1.2 Power Attacks exploit the dynamic power model and how CMOS cells work. Each time a cell transition from $0 \rightarrow 1$ or $1 \rightarrow 0$ occurs, for a brief moment, the two CMOS transistors are saturated creating a small spike in the power consumption. Therefore, from the analysis of the power consumption, information about processed data can be recovered using an appropriate leakage model and distinguisher.

2.3.2.1.3 Electromagnetic Attacks are similar to power attacks. The main difference lies in the method of leakage collection. The advantage of electromagnetic attacks compared to power attacks is the possibility to collect leakage on a smaller and more localized part of the system to get rid of noise generated by other elements of the device.

2.3.2.2 Leakage Model

The leakage model is one of the two requirements to perform a power/electromagnetic attack. The leakage model is the model used to estimate the power consumption or elec-

tromagnetic emission of the device under attack. In the literature, mainly two different leakage models are used, the Hamming Distance (HD) and the Hamming Weight (HW).

2.3.2.2.1 Hamming Distance (HD) leakage model is the model used to estimate the consumption of a register transitioning from a value different from 0 to another. In this case, the consumption $C(t)$ of a register can be estimated as follows:

$$C(t) = \sum_{i=0}^{n} H(A_i \oplus B_i) + \mathcal{B}(t), \tag{2.3}$$

where A_i is the i-th bit of the initial value of the register of size n, B_i is the i-th bit of the register after the load, $H(0)$ correspond to the idle consumption, $H(1)$ correspond to the consumption of a transition $\mathcal{B}(t)$ is the noise at an instant t. This noise is usually assumed to be Gaussian. This model also assumes that a transition from $0 \rightarrow 1$ or $1 \rightarrow 0$ consumes the same amount of power.

2.3.2.2.2 Hamming Weight (HW) leakage model is the model used to estimate the consumption of a register transitioning from 0 to another value. In this case, the consumption $C(t)$ of a register can be estimated as follows:

$$C(t) = \sum_{i=0}^{n} H(B_i) + \mathcal{B}(t). \tag{2.4}$$

2.3.2.3 Distinguisher

The distinguisher is a method to expose dependency between the physical measurement and a leakage model. The quality of a distinguisher is determined by its capacity to highlight the correct key hypothesis from the others using the minimum number of traces.

2.3.2.3.1 Simple Power Analysis (SPA) SPA is a distinguisher aiming to recover the secret key from only a few power consumption traces. Kocher [34] presents an attack on the naive RSA implementation. This distinguisher is not only used to recover the key directly it is often used to prepare for a more complex attack. The countermeasure against SPA is to develop a constant-time implementation of the algorithm. A constant-time implementation of an algorithm in an implementation where secret-dependent operations take exactly the same amount of time for any given key.

2.3.2.3.2 Differential Power Analysis (DPA) Kosher *et al.* [35] introduced the DPA distinguisher and was applied only to single-bit variables. Later, the distinguisher was extended to support multi-bit variables. This extended DPA is computed as the covariance between the leakage l and the leakage model h:

$$DPA : \Delta = Cov(l, h). \tag{2.5}$$

For the wrong key hypothesis, the difference trace is close to 0. On the other hand, when the hypothesis is correct, the differential trace shows a peak.

2.3.2.3.3 Correlation Power Analysis (CPA) Brier *et al.* [36] proposed a normalized version of DPA. The CPA distinguisher is less sensitive to noise and consequently is more precise. CPA computes the correlation coefficient ρ between the leakage l and the leakage model h:

$$CPA : \rho = \frac{Cov(l, h)}{\sigma_l \, \sigma_h}, \tag{2.6}$$

where σ_l and σ_h are the standard deviation of the leakage l and the leakage model h, respectively.

2.3.2.4 Countermeasures against Side-Channel Attacks (SCAs)

2.3.2.4.1 Data Masking consists of combining, at each execution, sensible data with a different random value, called a mask. Masked data are then processed through the system. Before outputting the result, the mask is removed [37, 38, 39]. Masking can be either Boolean or arithmetic. The type of masking depends on the operations performed the masked algorithm. Boolean masking masks the value using a XOR operation between the value v and the mask m,

$$v_m = v \oplus m. \tag{2.7}$$

Arithmetic masking is performed using an arithmetic operation and a modulo. For example, an additive masking of a value on n bits is computed as follows;

$$v_m = (v + m) \mod n. \tag{2.8}$$

Masking can be easily implemented on any linear operation γ respecting the following equality:

$$f(v \, \gamma \, m) = f(v) \, \gamma \, f(m). \tag{2.9}$$

However, for non-linear function, it can't be recovered mathematically and it is required to find solutions case by case. For S-Boxes, the most common non-linear part in encryption schemes, the topic of masking was vastly studied [40, 41, 42].

2.3.2.4.2 Constant Power Consumption This countermeasure consists of hiding the leakage inside a constant power usage. Exploiting the fact that hardware device uses CMOS cells and its high consumption when transitioning, each clock cycle must have the same amount of transition are occurring.

Part II

Contributions

CHAPTER 3

Enhanced Software Implementation of a Chaos-Based Stream Cipher

Stream ciphers are commonly used to encrypt data in real-time applications like, for example, in selective video encryption [43, 44]. It consists of performing an eXclusive OR (XOR) operation between a plain text and the output of a deterministic random generator. In the literature, multiple stream ciphers exist, the eSTREAM project was promoting the design of efficient and compact stream ciphers such as HC-128 [5] or Rabbit [3], but according to [45], eSTREAM ciphers are not all secured.

The chaos theory is used in cryptography for its natural property of deterministic randomness. Indeed, chaos-based ciphers generally use chaotic maps for their combination of security and relatively low complexity.

This section's objective is to assess and improve the security of the stream cipher proposed by Taha [1]. After a presentation of this method, an Side-Channel Attack (SCA) analysis is carried out to assess the vulnerabilities of the design. Then, a new design is proposed including countermeasures at the design and implementation levels. Finally, a performance analysis of the new design is proposed.

3.1 Original Stream Cipher

Taha [1] proposed a stream cipher based on the Pseudo-Chaotic Number Generator (PCNG) using a $N = 32$ bit precision, and represented in Figure 3.1. The PCNG is composed of two cells, depicted in orange, that use different chaotic maps called Skew Tent (ST) map and PieceWise Linear Chaotic (PWLC) map. The ST map is defined by Equation (3.1) and the PWLC map is defined by Equation (3.2). The output of the maps are periodically perturbed using a XOR operation with a Linear Feedback Shift Register (LFSR). The outputs of the two cells are fed back to the chaotic maps after being filtered using an Infinite Impulse Response (IIR) filter, depicted in green in Figure 3.1. The outputs of the cells are also XORed to generate the key stream X_G after reaching the end of the transient state (i.e. after tr turns).

$$X_S(n) = STmap(X_S(n-1), P_S) = \begin{cases} \left\lfloor 2^N \times \frac{X_S}{P_S} \right\rfloor & \text{if } 0 < X_S < P_S \\ \left\lfloor 2^N \times \frac{2^N - X_S}{2^N - P_S} \right\rfloor & \text{if } P_S < X_S < 2^N \\ 2^N - 1 & \text{otherwise} \end{cases} \tag{3.1}$$

$$X_P(n) = PLWCmap(X_P(n-1), P_P) = \begin{cases} \left\lfloor 2^N \times \frac{X_P}{P_P} \right\rfloor & \text{if } 0 < X_P < P_P \\ \left\lfloor 2^N \times \frac{X_P - P}{2^{N-1} - P_P} \right\rfloor & \text{if } P_P < X_P < 2^{N-1} \\ \left\lfloor 2^N \times \frac{2^N - P_P - X_P}{2^{N-1} - P_P} \right\rfloor & \text{if } 2^{N-1} < X_P < 2^N - P_P \\ \left\lfloor 2^N \times \frac{2^N - X_P}{P_P} \right\rfloor & \text{if } 2^N - P_P < X_P < 2^N \\ 2^N - 1 & \text{otherwise} \end{cases} \tag{3.2}$$

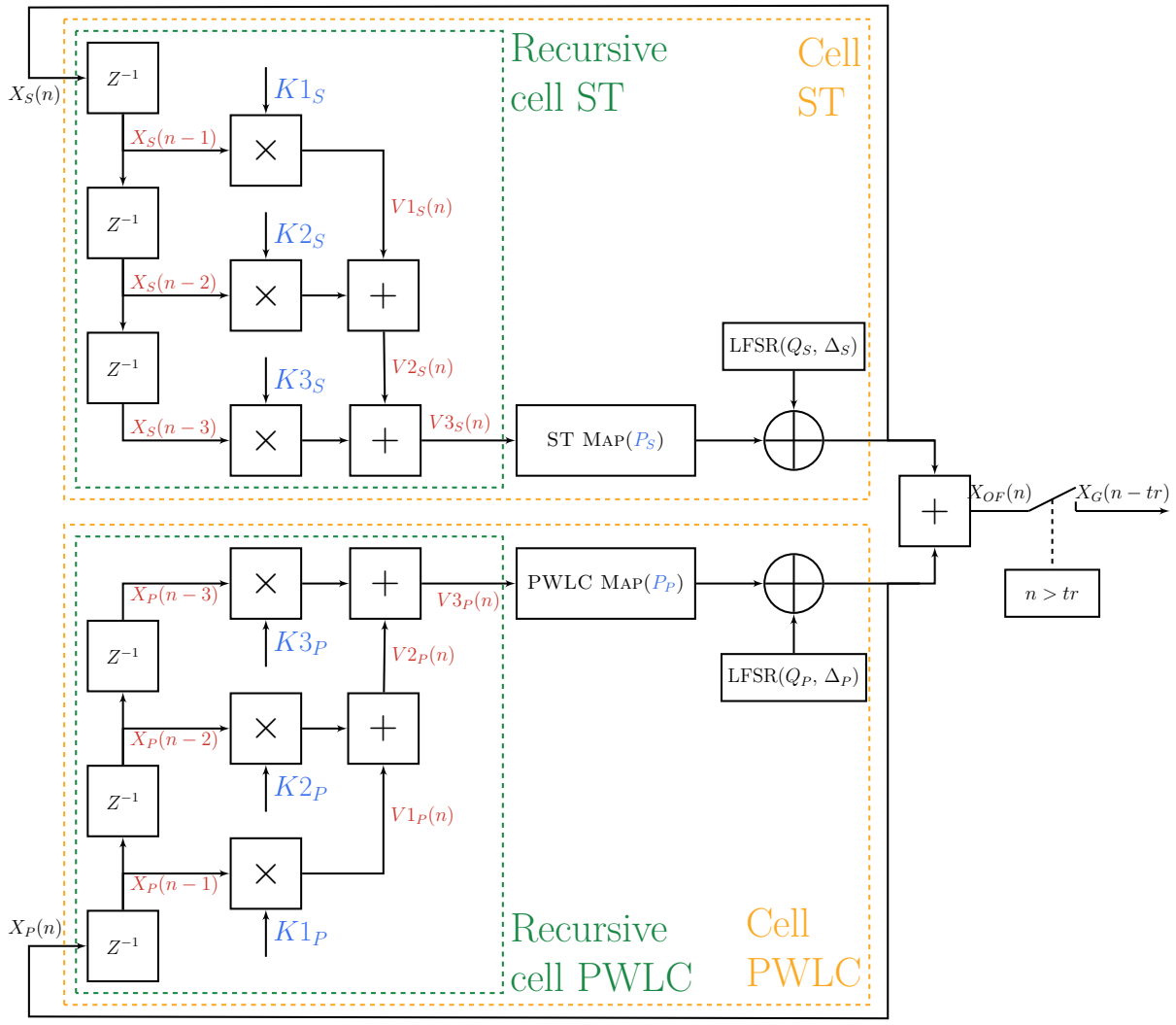


Figure 3.1 – Taha [1] PCNG diagram. In green are highlighted the recursive cells. Orange dotted line delimits the two cells.

The Initial Vector (IV) of the PCNG is a $2N$-bit-long word used as two separate N-bit-long sub-IV, one for each cell, IV_S for the ST map and IV_P for the PWLC map. The two sub-IVs are added to the input of each chaotic map for the first iteration of the system resulting in the following equation:

$$X_{\mathrm{IN}MP}(n) = \begin{cases} IV_{MP} + \sum_{i=1}^{3} Ki_{MP}\, X_{MP}(n-i) & \text{if } n = 0 \\ \sum_{i=1}^{3} Ki_{MP}\, X_{MP}(n-i) & \text{otherwise} \end{cases}, \tag{3.3}$$

where MP is set to S for the ST map and P for the PWLC map. $X\text{IN}_{MP}(n)$ is the input of the map MP at iteration n, Ki_{MP} are the coefficients of the MP map filter and $X_{MP}(n-i)$ are the delayed sample of the filter.

Table 3.1 lists all the elements composing the secret key of the PCNG. The parameter $P_S \in]0, 2^{32}[$ and the coefficients of the IIR filter $K_S(1),\ K_S(2),\ K_S(3) \in]0, 2^{32}[$, the initial delayed values $X_S(-1),\ X_S(-2),\ X_S(-3) \in]0, 2^{32}[$ are part of the secret key, for PWLC, the parameter $P_P \in]0, 2^{31}[$, $K_P(1),\ K_P(2),\ K_P(3) \in]0, 2^{32}[$ and $X_P(-1),\ X_P(-2),\ X_P(-3) \in]0, 2^{32}[$, respectively. The duration of the transient state $tr \in]0, 256[$ is also part of the key, putting the length of the secret key to 455-bit long.

Table 3.1 – Key composition of the original stream cipher.

Name	Length(in bits)	Description
P_S	32	Parameter of the ST map
$K1_S$	32	First coefficient of the ST map recursive cell
$K2_S$	32	Second coefficient of the ST map recursive cell
$K3_S$	32	Third coefficient of the ST map recursive cell
$X_S(-1)$	32	Initial value of the first delay of the ST map recursive cell
$X_S(-2)$	32	Initial value of the second delay of the ST map recursive cell
$X_S(-3)$	32	Initial value of the third delay of the ST map recursive cell
P_P	31	Parameter of the PWLC map
$K1_P$	32	First coefficient of the PWLC map recursive cell
$K2_P$	32	Second coefficient of the PWLC map recursive cell
$K3_P$	32	Third coefficient of the PWLC map recursive cell
$X_P(-1)$	32	Initial value of the first delay of the PWLC map recursive cell
$X_P(-2)$	32	Initial value of the second delay of the PWLC map recursive cell
$X_P(-3)$	32	Initial value of the third delay of the PWLC map recursive cell
tr	8	Number of samples to reach before outputting the first sample(ie. $X_G(0) = X_{OF}(tr)$)
Total	455	

3.2 Side Channel analysis of studied stream cipher

This section presents possible attacks on the previously presented PCNG. The attacks are all SCA-based attacks. Table 3.2 lists the 25 attacks to recover the full 455-bit key of the PCNG.

Table 3.2 – List of the attacks used to break PCNG.

Attacks	Targeted Value	Extracted Value	Prerequisite	Method	Nb run
AT01	$V1_S(0)$	$V1_S(0), PK1_S$	–	SCA	1
AT02	$V2_S(0)$	$V2_S(0), PK2_S$	$PK1_S$	SCA	1
AT03	$V3_S(0)$	$V3_S(0), PK3_S$	$PK1_S$ and $PK2_S$	SCA	1
AT04	$V1_P(0)$	$V1_P(0), PK1_P$	–	SCA	1
AT05	$V2_P(0)$	$V2_P(0), PK2_P$	$PK1_P$	SCA	1
AT06	$V3_P(0)$	$V3_P(0), PK3_P$	$PK1_P$ and $PK2_P$	SCA	1
AT07	$X_S(n)$	P_S	$PKi_S(i = 1, 2, 3)$	Timing Attack	1
AT08	$X_P(n)$	P_P	$PKi_P(i = 1, 2, 3)$	Timing Attack	1
AT09	$V1_S(1)$	$K1_S$	$P_S, V3_S(0)$	SCA Mult.	2
AT10	$PK1_S$	$X_S(-1)$	$K1_S, PK1_S$	Extented Euclidian	2
AT11	$V2_S(1) = K1_S\, X_S(0) + K2_S\, X_S(-1)$	$K2_S\, X_S(-1)$	$P_S, V3_S(0), K1_S$	SCA	2
AT12	$K2_S\, X_S(-1)$	$K2_S$	$K2_S\, X_S(-1)$, $X_S(-1)$	Extented Euclidian	2
AT13	$PK2_S$	$X_S(-1)$	$K2_S, PK2_S$	Extented Euclidian	2
AT14	$V3_S(1) = K1_S\, X_S(0) + K2_S\, X_S(-1) + K3_S\, X_S(-2)$	$K3_S\, X_S(-2)$	$P_S, V3_S(0), K1_S$, $K2_S\, X_S(-1)$	SCA	2
AT15	$K3_S\, X_S(-1)$	$K3_S$	$K3_S\, X_S(-2)$, $X_S(-2)$	Extented Euclidian	2
AT16	$PK3_S$	$X3_S$	$K3_S, PK3_S$	Extented Euclidian	2
AT17	$V1_P(1)$	$K1_P$	$P_P, V3_P(0)$	SCA Mult.	2
AT18	$PK1_P$	$X_P(-1)$	$K1_P, PK1_P$	Extented Euclidian	2
AT19	$V2_P(1) = K1_P\, X_P(0) + K2_P\, X_P(-1)$	$K2_P\, X_P(-1)$	$P_P, V3_P(0), K1_P$	SCA	2
AT20	$K2_P\, X_P(-1)$	$K2_P$	$K2_P\, X_P(-1)$, $X_P(-1)$	Extented Euclidian	2
AT21	$PK2_P$	$X_P(-2)$	$K2_P, PK2_P$	Extented Euclidian	2
AT22	$V3_S(1) = K1_P\, X_P(0) + K2_P\, X_P(-1) + K3_P\, X_P(-2)$	$K3_P\, X_P(-2)$	$P_P, V3_P(0), K1_P$, $K2_P\, X_P(-1)$	SCA	2
AT23	$K3_P\, X_P(-2)$	$K3_P$	$K3_P\, X_P(-2)$, $X_P(-2)$	Extented Euclidian	2
AT24	$PK3_P$	$X3_P$	$K3_P, PK3_P$	Extented Euclidian	2
AT25	execution time	tr	Data output pattern	Timing Attack	tr

3.2.1 AT01 → AT06: Extraction of PKi_S and PKi_P, products of recursive cells

Let PKi_S be $PKi_S = X_S(-i) \times Ki_S$ and PKi_P be $PKi_P = X_P(-i) \times Ki_P$ with $i \in \{1;2;3\}$. To recover the values of the products PKi_S and PKi_P inside the recursive cells, series of six SCA using a Correlation Power Analysis (CPA) distinguisher on Hamming Weight (HW) leak pattern are used to recover the intermediary results $V1_S(0)$, $V2_S(0)$, $V3_S(0)$, $V1_P(0)$, $V2_P(0)$, and $V3_P(0)$ represented in red in Figure 3.1. The PKi_S can be expressed as a function of the intermediary results as follows:

$$\begin{aligned} V1_S(0) &= X_S(-1) \times K1_S \Rightarrow \boldsymbol{PK1_S = V1_S(0)}, \\ V2_S(0) &= V1_S(0) + X_S(-2) \times K2_S \\ &= V1_S(0) + PK2_S \Rightarrow \boldsymbol{PK2_S = V2_S(0) - V1_S(0)}, \\ V3_S(0) &= V2_S(0) + X_S(-3) \times K3_S \\ &= V2_S(0) + PK3_S \Rightarrow \boldsymbol{PK3_S = V3_S(0) - V2_S(0)}. \end{aligned} \tag{3.4}$$

Similarly, the PKi_P are expressed as follows:

$$\begin{aligned} V1_P(0) &= X_P(-1) \times K1_P \Rightarrow \boldsymbol{PK1_P = V1_P(0)}, \\ V2_P(0) &= V1_P(0) + X_P(-2) \times K2_P \\ &= V1_P(0) + PK2_P \Rightarrow \boldsymbol{PK2_P = V2_P(0) - V1_P(0)}, \\ V3_P(0) &= V2_P(0) + X_P(-3) \times K3_P \\ &= V2_P(0) + PK3_P \Rightarrow \boldsymbol{PK3_P = V3_P(0) - V2_P(0)}. \end{aligned} \tag{3.5}$$

3.2.2 AT07 → AT08: P_S and P_P, parameters of the chaotic maps

The chaotic maps used in the system are characterized by a parameter P_S for the Skew Tent and P_P for PWLC. In the implementation, the execution time of the map depends on the input given to the map.

According to Equation (3.3), for the first iteration, the output of the Skew Tent map is equal to $X_S(0) = STmap(IV_S + V3_S(0), P_S)$. In Algorithm 3.1, the implementation shows that when $X_{IN_S} \in \{0; P_S\}$ the failsafe branch is executed. So, for the first iteration, $IV_S + V3_S(0) = P_S$ when the execution time is the shortest. Knowing $V3_S(0)$, recovered in the previous attacks, and by measuring the execution time while changing the value of IV_S, P_S can be recovered.

Algorithm 3.1: Calculate $X_S(n) = STmap(X\text{IN}_S(n), P_S)$.

Require: $X\text{IN}_S \in]0; 2^{32}[$ and $P_S \in]0; 2^{32}[$
  if $0 < X\text{IN}_S < P_S$ **then**
    $X_S \leftarrow X\text{IN}_S \times ratio1$
  else if $P_S < X\text{IN}_S < M_1$ **then**
    $X_S \leftarrow (M_1 - X\text{IN}_S) \times ratio2$
  else
    $X_S \leftarrow M_1 - 1$
  end if
  return X_S

where $M_1 = 2^{32}$, $M_2 = 2^{31}$ and ratios are precomputed floats.

Algorithm 3.2: Calculate $X_p(n) = PLWCmap(X_p(n-1), P_p)$.

Require: $X\text{IN}_P \in]0; 2^{32}[$ and $P_p \in]0; 2^{31}[$
  if $0 < X\text{IN}_P < P_p$ **then**
    $X_P \leftarrow X\text{IN}_P \times C_1$
  else if $(P_p < X\text{IN}_P < M_2)$ **then**
    $X_P \leftarrow (X\text{IN}_P - P_p) \times C_2$
  else if $M_2 < X\text{IN}_P < (M_1 - P_p)$ **then**
    $_P \leftarrow (M_1 - P_p - X\text{IN}_P) \times C_2$
  else if $(M_1 - P_p) < X\text{IN}_P < M_1$ **then**
    $X_P \leftarrow (M_1 - X\text{IN}_P) \times C_1$
  else
    $X_P \leftarrow M_1 - 1$
  end if
  return X_P

where $M_1 = 2^{32}$, $M_2 = 2^{31}$ and ratios C_1 and C_2 are pre-computed floats.

A similar method is applied to recover P_P. For the first iteration, the output of the PWLC map is equal to $X_P(0) = PWLCmap(IV_P{+}V3_P(0), P_P)$. When the execution time is the shortest, $X\text{IN}_P(0) \in \{0; P_P; 2^{31}\}$. However, the specification of the PCNG establishes that $X\text{IN}_P(0) \notin \{0; 2^{31}\}$. So if the execution time is the shortest and $X\text{IN}_P(0) \notin \{0; 2^{31}\}$ then $X\text{IN}_P(0) = P_P$.

3.2.3 AT09/AT17: $K1_S$ and $K1_P$, first coefficient of recursive cell

The parameters of the chaotic maps P_S, P_P are now known by the attacker. $X\text{IN}_S(0)$ and $X\text{IN}_P(0)$ can be set to a desired value by the attacker using the right IV. Con-

sidering this two facts, the attacker can now inject any value to the delay register of the recursive cells. The objective of AT09 and AT17 is to attack the multiplications $V1_S(1) = K1_S\, X_S(0)$ and $V1_P(1) = K1_P\, X_P(0)$ using the method described by Nguyen et al. [46] to recover $K1_S$ and $K1_P$. This attack requires to be able to measure $V1_S(1)$ and $V1_P(1)$ and to be able to set the value $X_S(0)$ and $X_P(0)$. The method is dividing the attacked words into 8-bit slices in order to reduce the memory footprint and increase the speed of the attack.

3.2.4 AT10 → AT16/AT18 → AT24: Xi_S, Ki_S, Xi_P and Ki_P, remaining keys of recursive cell

This section explains how the remaining keys of the recursive cell are recovered. At this point, we already recovered the PKi_S, the PKi_P, P_S, P_P, $K1_S$ and $K1_P$. At this point, four more CPA attacks are required to recover the last necessary intermediary values:

— AT11: $V2_S(1) = K1_S\, X_S(0) + K2_S\, X_S(-1)$,

— AT14: $V3_S(1) = K1_S\, X_S(0) + K2_S\, X_S(-1) + K3_S\, X_S(-2)$,

— AT19: $V2_P(1) = K1_P\, X_P(0) + K2_P\, X_P(-1)$,

— AT22: $V3_S(1) = K1_P\, X_P(0) + K2_P\, X_P(-1) + K3_P\, X_P(-2)$.

Once these last four values are recovered, successive applications of the extended Euclidean algorithm are required to recover all remaining keys.

— AT10: Knowing $\boldsymbol{PK1_S}$(AT01) and $\boldsymbol{K1_S}$(AT09), $X_S(-1) = PK1_S/K1_S$.

— AT12: Knowing $\boldsymbol{V1_S(1)}$(AT09), $\boldsymbol{X_S(-1)}$(AT10) and $\boldsymbol{V2_S(1)}$(AT11),
$V2_S(1) - V1_S(1) = K2_S\, X_S(-1) \Rightarrow K2_S = \frac{V2_S(1)-V1_S(1)}{X_S(-1)}$.

— AT13: Knowing $\boldsymbol{PK2_S}$(AT02) and $\boldsymbol{K2_S}$(AT12), $X_S(-2) = PK2_S/K2_S$.

— AT15: Knowing $\boldsymbol{V2_S(1)}$(AT11), $\boldsymbol{X_S(-2)}$(AT13) and $\boldsymbol{V3_S(1)}$(AT14),
$V3_S(1) - V2_S(1) = K3_S\, X_S(-2) \Rightarrow K3_S = \frac{V3_S(1)-V2_S(1)}{X_S(-2)}$.

— AT16: Knowing $\boldsymbol{PK3_S}$(AT03) and $\boldsymbol{K3_S}$(AT15), $X_S(-3) = PK3_S/K3_S$.

— AT18: Knowing $\boldsymbol{PK1_P}$(AT04) and $\boldsymbol{K1_P}$(AT17), $X_P(-1) = PK1_P/K1_P$.

— AT20: Knowing $\boldsymbol{V1_P(1)}$(AT17), $\boldsymbol{X_P(-1)}$(AT18) and $\boldsymbol{V2_P(1)}$(AT19),
$V2_P(1) - V1_P(1) = K2_P\, X_P(-1) \Rightarrow K2_P = \frac{V2_P(1)-V1_P(1)}{X_P(-1)}$.

— AT21: Knowing $\boldsymbol{PK2_P}$(AT05) and $\boldsymbol{K2_P}$(AT20), $X_P(-2) = PK2_P/K2_P$.

— AT23: Knowing $\boldsymbol{V2_P(1)}$(AT19), $\boldsymbol{X_P(-2)}$(AT21) and $\boldsymbol{V3_P(1)}$(AT22),
$V3_P(1) - V2_P(1) = K3_P\, X_P(-2) \Rightarrow K3_P = \frac{V3_P(1)-V2_P(1)}{X_P(-2)}$.

— AT24: Knowing $\boldsymbol{PK3_P}$(AT06) and $\boldsymbol{K3_P}$(AT23), $X_P(-3) = PK3_P/K3_P$.

3.2.5 AT25: tr, the transient phase

tr is a 8-bit number, element of the secret key, defining the number of iterations needed by the system before enabling the sample output. By definition, *tr* is directly correlated to the execution time of the transient phase, so is a weak element of the secret key. A Simple Power Analysis (SPA) or a timing attack enables an attacker to recover *tr*. For a timing attack, the attacker is required to measure the duration of the transient state Δ_{tr} and the duration of a single iteration Δ_{it}, resulting in $tr = \Delta_{tr}/\Delta_{it}$.

3.2.6 Recommended design improvements

Performing the previously presented attacks, some recommendations were highlighted.

— The design being composed of two distinct cells, it is fairly easy for any attacker to attack one cell at a time, dividing the complexity of the attack. It would be recommended to strongly improve how the two cells are associated.

— The current design is non-linear in its domain of definition, however, it is linear by pieces possibly enabling algebraic attacks. It would be advised to add more non-linearity.

— To counter the CPA attacks on the recursive cells, countermeasures like masking should be applied to prevent this type of attack.

— There is no real countermeasure to the timing attack on *tr*, however, the risk linked to recovering *tr* is negligible and gives no information on any other elements of the key.

3.3 New stream cipher

After exposing, in the previous section, the vulnerabilities of the studied generator, this section aims to propose a more secure LightWeight Chaos-Based (LWCB) Stream Cipher (SC) by proposing design and implementation level countermeasures. Figure 3.2 presents in red the modifications given to the previous version.

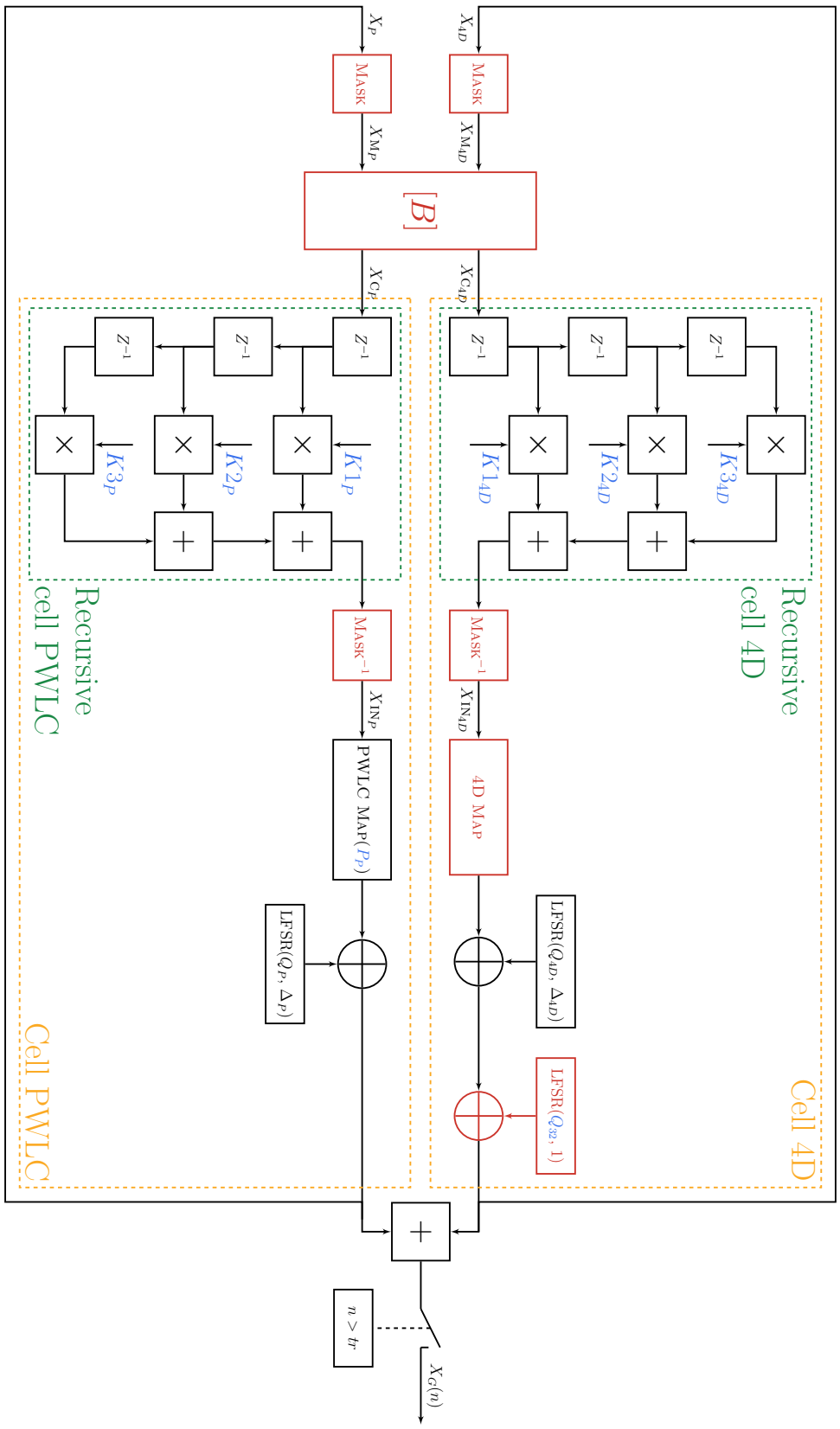


Figure 3.2 – Improved PCNG diagram. In red are shown the differences with [1]. In green are highlighted the recursive cell. Orange dotted line delimits the two cells.

3.3.1 Map bloc modification

Two modifications are proposed for the chaotic maps. First, the ST map and the PWLC map being partly linear, the design was missing a non-linear component to improve the resilience against algebraic attacks. The second modification relies on the implementations of the chaotic maps, the computation of the maps is not constant in time, creating leakage of parameters of the map.

3.3.1.1 New chaotic map

To replace the ST map, a chaotic map based on the Chebychev 4th order polynomial is proposed. Chebyshev polynomials are widely used among chaotic cryptography to perform encryption of all kinds. Due to their non-linearity and their chaotic behaviour on [-1,1] [47]. The polynomials are defined by the following suite:

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x). \end{aligned} \tag{3.6}$$

So the equation of the Chebychev 4th order polynomial is

$$T_4(x) = 8x^4 - 8x^2 + 1 = 8x^2(x^2 - 1) + 1. \tag{3.7}$$

Figure 3.3a represents the attractor of the T_4 polynomial.

However, the stream cipher is only using maps of type $M : E \to E$ where E is the subset taking all natural integers in $[1, 2^N - 1]$. To match the system's domain of definition of the chaotic maps, a discrete version of the T_4, called T_{4D}, is defined in Equation (3.8).

Figure 3.3b is the attractor of the discrete T_{4D} map.

$$\begin{aligned} T_{4D}(X) &= \left(T_4(\frac{X}{2^{N-1}} - 1) + 1\right) \times 2^{N-1} \mod 2^N \\ &= \left[8\left(\frac{X}{2^{N-1}} - 1\right)^2 \left(\left(\frac{X}{2^{N-1}} - 1\right)^2 - 1\right) + 2\right] \times 2^{N-1} \mod 2^N \\ &= \left[2^{N+2}\left(\frac{X}{2^{N-1}} - 1\right)^2 \left(\left(\frac{X}{2^{N-1}} - 1\right)^2 - 1\right) + 2^N\right] \mod 2^N \\ &= \left[\frac{2^{N+2}}{2^{4N-4}}(X - 2^{N-1})^2 \left((X - 2^{N-1})^2 - 2^{2N-2}\right)\right] \mod 2^N \\ &= \frac{1}{2^{3N-6}}\left[(X - 2^{N-1})^4 - 2^{2N-2}(X - 2^{N-1})^2\right] \mod 2^N \end{aligned} \tag{3.8}$$

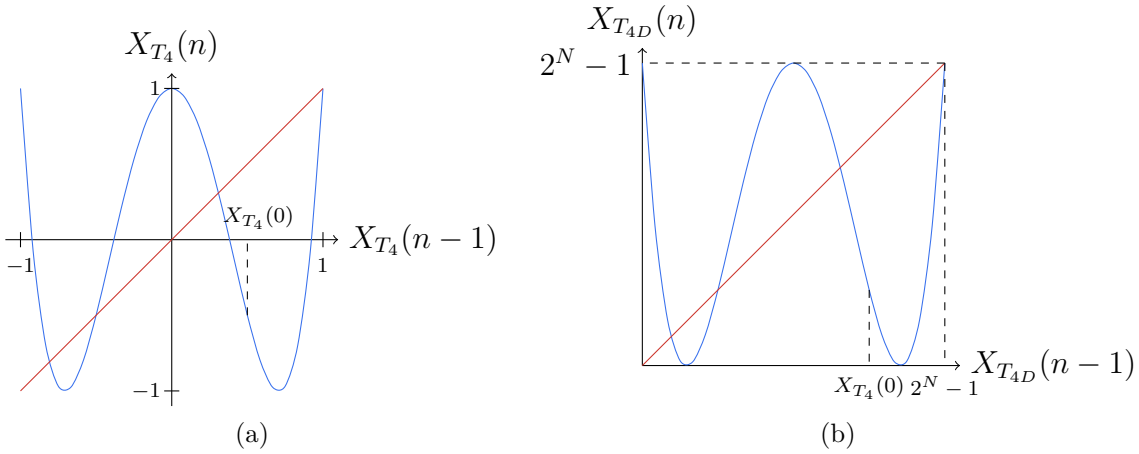


Figure 3.3 – Map's attractor of (a) Chebychev 4-th order polynomial T_4 and (b) Discrete Chebychev 4-th order polynomial T_{4D}.

The drawback with Chebychev polynomial chaotic maps is its non-uniform histogram in the neighbourhood. The histogram of the discrete T_{4D} is depicted in Figure 3.4a. The sample generated tends to concentrate on the extremes. To attenuate this behaviour, the proposed solution is to XOR the samples of the chaotic maps and a 32-bit LFSR. Figure 3.5 represents the schematic of the solution and Figure 3.4b the histogram of the proposed solution.

The ST map is replaced by the T_{4D} discrete 4th order Chebychev polynomial and an LFSR. The initial value of the LFSR will replace the key parameter of the ST map. This change does not affect the length of the global key.

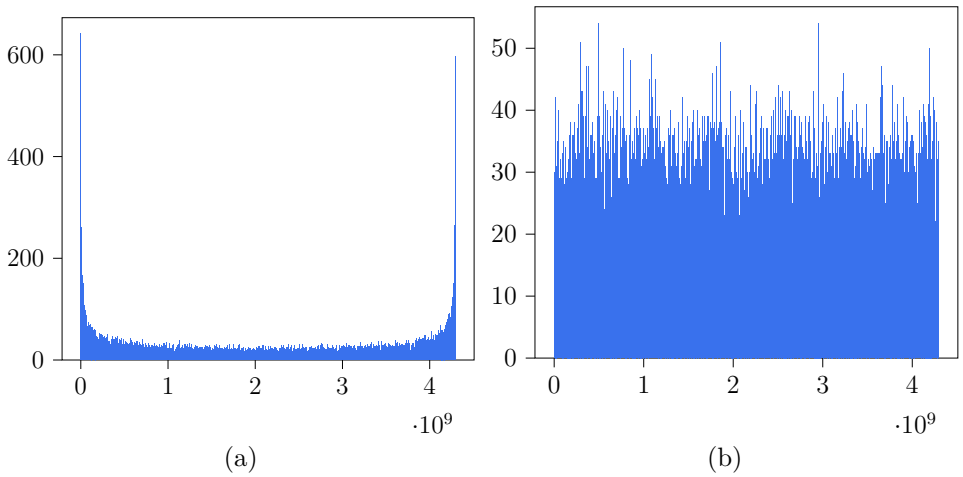


Figure 3.4 – Histogram for 31250 samples and 1000 bins of the 4D map (a) without any additional LFSR (b) with an additional LFSR.

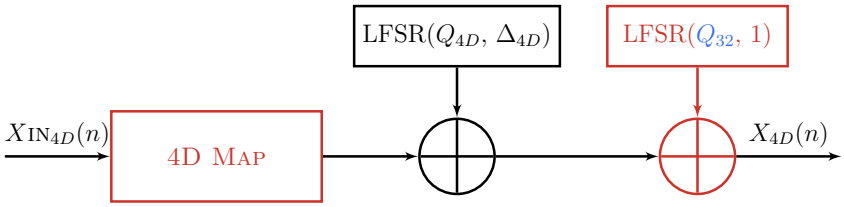


Figure 3.5 – New map block diagram replacing the ST map.

3.3.1.2 Constant time implementation

Section 3.2.2 presents a timing attack on the chaotic maps to recover the parameters P_S and P_P. Both maps are partly linear, and are implemented using a *case* structure, implying that only one branch is computed each time and the processing time of each processing is different. The ST map being removed and replaced by a continuous chaotic map, only the PWLC map implementation presented in Algorithm 3.2 is revised. The countermeasure proposed in Algorithm 3.3 is to compute all the cases each time and then select the correct value.

The time variation of this new implementation is verified running multiple times the map implementation with multiple keys comparing the standard deviation between the two implementations.

Figure 3.6 presents the NCpB for 100 different secret keys of both implementation. The standard deviation of the NCpB drops from 0.97 to 0.33, showing that the implementation

Algorithm 3.3: Constant-time implementation of $X_P(n) = PLWCmap(X\text{IN}_P(n-1), P_P)$.

Require: $X\text{IN}_P \in]0; 2^{32}[$ and $P_P \in [0; 2^{31}[$
$B_1 \leftarrow 0 < X\text{IN}_P < P_P$
$B_2 \leftarrow P_P < X\text{IN}_P < M_2$
$B_3 \leftarrow M_2 < X\text{IN}_P < (M_1 - P_P)$
$B_4 \leftarrow (M1 - P_P) < X\text{IN}_P < M_1$
$B_5 \leftarrow (B_1 + B_2 + B_3 + B_4) = 0$
$X_1 \leftarrow (X\text{IN}_P \times ratio3)\&\text{mask}(B_1)$;
$X_2 \leftarrow ((X\text{IN}_P - P_P) \times ratio4))\&\text{mask}(B_2)$
$X_3 \leftarrow ((M_1 - P_P - X\text{IN}_P) \times ratio4))\&\text{mask}(B_3)$
$X_4 \leftarrow ((M_1 - X\text{IN}_P) \times ratio3))\&\text{mask}(B_4)$
return $(X_1 + X_2 + X_3 + X_4 + ((M_1 - P_P)\&\text{mask}(B_5))$
where $M_1 = 2^{32}$, $M_2 = 2^{31}$ and $\text{mask}(B_X)$ returns 0xFFFFFFFF if $B_X = 1$, otherwise 0.

is close to be constant in time. However the average goes from 42.16 to 45.04. The proposed countermeasure degrades the time performances.

3.3.2 Coupling Matrix

The use of the coupling matrix was largely studied in previous work [48]. In this case, the aim of this cipher is to add more interdependence between the two maps. The divide and conquer approach used in Section 3.2 was possible because the two cells of the generator were independent and only blended at the end to produce a sample. The idea is to blend the two cells each iteration and force an attacker to consider both cells during an attack.

The coupling matrix is a matrix product between the two inputs $X_{4D}(n)$ and $X_P(n)$ and coefficients B_{ij}, each coefficient is set by 5 bits from the secret key. The matrix product is defined as follows:

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

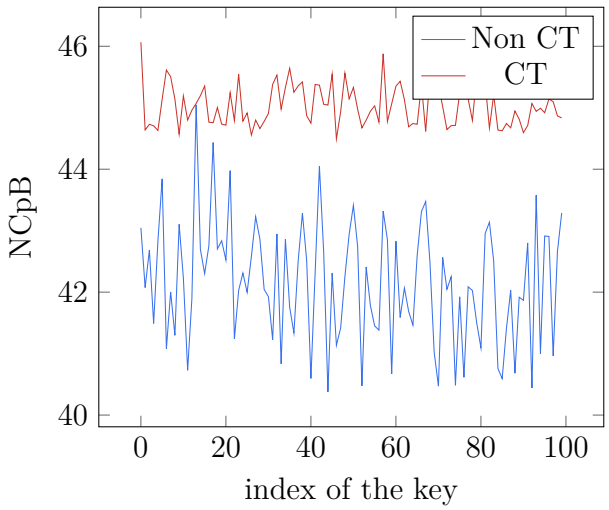


Figure 3.6 – Number of Cycles per Byte (NCpB) function of the secret key of the PCNG using, in blue, the non-Constant-Time (CT), in red, the CT implementation of the PWLC map.

3.3.3 Shift in REC CELL improves key sensitivity

Uniqueness of reduced products inside the IIR filter is primary. Indeed, the filter initialization being based on the secret key, filter output needs to be different for each key; otherwise the generated sequence is the same. Two solutions are possible, the key space can be reduced removing the weak keys or, as proposed below, to shift the result before the reduction to N bits, where N is the internal resolution of the chaotic maps, here $N = 32$.

Let $q = P(C = C')$ be the probability of having $C = C'$ with $C = A \times B$, $C' = A' \times B'$ and $A,\ A',\ B,\ B'$ being four distinct unsigned integers defined on N bits. Equation (3.9) presents the probability of having q in different cases. In our case, the generator is included in the second case, i.e. $q \neq 0$. The proposed solution aims to minimize the probability q.

$$\begin{cases} q = 0 & \text{if } C \text{ or } C' \text{ is defined on 2N-bits} \\ q \neq 0 & \text{if } C \text{ and } C' \text{ are defined on } M,\ M' \text{ bits,} \\ & \text{with } M,\ M' < 2N \end{cases} \tag{3.9}$$

Let $\epsilon(j)$ be equal to $1 << j$ with $\{j \in \mathbb{N} \mid j < N\}$ and let i be an integer in $[0; N-1]$ where i number of right shifts executed before the reduction to N bits. In the worst case,

i.e. $A' = A,\ B' = B \oplus \epsilon(j)$ or $A' = A \oplus \epsilon(j),\ B' = B$, the probability q is equal to:

$$\begin{aligned} q_N(i) = &\ P((A \times B) >> i = (A \times (B \oplus \epsilon(N-1))) >> i) \\ &+ P((A \times B) >> i = (A \times (B \oplus \epsilon(0))) >> i) \\ = &\ 2^{-(i+1)} + 2^{-(N-i)} \end{aligned}$$

Figure 3.7 shows the value of q_N depending on the value i for $N = 32$. Minimum of q_{32} is obtained for $i = 15$ and $i = 16$. In the rest of the chapter, we consider the value $i = 16$. The new generic block diagram of a cell using shifting is presented in Figure 3.8.

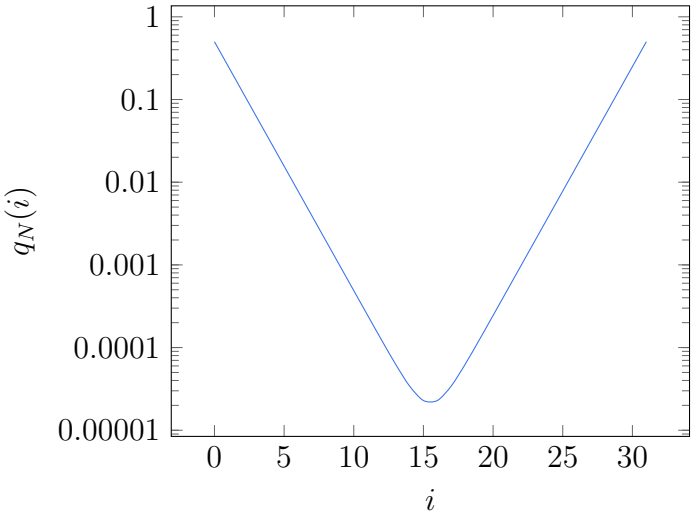


Figure 3.7 – Probability $q_N(i)$ of having $A \times B = A' \times B'$ where i is the number of right shift and A, B, A' and B'being four distinct unsigned integers defined on N bits, for $N = 32$.

3.3.4 Masking

In the assessment of the security of the original implementation of the cipher, the IIR structure is weak against SCA. The straightforward countermeasure for linear application is masking. In this context, we propose to add the masking countermeasure to first only the IIR structure, then to the coupling matrix and the IIR structure, with an unmasking just before the input of the chaotic maps.

Masking consists of adding or XORing a randomly generated value to an input or intermediary value, and then carry on computations with the value masked. The real

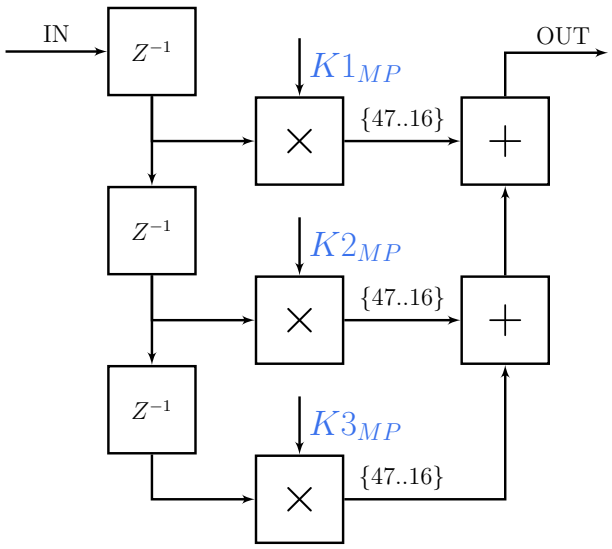


Figure 3.8 – New generic block diagram of a cell using shifts.

challenge of this countermeasure is to recover the correct value from the result of the computation.

3.3.4.1 Masking only the IIR structure

Studied cipher using Multiply–ACcumulate (MAC) operations, additive masking is more suited for IIR structure. So masked value $XM_C(n)$ at iteration n is defined

$$XC_{MP_M}(n) = XC_{MP}(n) + mask_{MP}(n),$$

where $XC_{MP}(n)$ is the output of the coupling matrix and $mask_{MP}(n)$

The IIR function is defined by

$$F_{IRR_{MP}}(X(n)) = \sum_{i=1}^{D} X(n-i) \times Ki_{MP}.$$

To unmask the result of the IIR structure, which is a linear function, can be written

$$\begin{aligned} Xin_{MP}(n) =& F_{IRR_{MP}}(XC_{MP}(n)) = F_{IRR_{MP}}(XC_{MP_M}) - F_{IRR_{MP}}(mask_{MP}(n)) \\ =& \sum_{i=1}^{D} XC_{MP_M}(n-i) \times Ki_{MP} - \sum_{i=1}^{D} mask_{MP}(n-i) \times Ki_{MP}. \end{aligned}$$

3.3.4.2 Masking coupling matrix and IIR structure

Masking operation in this case is similar. Additive masking is also used,

$$X_{MP_M}(n) = X_{MP}(n-1) + mask_{MP}(n).$$

In this case, the masked data goes through the coupling matrix and the IIR structure without any unmasking. The unmasking operation is defined as follows:

$$Xin_{MP}(n) = F_{IRR_{MP}}(F_{Coupling_{MP}}(X_{MP_M}(n)) - F_{IRR_{MP}}(F_{Coupling_{MP}}(mask_{MP}(n))).$$

3.3.5 Set up of the key

Figure 3.2 depicted the PCNG being used in the stream cipher. To be a cipher, it is required to have a secret key and an IV. To do so, multiple parameters, and initial register values are defined as the secret key. All secret key elements are defined in Table 3.3.

Table 3.3 – Key composition of the proposed stream cipher.

Name	Length(in bits)	Description
$K1_{4D}$	32	First coefficient of the 4D map recursive cell
$K2_{4D}$	32	Second coefficient of the 4D map recursive cell
$K3_{4D}$	32	Third coefficient of the 4D map recursive cell
$X_{4D}(-1)$	32	Initial value of the first delay of the 4D map recursive cell
$X_{4D}(-2)$	32	Initial value of the second delay of the 4D map recursive cell
$X_{4D}(-3)$	32	Initial value of the third delay of the 4D map recursive cell
P_P	31	Parameter of the PWLC map
$K1_P$	32	First coefficient of the PWLC map recursive cell
$K2_P$	32	Second coefficient of the PWLC map recursive cell
$K3_P$	32	Third coefficient of the PWLC map recursive cell
$X_P(-1)$	32	Initial value of the first delay of the PWLC map recursive cell
$X_P(-2)$	32	Initial value of the second delay of the PWLC map recursive cell
$X_P(-3)$	32	Initial value of the third delay of the PWLC map recursive cell
reg_{32}	32	Initial value for the LFSR register
$e_{ij}(0)$	5	Coefficient B_{11} of the coupling matrix
$e_{ij}(1)$	5	Coefficient B_{12} of the coupling matrix
$e_{ij}(2)$	5	Coefficient B_{21} of the coupling matrix
$e_{ij}(3)$	5	Coefficient B_{22} of the coupling matrix
tr	8	Number of samples to reach before outputting the first sample(ie. $X_G(0) = X_{OF}(tr)$)
Total	475	

The computation of the input of the chaotic maps without masking is defined in Equation (3.3) and MP is equal to $4D$ for the 4D map and P for PWLC.

The output of the generator go through a transient phase. Until the number of iteration reaches tr, the generator is outputting 0, then the generator start outputting the computed samples. This phase ensures the generator is not in a transient state anymore.

3.4 Results and discussion

3.4.1 National Institute of Standards and Technology (NIST) Statistical Tests Suite (STS) SP 800-22

NIST STS [49] the popular test suite for investigating the randomness of binary data is applied. The suite contains 188 tests and sub-tests that assess the randomness of arbitrarily long binary sequences. These tests focus on different types of non-randomness that could exist in a sequence.

To perform the different tests, 100 sequences of 31250 32-bit samples (i.e., 1 million bits per sequence) are generated using 100 different secret keys. All 188 tests and sub-tests of the suite are run. For each test, a set of 100 P_{value} is produced and a sequence passes a test whenever the $P_{value} \geq \alpha = 0.01$, where α is the level of significance of the test. A value of $\alpha = 0.01$ means that 1% of the 100 sequences are expected to fail. The proportion of sequences passing a test is equal to the number of $P_{value} \geq \alpha$ divided by 100.

Table 3.4 presents the NIST STS's results of the constant-time version. The P_{values} of all the tests are strictly over 0.01, meaning that the cipher passed all the tests. Passing this test is necessary, but not sufficient to state that generated sequences are random.

3.4.2 Histogram distribution

The aim of this test is to determine if the histogram distribution is uniform. To assert that, the χ^2 test is used. If generated sequence verifies (3.10), the key associated passes the χ^2 test.

$$\sum_{i=0}^{C} \frac{(V_{observed}(i) - V_{expected})^2}{V_{expected}} < V_{critical},$$

where $V_{expected}$ is the expected histogram count for a flat histogram,
$V_{observed}(i)$ is the histogram count for the i-th class,
and $V_{critical}$ is the maximum value the test can reach. (3.10)

Table 3.4 – Average results on NIST STS for 100 keys.

Statistical Test	P_{value}	Proportion
ApproximateEntropy	0.508188	98.96
BlockFrequency	0.474995	99.07
CumulativeSums	0.539631	99.08
FFT	0.478261	98.85
Frequency	0.515495	99.06
LinearComplexity	0.453463	99.11
LongestRun	0.464725	99.01
NonOverlappingTemplate	0.501178	98.99
OverlappingTemplate	0.463584	98.73
RandomExcursions	0.424253	98.91
RandomExcursionsVariant	0.422496	99.12
Rank	0.471351	99.08
Runs	0.501347	99.04
Serial	0.495546	98.90
Universal	0.527559	98.93

This test is run on our algorithm and some reference algorithms. The test conditions are the following.

— The test is run independently over 1000 randomly generated keys, and IVs.

— Samples are unsigned 32-bit integers.

— 10^8 samples are generated per sequence.

— $C = 1000$ classes are used.

— $V_{expected} = \frac{10^8}{C} = 10^5$

— $V_{critical}$ is computed using the inverse of the chi-square cumulative distribution function as defined in [50, 51]. For this chapter, $V_{critical} = 1073.6$.

Table 3.5 shows the percentage of keys passing the χ^2 test with a set of 1000 random keys and different algorithms. The performance of literature algorithms is close to 95%. Taha [1] 3-delay PCNG is only presenting 88,1% passing keys, but 95.5% keys for the proposed PCNG pass the test and is close to standard algorithms.

3.4.3 Correlation - Hamming Distance (HD)

These tests show the non-similarity of two generated streams from two different keys. For these tests, a data set of 100 125000-bit long streams are generated using 100 randomly generated keys. Then, correlation coefficients and average HDs between the streams are

Table 3.5 – Histogram performance.

Algorithm	Key passing χ^2 test
Taha [1] - 3 delays	88,1%
LWCB PCNG - 3 delays	95.5%
Advanced Encryption Standard (AES)	94.9%
HC-128 [5]	95.4%
Rabbit [3]	95.5%

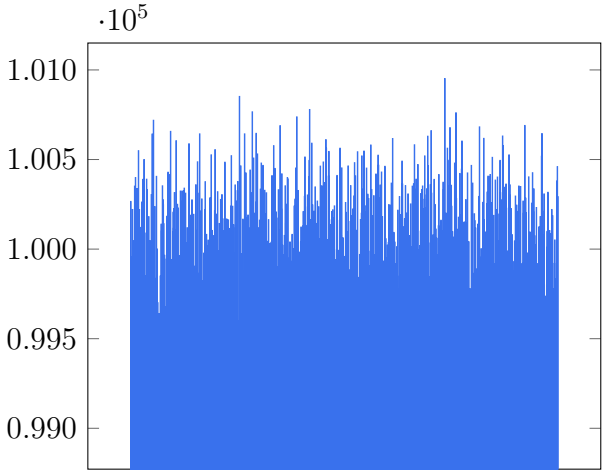


Figure 3.9 – Histogram of the PCNG for one billion samples and 1000 bins.

computed. The correlation coefficient is computed using the binary representation of the sequences where $1 \rightarrow 1$ and $0 \rightarrow -1$. The expected value of the correlation coefficient ρ_{ij}, for two completely random sequences, should be equal to 0. The average HD is defined in (3.11), where S_x is the generated sequence of size L, x is the index of a key inside an array of 100 random keys. The expected value, for two completely random sequences, should be equal to $\frac{1}{2}$.

$$HD(S_i, S_j) = \begin{cases} \frac{1}{L} \times \sum_{k=1}^{L} S_i(k) \oplus S_j(k) & \text{if } i \neq j, \\ \frac{1}{2} & \text{otherwise.} \end{cases} \tag{3.11}$$

Figure 3.10 shows the obtained correlation coefficients between two-by-two different sequences. As we can see, all correlation coefficients are centred around 0 and maximum and minimum values are bounded by 4.13×10^{-3}, result expected for non-correlated sequences.

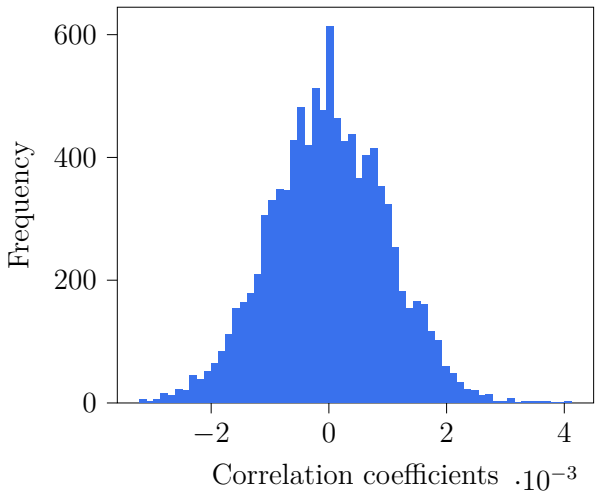


Figure 3.10 – Frequency distribution of the correlation coefficients between generated stream with different keys.

Figure 3.11 shows HDs centred around $\frac{1}{2}$, and maximum deviation is bound by $2,07 \times 10^{-3}$, meaning there is equal chance to generate a 0 or 1.

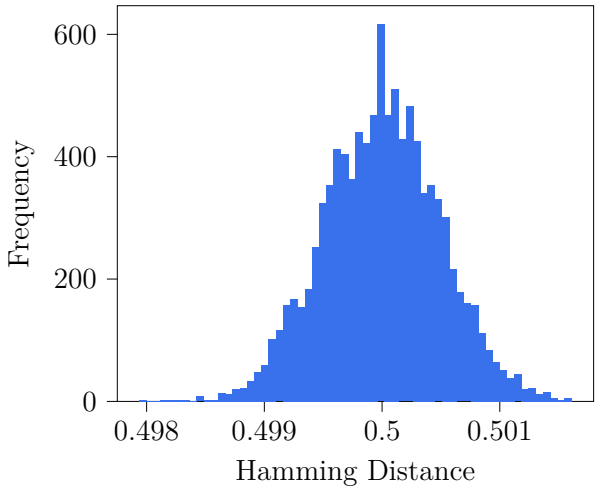


Figure 3.11 – Frequency distribution of the HDs between generated stream with different keys.

3.4.4 Key sensitivity

Evaluation of the system robustness against key sensitivity attacks can be assessed using many existing tools such as Unified Average Changing Intensity (UACI) and Number

of Pixels Change Rate (NPCR) [52, 53]. To compute these metrics, one random key is generated K_1 and a key with only one bit difference K_2 is created. The two keys are then used to generate a 8-bit-word stream S_1 using K_1, and S_2 using K_2 of length L. The UACI and the NPCR are defined as follows:

$$UACI(S_1, S_2) = \frac{1}{L\,2^8} \sum_{k=1}^{L} |S_1(i) - S_2(i)|\ 100\%, \tag{3.12}$$

$$NPCR(S_1, S_2) = \frac{1}{L} \sum_{k=1}^{L} D_{S_1,S_2}(k)\ 100\%, \tag{3.13}$$

with:

$$D_{S_1,S_2}(k) = \begin{cases} 0, & \text{if } S_1(k) = S_2(k) \\ 1, & \text{if } S_1(k) \neq S_2(k) \end{cases}. \tag{3.14}$$

The optimal NPCR and UACI values of a secure encryption scheme against key sensitivity attacks are 99.58% and 33.46%, respectively [54].

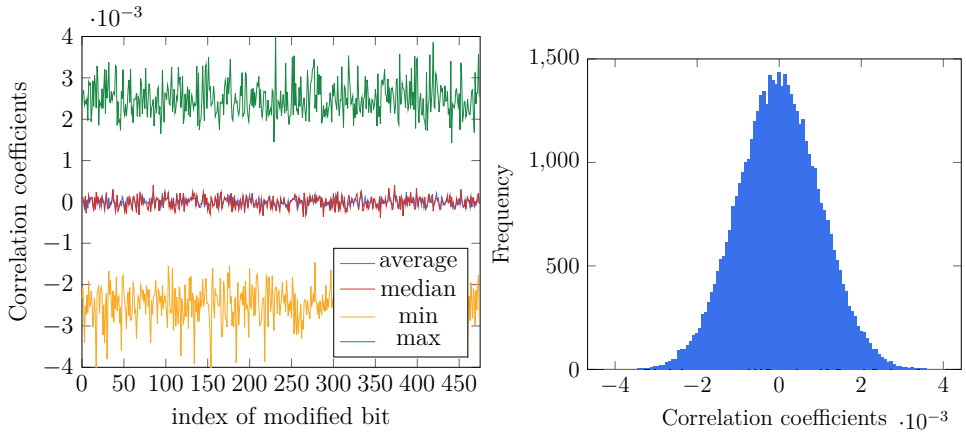
To ensure the uniqueness of the streams generated by similar keys, correlation coefficients, HDs, UACIs and NPCR between a 125000-byte long stream S_{K_i} generated by key K_i and the corresponding stream $S_{K_{ij}}$ generated by the key K_i with the j^{th} bit modified, noted K_{ij}.

Figure 3.12 depicts the average correlation coefficient function of the position of the modified bit of the key and the frequency distribution of the coefficients. Like in the previous section, the binary sequences are interpreted with $1 \rightarrow 1$ and $0 \rightarrow -1$ and the expected value is 0. The overall averages around -3.94×10^{-6} and with a standard deviation at 1.02×10^{-4}.

Figure 3.13 illustrates the average HDs function of the position of the modified bit of the key and the frequency distribution of the HDs. The average is close(-1.96×10^{-6}) to the expected value($\frac{1}{2}$) with a standard deviation of 5.13×10^{-5}.

Figure 3.14 and Figure 3.15 represent the average UACIs and NPCRs function of the position of the modified bit of the key and its frequency distribution. The UACIs average at the optimal value of 33.46% with a standard deviation of 6.2×10^{-3}. However, NPCRs average with a small offset of 0.03% and standard deviation of 1.8×10^{-3}.

The computed correlation coefficients, HDs, UACIs and NPCR show that similar secret keys produce completely different and uncorrelated streams.



(a) (b)

Figure 3.12 – (a) Average correlation coefficients and (b) frequency distribution of correlation coefficients between stream S_{K_i} and $S_{K_{ij}}$.

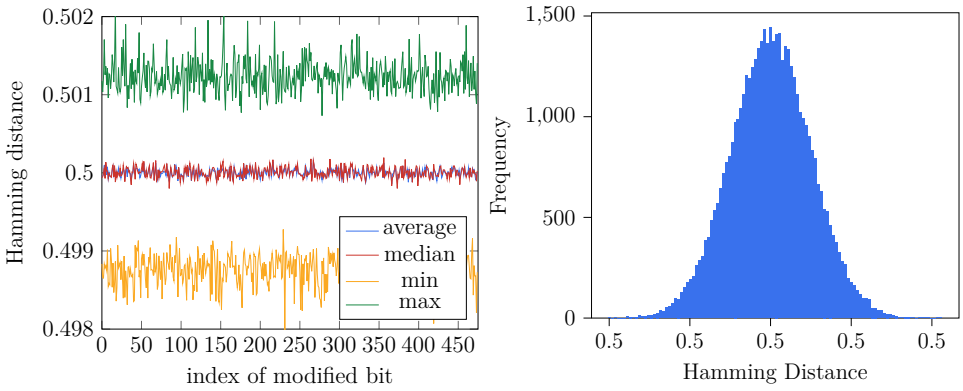
3.4.5 Confusion Analysis

This section aims to evaluate the capacity of the stream cipher to encrypt a plain text. For the following tests, a set of 100 randomly generated key noted K_i with i being the index of the key, a data set of 48 512x512 grayscale images, noted I_j where j is the index of the image, are used to generate for each image I_j a cipher text noted $C_{K_i}(I_j)$, using the keys K_i. The entropy $H(C_{K_i}(I_j))$, the redundancy $R(C_{K_i}(I_j)) = 8 - H(C_{K_i}(I_j))$, horizontal $\rho_h(C_{K_i}(I_j))$ and vertical $\rho_v(C_{K_i}(I_j))$ correlation coefficient are computed on the ciphered images. The horizontal correlation coefficient of an image I of width W and height H is computed as follows:

$$\begin{aligned} \rho_h(I) &= \frac{cov(X_h, Y_h)}{\sigma_{X_h}\,\sigma_{Y_h}}, \\ \text{where } X_h &= [I(0,0), ..., I(W,0), I(0,1), ..., I(W-1,H)], \\ \text{and } Y_h &= [I(1,0), ..., I(W,0), I(0,1), ..., I(W,H)]. \end{aligned} \tag{3.15}$$

The vertical correlation coefficient of an image I of width W and height H is computed as follows:

$$\begin{aligned} \rho_v(I) &= \frac{cov(X_v, Y_v)}{\sigma_{X_v}\,\sigma_{Y_v}}, \\ \text{where } X_v &= [I(0,0), ..., I(0,H), I(1,0), ..., I(W,H-1)], \\ \text{and } Y_v &= [I(0,1), ..., I(0,H), I(1,0), ..., I(W,H)]. \end{aligned} \tag{3.16}$$



(a) (b)

Figure 3.13 – (a) Average HDs and (b) frequency distribution of HDs between stream S_{K_i} and $S_{K_{ij}}$.

The redundancy of the ciphered images illustrated in Figure 3.16 shows an average of 8.78×10^{-5} and a standard deviation of 6.63×10^{-7}.

Figure 3.17 and Figure 3.18 shows the average horizontal and the vertical correlation coefficients function of the images and its frequency distribution. The correlation coefficients average at 1.22×10^{-4} with a standard deviation of 1.38×10^{-4} and 2.55×10^{-5} with a standard deviation of 1.57×10^{-4} respectively.

3.4.6 Time performance

Time measurements are done on an Intel Core i7-7700 Central Processing Unit (CPU) @3.60GHz. The test environment is set as follows:

— CPU frequencies are fixed at 3.60 GHz.

— Hyper-Threading is disabled.

— Pre-fetching is disabled.

— Process is assigned to a core using *taskset* command.

The function *gettimeofday()* is used to measure the time elapsed between the beginning and the end of the encryption. The message to encrypt is 125000 bytes long.

The metric used in this chapter is defined as follows.

$$NCpB = \frac{F \times t}{M \times K} \tag{3.17}$$

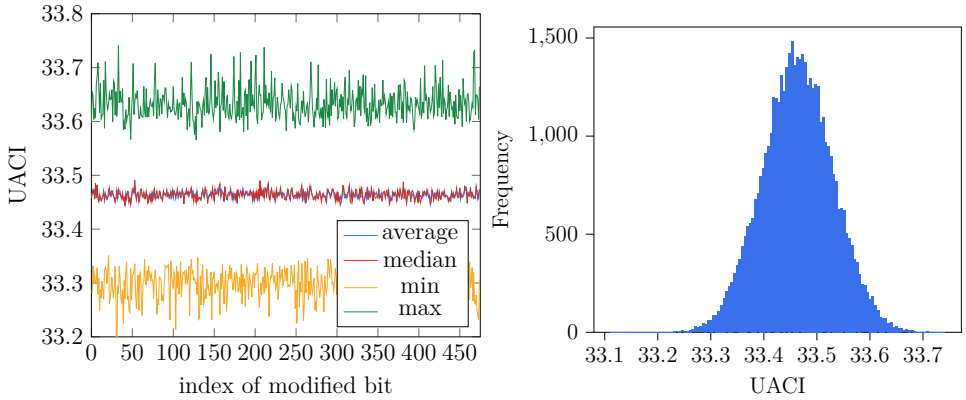


Figure 3.14 – (a) Average UACIs and (b) frequency distribution of UACIs between stream S_{K_i} and $S_{K_{ij}}$.

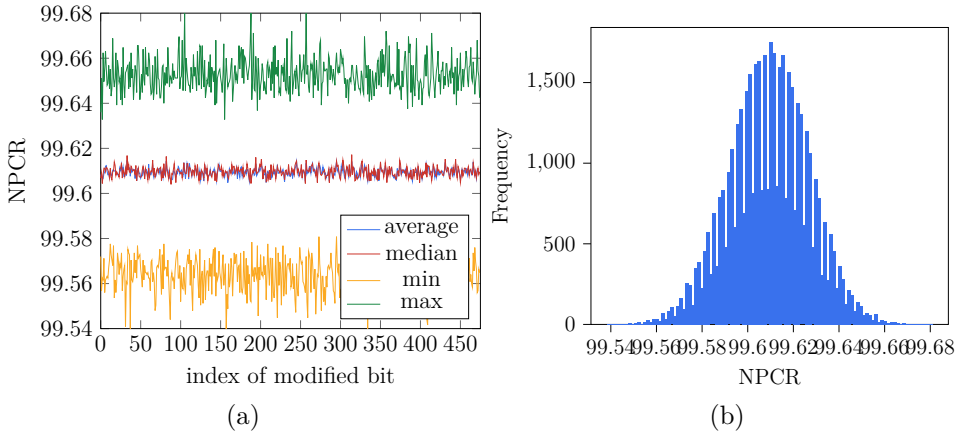


Figure 3.15 – (a) Average NPCRs and (b) frequency distribution of NPCRs between stream S_{K_i} and $S_{K_{ij}}$.

where t is the time measured, K is the number of keys used, M is the size, in bytes, of the message and F is the frequency of the CPU. In this chapter:

— $F = 3.60$ GHz.

— $M = 125000$ Bytes.

— $K = 100$ Keys.

Table 3.6 presents timing performance for different implementations of our cipher and some standard encryption methods. As shown in Table 3.6, the proposed LWCB PCNG is two times slower than in AES-CounTeR (CTR). HC-128 and Rabbit present better

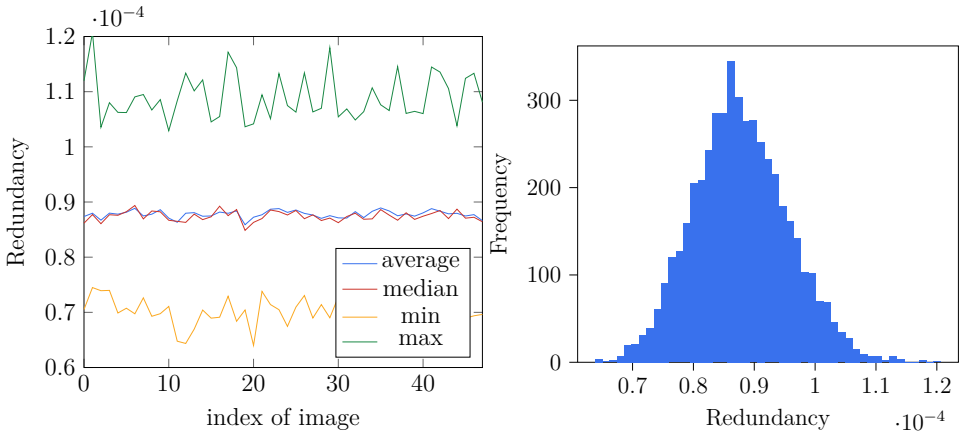


Figure 3.16 – (a) Average redundancy and (b) frequency distribution of ciphered images $C_{K_i}(I_j)$.

performance, however, these algorithms manifest some weaknesses against some attacks such as injection and side-channel attacks mentioned in [45].

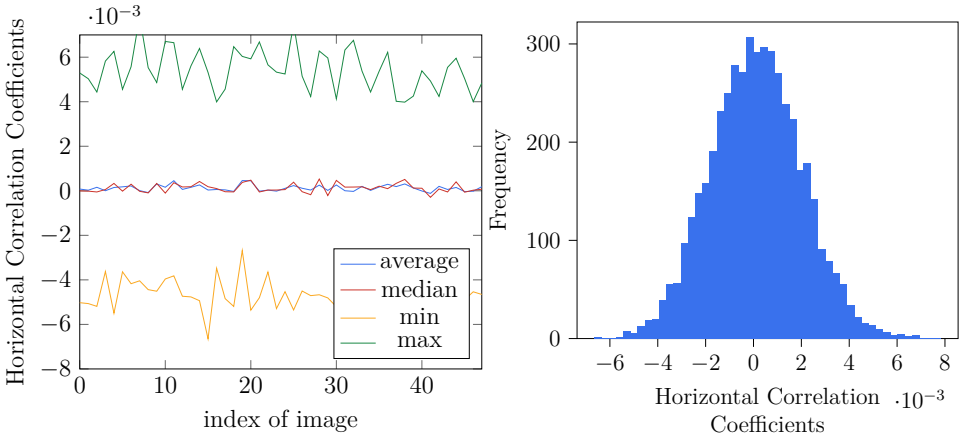
Table 3.6 – Timing of the different cipher versions compared to standard ciphers.

Cipher	NCpB
Taha [1]	22.59
LWCB-SC	45.04
HC-128 [5]	8.59
Rabbit [3]	5.27
AES CTR	24.38

3.5 Conclusion

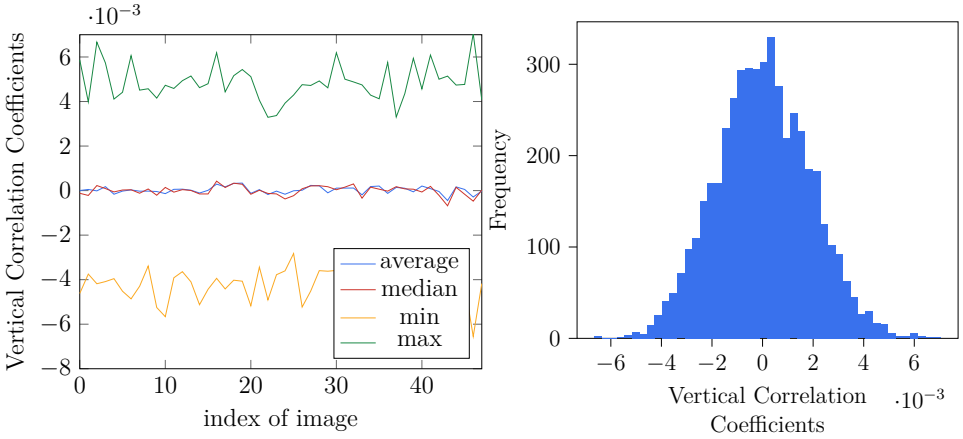
This chapter presents a method of attack using SCA on the stream cipher proposed by Taha [1]. By understanding this method of attack, a new PCNG and an associated stream cipher is proposed. This PCNG contains design modifications to counter some of the attacks, and classic countermeasures, like masking are applied on some part of the design.

Then the security against statistical attacks of the new design is checked using several statistical tests like the NIST STS, correlation, histogram analysis, key sensitivity and confusion analysis. A measurement of the time performance is also proposed. However



(a) (b)

Figure 3.17 – (a) Average horizontal correlation coefficients and (b) frequency distribution of ciphered images $C_{K_i}(I_j)$.



(a) (b)

Figure 3.18 – (a) Average vertical correlation coefficients and (b) frequency distribution of ciphered images $C_{K_i}(I_j)$.

the time measurements are not better than the state-of-the-art encryption schemes due to the relatively high complexity added by the 4D map for example.

A hardware implementation of the proposed encryption scheme should be proposed to study the impact of the modifications on SCA.

CHAPTER 4

Hardware Implementation of the proposed Lightweight Chaos-Based Stream Cipher

In the literature, multiple ciphers are defined and implemented in hardware devices. We especially studied, for its low-latency, the stream ciphers in which the ciphertext is obtained by performing an eXclusive OR (XOR) between the plaintext 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 [55, 56], each optimizing a trade-off between surface and speed. Moreover, some ciphers, like Trivium [6], are designed to be hardware friendly and minimize logic resources. Some ciphers, based on chaos theory, have also hardware implementations [57, 58]. Those ciphers usually require a high logic resources to digitize chaotic systems.

Based on works in Chapter 3, an implementation of the LightWeight Chaos-Based (LWCB) Stream Cipher (SC) on Field-Programmable Gate Array (FPGA) hardware platform is proposed in this chapter. The system includes some countermeasures against Side-Channel Attack (SCA) [59], such as Correlation Power Analysis (CPA) and Differential Power Analysis (DPA). This effectiveness of theses countermeasures is assessed. The proposed hardware implementation achieves a throughput of 673.68 Mbps at an operating frequency of 21.05 MHz. A pipelined implementation operating at 80 MHz and achieving a throughput of 2 560 Mbps is also proposed.

The chapter is organized as follows. The existing stream ciphers and their implementation are first presented in Section 4.1. Section 4.2 investigates the design and details of the hardware implementation of the chaos-based stream cipher such as the implementation of the chaotic maps, and the masking of the coupling matrix and the recursive cells. The performance of the proposed system is assessed in Section 4.3 in terms of both throughput and used logic resources. Moreover, the methodology proposed by Schneider and Moradi [60] to test the resilience of a system against SCAs is used on the proposed implementation. Finally, Section 4.4 concludes this chapter.

4.1 Related work

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

AES [2] developed in 2001 is the most widely used system. It can be considered as a stream cipher only in its CounTeR (CTR) mode. It takes 128 bits as input and can use a 128, 192 or 256-bit key. Its round function consists of three layers including key addition layer, byte substitution layer called S-Box and diffusion layer. Selected AES hardware implementations [55, 56] have two completely different approaches, [55] try to achieve the highest throughput while [56] aims to minimize the hardware resource usage.

The Rabbit stream cipher [61] 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. The implementation of Rabbit proposed in [62] on a Virtex V achieves the throughput of 9 160 Mbps at 71.582 MHz operating frequency.

Salsa20 [63] 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. Sugier [64] proposed an implementation of Salsa20 on Spartan 3 and Spartan 6 FPGAs. The Spartan 3 implementation operating at 19.4 MHz to provide a throughput of 1 104 Mbps. The Spartan 6 implementation has a throughput of 2 519 Mbps and is operating at 48 MHz.

Trivium [6] is also a cipher from the eSTREAM Project. It is particularly well suited for applications requiring a flexible hardware implementation. The 288-bits internal state is stored in three shift registers which are the heart of the cipher and can be viewed as a circular register. Gaj *et al.* [65] and in their more recent work [66] provided a comparison of

multiple hardware-oriented eSTREAM candidates including Trivium. The two proposed implementations were optimized to maximize the ratio performance per slices. Gaj *et al.* [65] achieves a throughput of 12 160 Mbps and an operating frequency of 190 MHz. In [66], their implementation achieves a throughput of 13 504 Mbps at an operating frequency of 211 MHz.

Several chaotic systems have been developed and used for designing chaotic hardware key generation for secure cryptosystems. Lorenz's[57] and Lü's[58] systems are the most famous ones. A chaotic system can be considered as discrete or continuous. The previously cited systems are continuous and defined by a differential equation. In [57], a cipher to encrypt images is developed and implemented on a FPGA platform. It uses a key generator based on the Lorenz's chaotic system. This implementation achieves a throughput of 124 Mbps at an operating frequency of 15.598 MHz on a Virtex II FPGA. Another example of such system is presented in [58] where a Lü's-system-based chaotic key generator is used. This system is also implemented of a Virtex II and achieves a throughput of 182.9 Mbps with an operating frequency of 22.868 MHz.

4.2 Hardware-friendly Architecture of the proposed Pseudo-Chaotic Number Generator (PCNG)

The block diagram of the proposed generator is depicted in Figure 3.2. 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 block diagrams of the two cells are delimited in orange in Figure 3.2. Both cells are composed of a recursive cell illustrated in green, a discrete map and Linear Feedback Shift Registers (LFSRs). The next two sections present the hardware implementations of the chaotic maps: 4D and PieceWise Linear Chaotic (PWLC).

4.2.1 4D map

The 4D map is a discrete version of the Chebyshev polynomial of degree 4 and is defined in Chapter 3. Taking advantage of the multiplication and division by power of 2, Equation (3.8) is expressed in Equation (4.1) to minimize the logic resources of the hardware implementation.

Figure 4.1 is the simplified Register-Transfer Level (RTL) representation of the 4D map implementation. This implementation of Equation (4.1) requires only two multipliers of 32 and 64-bit inputs, respectively and three adders of 64, 64 and 128 bits inputs, respectively. The left and right shifting are implemented using routing and are not adding any extra resources.

$$\begin{aligned} T_{4D}(X) &= \frac{1}{2^{3N-6}} \left[(X - 2^{N-1})^4 - 2^{2N-2} (X - 2^{N-1})^2 \right] \mod 2^N \\ &= \frac{1}{2^{3N-6}} (Y^2 - 2^{2N-2} Y) \mod 2^N \\ &= [Y \times Y - Y << (2N-2)] >> (3N-6) \mod 2^N, \\ \text{with } Y &= (X - 2^{N-1})^2 = X \times X + (1 << 2N-2) - (X << N). \end{aligned} \tag{4.1}$$

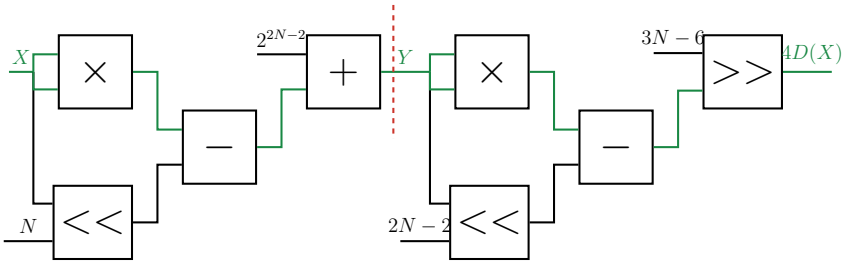


Figure 4.1 – Simplified RTL representation of the 4D map implementation. The position of the pipeline register is represented in red and the critical path in green.

4.2.2 PWLC map

The PWLC map defined by Equation (3.2) can be written as follows:

$$PLWCmap(X, P_P) = \begin{cases} C_1 \times F_1(X) & \text{if } 0 < X < P_P, \\ C_2 \times F_2(X) & \text{if } P_P < X < 2^{N-1}, \\ C_2 \times F_3(X) & \text{if } 2^{N-1} < X < 2^N - P_P, \\ C_1 \times F_4(X) & \text{if } 2^N - P_P < X < 2^N, \\ F_5(X) & \text{otherwise,} \end{cases} \tag{4.2}$$

$$\text{where , } \begin{array}{ll} C_1 = \frac{2^N}{P_P}, & C_2 = \frac{2^N}{2^{N-1}-P_P}, \\ F_1(X) = X, & F_2(X) = X - P_P, \\ F_3(X) = 2^N - P_P - X, & F_4(X) = 2^N - X, \\ F_5(X) = 2^N - 1, & \end{array}$$

and X is the input of the PWLC map and P_P is the parameter of the PWLC map defined in the secret key.

Figure 4.2 shows a simplified RTL representation of the proposed solution. In this implementation, the ratios C_1 and C_2 are precomputed by the key generator to avoid the implementation of 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 to perform the correct computation depending on the value of X. The PWLC map requires only one multiplier with 84-bit inputs.

4.2.3 4-stage pipeline implementation

The system is completely recursive, the previous sample is needed to compute next. Consequently, the proposed implementation cannot take advantage of any temporal parallelism. However, we investigated the possibility of adding pipeline stages by modifying the design.

Figure 4.3 presents, in green, the critical path and where the stages of the pipeline are added to the system in red. In this system, the bottleneck is the 64-bit-input multiplier used inside the 4D map for squaring Y. Therefore, a pipeline stage is nearly dedicated to

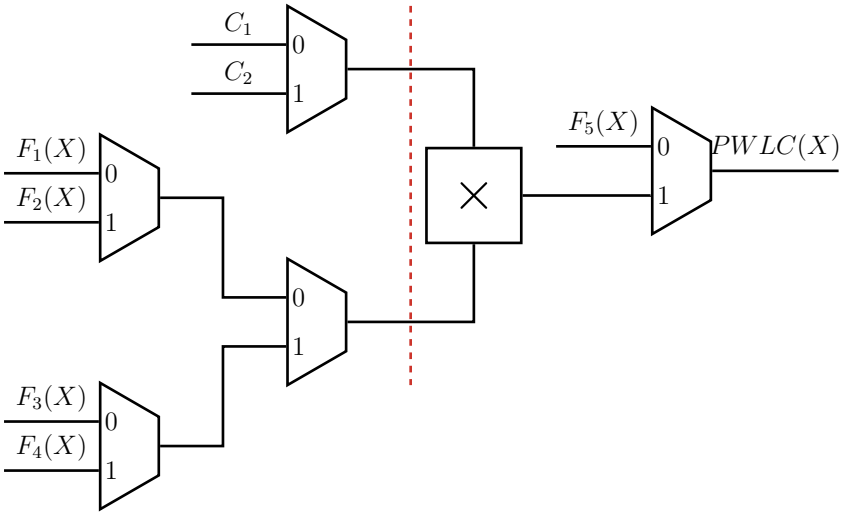


Figure 4.2 – Simplified RTL representation of the PWLC map implementation composed of 1 multiplier and 5 multiplexers. The position of the pipeline register is represented in red.

computing Y^2 and is the reference to the other stages resulting in only four stages of the pipeline:

1. compute recursive cells,
2. compute the first part of the chaotic maps:
   - compute Y for the 4D map,
   - compute and select the inputs of the multiplier for the PWLC map,
3. compute the end of the cells,
4. compute the last LFSR, the output function and the coupling matrix.

At initialization, the generator uses an identical key and 4 different IVs. Once the pipeline stages are filled, the recursive cells start updating the samples. Therefore, the inputs of the key of the maps is defined as follows:

$$X\text{IN}_{MP}(n) = \begin{cases} IV_{MP} + \sum_{i=1}^{3} Ki_{MP}\, X_{MP}(-i) & \text{if } n = \{0;1;2;3\} \\ \sum_{i=1}^{3} Ki_{MP}\, X_{MP}(n-i-3) & \text{otherwise} \end{cases}, \tag{4.3}$$

where MP is set to $4D$ for the 4D map and P for the PWLC map. $X\text{IN}_{MP}(n)$ is the input of the map MP at iteration n, Ki_{MP} are the coefficients of the MP map filter and $X_{MP}(n-i)$ are the delayed sample of the filter.

The output of the PCNG being changed, a statistical analysis of it must be performed.

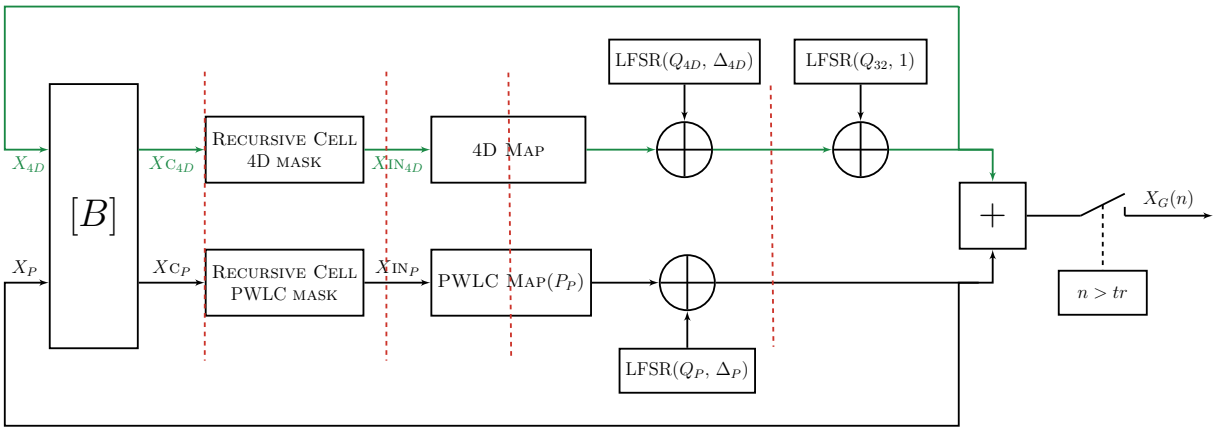


Figure 4.3 – Block diagram of the pipeline LWCB PCNG. In red are represented the different pipeline stages and the critical path in green.

4.2.4 A countermeasure against SCA

As mentioned in the previous chapter, to protect the generator against CPA and DPA attacks [59], masking operations are added to the recursive cells and the coupling matrix. The aim of masking operations is to randomize intermediate results for the same couple (secret key, IV).

To preform masking in a hardware implementation, a source of randomness needs to be added. This less secured Pseudo-Random Number Generator (PRNG) will be used to mask intermediary results. In our case, XOR shift [67] is chosen and defined in Algorithm 4.1.

Algorithm 4.1: Implementation of XOR shift PCNG.

$x \leftarrow x \oplus (x << 16 \mod 2^{32})$
$x \leftarrow x \oplus (x << 5 \mod 2^{32})$
$x \leftarrow x \oplus (x << 1 \mod 2^{32})$
$t \leftarrow x$
$x \leftarrow y$
$y \leftarrow z$
$z \leftarrow t \oplus x \oplus y$
return z

where x, y, z are 32-bit words, initialized with 96-bit IV.

To reverse the mask, the random value must go through the same computation step as the masked value. Then, in our case, the masking requires to add two more recursive cells and one coupling matrix. The hardware resources used by the masked implementation

increase compared to the initial version, however, the throughput will slightly change. Figure 4.4 illustrates how the masking operation is added to the implementation.

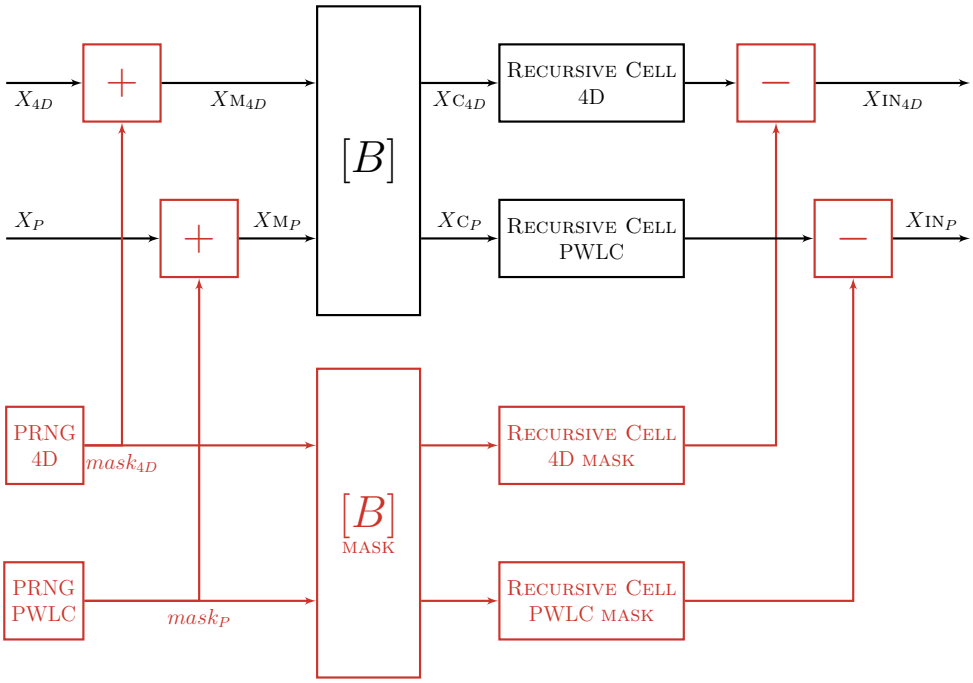


Figure 4.4 – Block diagram of the masking operation on the LWCB PCNG. In red are represented the block added to perform the masking.

4.3 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 of resource usage and speed, the Xilinx Vivado 2018.2 set of tools is used. All measurements are performed after the placement/routing step on a Kintex 7. The hardware resource usage is expressed with Block RAMs (BRAMs) and slices which is usually a group of Lookup Tables (LUTs), Flip-Flops (FFs), registers and Digital Signal Processing (DSP) blocks. A LUT is a block that performs a combinatorial function. A DSP block is block specialized in computing Multiply–ACcumulate (MAC) operations. In these experiments, the bit depth N is set to 32 bits.

4.3.1 Implementation of the LWCB SC

Table 4.1 gives the hardware resources used for the PCNG. It uses in total 2 737 LUTs, 665 registers and 96 DSP blocks. Most of the resources are used by the map cells with 938 LUTs, 189 registers and 33 DSP blocks for the PWLC cell and 973 LUTs, 180 registers and 44 DSP blocks for the 4D one.

Table 4.1 – Hardware resource usage of the proposed masked implementation without pipeline

Component	Slices	LUTs	Registers	DSP Blocks
LWCB SC	797	2737	665	97
Matrix	49	160	2	10
4D Cell	303	973	180	44
4D Map	93	340	–	20
4D perturbation LFSR	39	81	48	–
4D Recurssive Cell	96	270	67	12
4D Recurssive Cell mask	91	283	65	12
4D LFSR	24	41	32	–
PWLC Cell	306	938	189	33
PWLC Map	45	102	–	9
PWLC perturbation LFSR	48	88	57	–
PWLC Recurssive Cell	186	542	67	12
PWLC Recurssive Cell mask	84	239	65	12
4D mask random	37	176	97	0
PWLC mask random	35	177	97	0
Matrix mask	51	162	2	10

The proposed implementation produces one sample of the PCNG at each clock cycle and can operate at 21.05 MHz with a throughput of 673.68 Mbps.

4.3.2 Comparison with other existing stream ciphers

Table 4.2 presents a comparison with existing implementations of the ciphers presented in Section 4.1 and the different versions of our stream cipher.

It is difficult to compare the ratio speed/surface on different implementations using different platforms generated using different tools. To have a fair comparison between all implementations, the best would be to implement and optimize each cipher on the same platform. However, this is not possible in the limited time of a thesis.

This section proposes a comparison taking this problematic into account.

The LWCB SC implementation with masking used nearly twice the hardware resources of the implementation without masking. However, the masking operation is not tanking

Table 4.2 – Speed performance and hardware resources usage comparison of several systems

Cipher	Device	Slices	BRAM	LUTs	FF	Registers	DSP Blocks	Max Freq (MHz)	Throughput (Mbps)
LWCB SC	**Kintex 7**	441[1]	–	1 489	–	337	63	22.22	711.11
LWCB SC - 4 stages of pipeline	**Kintex 7**	611[1]	–	1 704	–	1 249	72	80	2 560
LWCB SC - Recursive cells masked	**Kintex 7**	791[1]	–	2 611	–	663	87	21.74	695.65
LWCB SC - Coupling matrix and Recursive cells masked	**Kintex 7**	797[1]	–	2 737	–	665	97	21.05	673.68
Lorenz's chaotic system [57]	Virtex II	1 926[2]	–	2 718	791[2]	–	40	15.598	124
Lü's chaotic system [58]	Virtex II	1 115[2]	–	1 926	885[2]	–	40	22.868	182.9
AES [55]	Virtex V	–	–	9 276	161[3]	255	–	644.33	82 470
	Spartan 6	–	–	9 375	146[6]	256	–	886.64	113 500
AES [56]	Spartan 2	222[4]	3	–	–	–	–	60	166
Rabbit [62]	Virtex V	568[3]	–	–	–	–	24	71.582	9 160
Trivium [66]	Spartan 3	344[5]	–	–	–	–	–	211	13 504
Trivium [65]	Spartan 3	388[5]	–	–	–	–	–	190	12 160
Salsa20 [64]	Spartan 3	2 036[5]	–	3 374	–	1 286	–	19.4	1014
	Spartan 6	818[6]	–	2 955	–	1 317	–	48	2519

[1] Kintex 7: 1 Slice = 4 LUTs = 8 FFs.
[2] Virtex II: 1 Slice = 2 LUTs = 2 FFs.
[3] Virtex V: 1 Slice = 4 LUTs = 4 FFs.
[4] Spartan 2: 1 Slice = 2 LUTs = 2 FFs.
[5] Spartan 3: 1 Slice = 2 LUTs = 2 FFs.
[6] Spartan 6: 1 Slice = 8 LUTs = 16 FFs.

the throughput much, going from 711.11 Mbps to 673.68 Mbps. The proposed pipelined implementation can achieve a throughput of 2 560 Mbps at an operating frequency of 80 MHz and a latency of 4 clock cycle.

Our stream cipher, easily challenges other chaos-based ciphers in terms of hardware resources usage and throughput. However, more conventional ciphers tend to use fewer hardware resources to achieve the same throughput.

It can be noted that the fastest cipher is AES [55]. The AES implementations presented in [55, 56] operate at 644.33 MHz and 886.64 MHz and the throughput is equal to 82 470 Mbps and 113 500 Mbps, respectively. Even though, Trivium [66, 65] reaches a high frequency the throughput is not better than other slow frequency ciphers. For example, Salsa20 [64] implementations present a throughput of 1 014 Mbps and 2 519 Mbps with an operating frequency of 19 MHz and 48 MHz, respectively. Rabbit [62] implementation reaches a throughput of 9 160 Mbps with an operating frequency of 71.582 MHz. These frequencies are the same as those achieved by the implementations of ciphers based on chaotic key generators [57, 58]. However, these ciphers are still, in terms of throughput

less efficient than all the other implementations, as they have a 124 Mbps and 182.9 Mbps throughput.

A correlation between the speed performance and the hardware resource usage can be noted. In fact, the implementation of AES [55] which is the fastest is also the one using the most resources. This is also the case with the implementations of Rabbit [62] and Salsa20 [64], which use approximately 3 000 LUTs for a quite important throughput. Moreover, the implementation using fewer resources is the compact implementation of AES [56], it uses 444 LUTs to achieve a throughput of 166 Mbps. With less than twice the size, Trivium's [66, 65] implementations, with only 688 and 756 used LUTs, achieve a throughput of 13 505 Mbps and 12 160 Mbps.

For the implementations of the ciphers based on a chaotic generator [57, 58], 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.

4.3.3 Side-Channel Attack (SCA) analysis on LightWeight Chaos-Based (LWCB) stream cipher hardware implementation

In this section, a leakage assessment of the proposed hardware implementation is performed using the methodology proposed by Schneider and Moradi [60, 68]. The methodology is based on the Welsh's t-test used. A potential leakage occurs with the implementation when the result of a non-specific t-test exceeds 4.5. To perform a non-specific t-test, two sets of traces need to be generated, one with a fixed IV and one with a random IVs. This complementary short study was conducted in collaboration with Noafumi Homma from the Research Institute of Electrical Communication at Tohoku University.

To perform the power measurement, the SAKURA-X/SASEBO-GIII [69] evaluation board is used. This evaluation board, developed within the projects SASEBO and its successor SAKURA, is equipped with two different FPGAs, a Kintex 7 where the target implementation is implemented and a Spartan 6 handling the communication between the target implementation and the host PC. These two FPGAs are powered using two isolated power circuits allowing relatively clean power measurements.

Figure 4.5 presents the setup we used. The evaluation board is connected to a 2-channel numerical oscilloscope. The first channel is used to detect the trigger signal indicating the start of the encryption from the target implementation. The second one is used to mea-

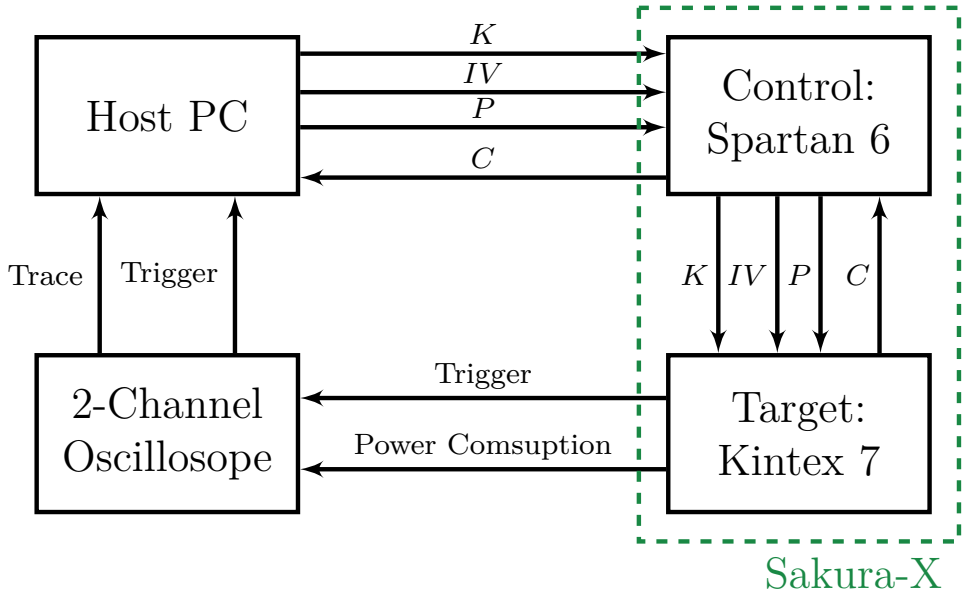


Figure 4.5 – Experimental setup for SCA measurement. K is the secret key, P is the plaintext and C is the ciphertext.

sure the power consumption of the Kintex 7, i.e. the target implementation, using one of the measurement points provided by the SAKURA-X board. This power consumption is usually amplified to improve the measurement quality. The host PC is connected to the evaluation board through Spartan 6 FPGA to handle the inputs of the target implementation and is also connected to the oscilloscope through the serial port to save the traces.

The host PC will execute the following steps to measure one trace for the t-test:

1. set the target with a given key and a fixed plaintext P,
2. set the target in a completely random state by running the target with a random IV for a random time, for example.
3. measure with a fixed IV:
   (a) set the IV,
   (b) wait for the trigger signal,
   (c) recover the trace form the oscilloscope,
   (d) save the trace;
4. set the target in a random state,
5. measure with a random IV.

In this section, the t-test is performed on 1 000 000 traces with 1 000 points per trace. T-tests are performed on three different configurations. The first configuration is the system without masking, the second is the system with only the recursive cells masked and finally the coupling matrix and the recursive cells are masked. Then, a zoom-in proposed by studying only a system with the coupling matrix and the recursive cells in the same previous configurations.

Figure 4.6 represents the results of non-specific t-tests on the different configurations. The first two graphs, representing the t-test results without masking, show that the limit of the null hypothesis ($|t| > 4.5$) is reached during almost all the execution, meaning that the proposed implementation without masking is leaking. As shown by third and fourth graphs, the recursive cell masking has a positive impact on the t-test results. The amplitude of the t value is divided by 2 thanks to the addition of recursive cell masking. Then fifth and sixth graphs show the impact of the masking of the coupling matrix. The impact is less significant than the recursive cell masking. However a slight improvement can be observed on the last graph.

While the proposed implementation is not passing the t-test, we can observe that the implemented masking operation is quite effective but still not enough. The masking operation, as implemented, is still leaking as shown by the last graph of Figure 4.6. A gate-level implementation of the masking operation might be a good solution to remove at least first order leakage. However, even with a perfect masking operation, the implementation still has other problems to assess. The masking operation was performed only on the linear part of the system and is not protecting the implementation from leaking. A method of protection against SCAs for chaotic maps should be deeply investigated. From the observations of Figure 4.6, the leakage of the implementation seems to come from the chaotic maps. Moreover, the chaotic maps used in ciphers are usually non-linear or partly linear and are not maskable using common masking operation.

4.4 Conclusion

In this chapter, several implementations of the proposed lightweight chaos-based stream cipher are presented and compared to state-of-the-art stream-cipher implementations. The first implementation is the stream cipher described in the previous chapter without any countermeasures against SCAs. This achieves a throughput of 711.11 Mbps at an operating frequency of 22.22 MHz using only 441 slices. The second implementation

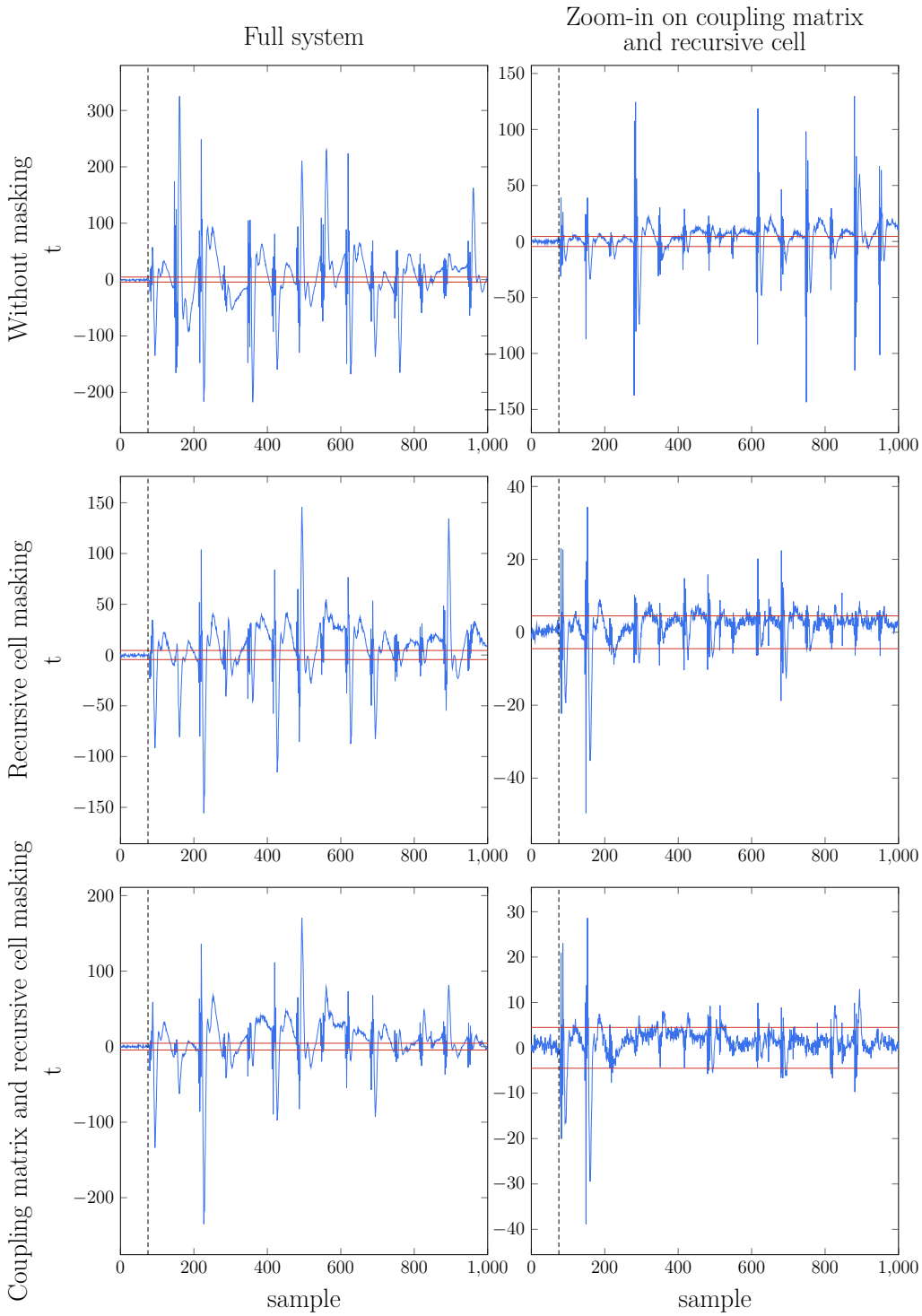


Figure 4.6 – Non-specific t-test results for the proposed implementation with and without masking. The dashed line represents the start of encryption and the red lines are the limit if the t-test is admissible.

presented is a 4-stage-pipeline version. It operates at 80 MHz achieving a throughput of 2 560 Mbps with 611 slices. Finally, a masked implementation of the first one is proposed. This implementation slightly degrades the speed performance with a throughput of 673.68 Mbps operating at 21.05 MHz using 797 slices.

Then, the security against SCAs is assessed using the methodology proposed by Schneider and Moradi [60, 68]. The results of the non-specific t-test showed that with and without masking, the proposed implementations are leaking. A gate-level implementation might be a solution to the leakage of the coupling matrix and recursive cell masking. This results also exposed the need for investigating method in order to mask chaotic maps.

CHAPTER 5

Selective Encryption of the Versatile Video Coding (VVC) standard

5.1 Introduction

Security and confidentiality of multimedia contents are of prominent importance in many applications to ensure safe storage and transmission of images and videos. The straightforward solution to perform secure transmission of a video is to encrypt the whole video bitstream with a secure encryption protocol such as Advanced Encryption Standard (AES) [70]. However, this solution when applied to video has several limitations related to their high computational complexity increasing both the energy footprint and end-to-end latency. This increase in complexity/latency is mainly caused by the processing complexity of the encryption algorithm used to cipher the whole video especially when the video is encoded at high bitrate. Moreover, the deciphering and ciphering processes are required to perform post-processing operations such as transcoding for network adaptation. This may harm security since the secret key is shared with untrusted middlebox in the network to perform splicing, quality monitoring, watermarking and transcoding. The selective encryption solution has emerged as an effective alternative to perform secure and low complexity encryption of images and videos. The encryption process is performed in the compressed-domain where only a set of the most sensitive information is encrypted. This enables performing both format-compliant and constant bitrate encryption. The format-compliant property is very important enabling to decode the video bitstream without

deciphering and thus all post-processing operations can be performed including packaging and transcoding without requiring access to the secret key used for encryption. Moreover, this property enables encrypting only some spatial regions in the image identified as Region Of Interest (ROI) while keeping the rest of the image clear. The constant bitrate property preserves the encoder coding efficiency. Selective encryption has been widely investigated for different still image and video coding standards including JPEG [71], JPEG-2000 [**liu2006efficient**, 72], Advanced Video Coding (AVC) [73], Scalable Video Coding (SVC) [74] and more recently High Efficiency Video Coding (HEVC) [75, 76, 77] and its scalable extension SHVC [44]. Selective encryption of the HEVC standard has been widely investigated in the literature [78, 79, 80, 81] enabling format-compliant, secure and low complexity encryption.

The ISO/Moving Picture Experts Group (MPEG) and ITU/Video Coding Experts Group (VCEG) developed the next generation video coding standard called Versatile Video Coding (VVC). This latter, released in July 2020, introduces new coding tools outperforming HEVC by up to 50% in terms of bitrate reduction for a similar visual quality [82]. To the best of our knowledge, format-compliant and constant bitrate encryption of VVC has not yet been addressed. Moreover, it is well known from information theory [83] that enhancing the coding efficiency adds more dependencies in the bitstream making format-compliant and constant bitrate encryption more challenging.

This chapter investigates a format-compliant and constant bitrate encryption of a video bitstream encoded with the VVC standard. To meet these two constrains, the encryption is performed at the level of the Context-Adaptive Binary Arithmetic Coding (CABAC) engine. We first investigate all possible syntax elements that can be encrypted in both format-compliant and constant bitrate. A set of VVC syntax elements including Transform Coefficient (TC) values and signs, chroma prediction candidate, Motion Vector (MV) differences and signs are encrypted. We propose a new algorithm that determines the encryptable bins within the TCs. The proposed selective encryption solution has been extensively assessed under the VVC Common Test Conditionss (CTCs) using three image and video quality assessment metrics including Peak Signal to Noise Ration (PSNR), Structural SIMilarity (SSIM) and Video Multimethod Assessment Fusion (VMAF), and security metrics such as Encryption Qualilty (EQ) [84], histogram analysis, edge detection and Edge Differential Ratio (EDR) [85]. The proposed solution has also been tested against brute force attack, Number of Pixels Change Rate (NPCR) and Unified Average Changing Intensity (UACI) [86]. The encryption space giving the percentage of encrypted bits in the

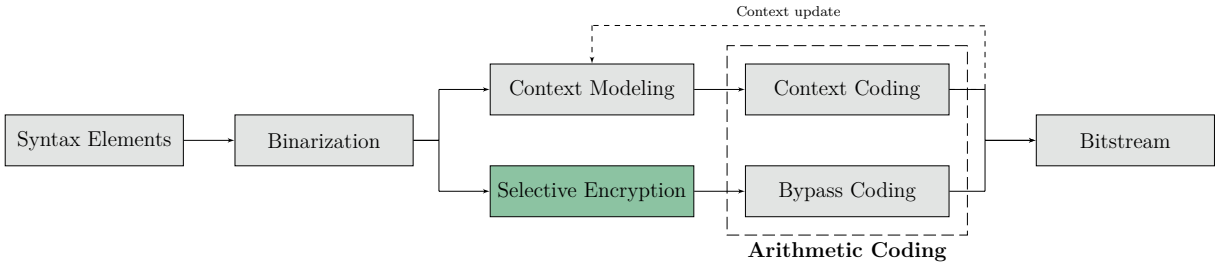


Figure 5.1 – Overall architecture of the Context-Adaptive Binary Arithmetic Coding (CABAC) engine in Versatile Video Coding (VVC). The selective encryption block is illustrated in green

bitstream varies in the range of 15% to 26% for different targeted bitrates. This results in a very low decryption complexity which remains lower than 6% of the decoding time.

The rest of this chapter is organized as follows. Section 5.2 gives a brief review on selective encryption solutions proposed for HEVC and then Section 5.3 describes the entropy coding of syntax elements in VVC. Section 5.4 presents the proposed solution to encrypt VVC syntax elements in format-compliant and constant bitrate. The performance of the selective encryption solution is assessed in Section 5.5 in terms of video quality degradation, resilience to different attacks and complexity overhead. Finally, Section 5.6 concludes this chapter.

5.2 Related Works

In this section, we review the existing solutions for High Efficiency Video Coding (HEVC) standard encryption. The first format-compliant encryption solution of HEVC was proposed by Shahid *et al.* [75]. In this solution, Advanced Encryption Standard (AES) was used in Cipher FeedBack (CFB) mode to perform selective encryption of the selected syntax elements at the CABAC stage. This work considered an earlier version of the HEVC standard and some encryptable syntax elements of HEVC were not identified in this solution. Farajallah *et al.* [76] proposed a selective encryption solution to cipher the Region Of Interest (ROI) in HEVC standard. This solution relies on the tile concept introduced in HEVC enabling a frame partitioning into independent rectangular regions. The encryption process encrypts only tiles within the ROI and keeps the background clear. The tiles within the ROI are encrypted in format-compliant and constant bitrate by ciphering only a set of sensitive syntax elements. Moreover, to prevent the encryption propagation

outside the ROI, Motion Vectors (MVs) of the background tiles are constrained to only refer to background area (no ROI) in the reference frames. Boyadjis *et al.* [77] presented a selective encryption algorithm in order to increase the visual distortion. The presented research moves selective encryption from bypass mode to regular mode, which negatively affects the bitrate. Luma intra prediction modes are selected to be encrypted in addition to the residuals. The presented solution enables more scrambling performance while the compression efficiency has changed leading to a slight bitrate increase. Hamidouche *et al.* [44] investigated a selective encryption of the final version of HEVC. The authors have proposed a real time selective encryption solution for the scalable extension of HEVC named Scalable HEVC (SHVC). The presented solution has analyzed all SHVC syntax elements in order to perform format-compliant, constant bitrate and low latency encryption while preserving all SHVC features. The presented results showed the high security level of the selective encryption solution with a low complexity overhead below 6% of the decoder complexity. Van Wallendael *et al.* [78] presented a format-compliant selective encryption solution for HEVC. They selected a set of syntax elements from HEVC that preserve the format compliance. Several techniques to selectively encrypt the video are investigated. The obtained result showed that most of the selected syntax elements have a low effect on the rate-distortion performance while having a broad range in scrambling performance. Memos *et al.* [79] presented an algorithm that encrypts only Intra frame (I frame) frames of the HEVC bitstream based on the idea that Predicted frame (P frame) and Bidirectional predicted frame (B frame) frames are useless without I frame frame. Moreover, encrypting only I frame frames will decrease the encryption time by 50% and propagates the encryption to other frames. The presented algorithm merged two algorithms proposed in [87, 88], while introducing some modifications to the selection and management of the encrypted data to be amendable to HEVC. This work relies on the AES algorithm for secure transmission of HEVC bitstream with 256 bits as key length. It collects sign bits of each transform coefficient of I frame frames until the collected signs reached 256 bits. However, it is not clear in the proposed algorithm whether the collected bits are used as key value or as state value since AES-256 state size is 128 and not 256. The proposed algorithm performs conventional AES encryption on the collected bits and swap the original sign bits by the encrypted ones. Finally, the Shamir's Secret Sharing (SSS) input parameters are collected from the non-zero Alternating Current (AC) coefficients of each transform block within the I frame frame. It is clear from the description that the proposed algorithm performs partial encryption algorithm. It is also important

to note that the proposed solution is not format-compliant, none constant bitrate since it increases the bitstream size at least by 8%. Long *et al.* [80] presented a format-compliant encryption in order to secure HEVC streams in multimedia social networks. The presented algorithm is tightly integrated with the encoding/decoding processes. The presented work performs encryption in two steps. First, a stream cipher is used to encrypt sign of the nonzero Transform Coefficients (TCs), and the first sign bit hiding of TCs. Second, based on a control factor, only one parameter from merging index, MV prediction index, sign of MV difference and reference frame index is encrypted. The presented research increases the bitrate, while it is format-compliant solution. Finally, the presented work was assessed regarding security and complexity which confirms the good security level and acceptable complexity overhead. Ahmed *et al.* [81] presented a new solution for efficient selective encryption based on the chaotic logistic map for HEVC. The presented solution encrypts the sign bit of the MV differences and the TCs. The encryption process is performed at the entropy coding stage of the HEVC encoding process. They focused on achieving a low complexity ciphering targeting real time applications, constant bitrate and format-compliant encryption. The presented work was compared with the solution proposed in [78] and the obtained results confirm the suitability for real time applications with an intermediate level of security. Peng *et al.* [89] presented a tunable selective encryption scheme for HEVC based on chroma Intra Prediction Mode (IPM) and TCs scrambling. The presented work has two security levels. The first one encrypts HEVC syntax elements including Luma IPMs, Chroma IPMs, the suffix part of the TCs, sign and value of the MV differences, merge index, advanced MV prediction, reference frame index, and Sample-Adaptive Offset (SAO) filter parameters. The second security level relies on edge extraction of each transform block. The transform block coefficients are scrambled to increase the security level only when the current transform block contains edges. Finally, the AES is used in CounTeR (CTR) mode in order to generate the pseudo-random number sequences. These sequences are used to encrypt all previously mentioned parameters with a simple eXclusive OR (XOR) operation. Xu [90] proposed to perform data hiding inside the selected encrypted bitstream of HEVC. The secret message is hidden using a Quantized Transform Coefficient (QTC) modification technique. It only changes bits value based on the data hiding without changing the data size in bypass coding, which confirms that the obtained solution is constant bitrate and format-compliant. Since the used operation is a XOR, the extraction process of the hiding data can be achieved on both encrypted as well as original videos. Obtained results confirm the resilience of

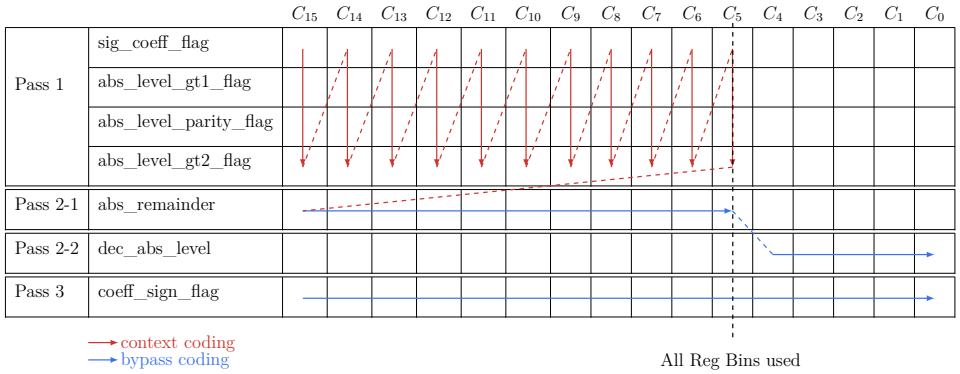


Figure 5.2 – Binarization of the Transform Coefficients (TCs) of a 4×4 sub-block in TC mode JVET-S2002 [91].

the presented work against replacement attacks. Moreover, the degradation on the video quality introduced by data hiding is negligible. However, the presented algorithm was not evaluated regarding important general video attacks such as Unified Average Changing Intensity (UACI), Number of Pixels Change Rate (NPCR), Edge Differential Ratio (EDR), Encryption Qualilty (EQ), histogram analysis and key sensitivity attacks.

5.3 CABAC engine in VVC

The CABAC engine defined in VVC is similar to HEVC consisting of three main functions: binarization, context modeling and arithmetic coding [92]. The overall CABAC architecture is illustrated in Fig. 5.1. First, the binarization step converts syntax elements to binary symbols (bins). Second, the context modeling updates the probabilities of bins, and finally the arithmetic coding compresses the bins into bits according to the estimated probabilities.

5.3.1 Binarization methods

Six binarization methods are used in VVC, namely Unary (U), Truncated Unary (TU), Fixed Length (FL), Truncated Binary (TB), Truncated Rice Code with context p (TRp) and Exp-Golomb k-th order code (EGk). The U code represents an unsigned integer B with a binstring of length $B+1$ composed of B 1-bins followed by one 0-bin. The TU code is defined with the largest possible value of the syntax element $cMax$ ($0 \leq B \leq cMax$). When the syntax element value $B < cMax$, the TU is equivalent to U code, otherwise B

is represented by a binstring of $cMax$ 1-bins. The FL code represents a syntax element B with its binary representation of length $\lceil \log_2(cMax+1) \rceil$ with $\lceil x \rceil$ is smallest integer greater than or equal to x. The TB code is similar to the FL code, except when the $cMax+1$ value is not a power of 2. In this case, let k be $k = \lfloor \log_2(cMax+1) \rfloor$ (with $\lfloor x \rfloor$ is largest integer less than or equal to x). The first $u = 2^{k+1} - cMax$ elements are coded with a FL code of length k. The remaining $cMax+1-u$ symbols are offseted by u and coded by $k+1$ bins. The TRp code is a concatenation of a quotient $q = \lfloor B/2^p \rfloor$ and a remainder $r = B - q2^p$. The quotient q is first represented by the TU code as a prefix concatenated with a suffix r represented by the FL code of length p. The EGk code is also a concatenation of prefix and suffix. The prefix part of the EGk code is the U representation of $l(B) = \lfloor \log_2(\frac{B}{2^k}+1) \rfloor$. The suffix part is the FL code of $B + 2^k(1-2^{l(B)})$ with $cMax = k + l(B)$.

5.3.2 Transform Coefficients (TCs) coding

In this section we describe the CABAC coding of the TCs. Similar to HEVC, VVC coefficients are either coded in regular TCs mode or Transform Skip (TS) mode. In both modes, the transform block is first divided into sub-blocks.

5.3.2.1 VVC Transform Coefficient (TC) coding mode

The coefficients of each sub-block are encoded in three passes as illustrated in Fig. 5.2 for a 4×4 sub-block. The coefficients are processed in reverse diagonal scan order, as depicted in Fig. 5.3a. The first pass processes a group of flags until it reaches a limit of used bins specified by the standard. This maximum number of bins used in the first pass is computed with respect to the block size $(W_b \times H_b)$ as follows $\lfloor (2^{\log_2(W_b)+\log_2(H_b)})\, 7/4 \rfloor$. Once this limit is reached, the second pass starts encoding the remainders computed from the coefficient value C as follows

$$abs\_remainder = \begin{cases} \left\lfloor \frac{|C|-4}{2} \right\rfloor & |C| \geq 4, \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

The coefficients of value lower than 4 are binarized by the flags in the first pass. The second pass relies on the TRp/EGk binarization, until the position of the last coefficient processed by the first pass is reached. Then, the *dec_abs_level* syntax element, computed

by (5.2) for the remaining coefficients is bypassed and binarized also using the TRp/EGk binarization.

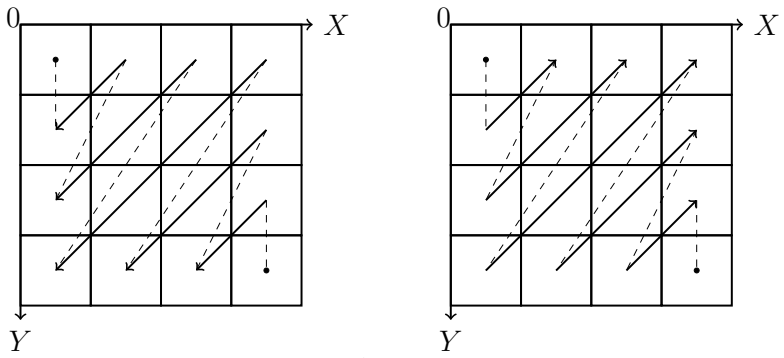
$$dec\_abs\_level = \begin{cases} V, & \text{if } C = 0, \\ |C|, & \text{if } |C| \leq V, \\ |C| + 1, & \text{if } |C| > V, \end{cases} \tag{5.2}$$

the V constant is derived from a Lookup Table (LUT) $VArr$ according to the state and the local absolute sum $LocAbsSum$ computed for the current coefficient by (5.3). The V value updates the 0 coefficient value such that coefficients have smaller binarization when large coefficients are mixed with definite 0 values. The $LocAbsSum$ is a saturated sum in the interval $[0, 31]$ of a set of neighboring coefficients S_1 illustrated in green in Fig. 5.4a

$$LocAbsSum = \left[\sum_{i \in S_1} |C_i| - 5\, BaseLvl \right]_0^{31}, \tag{5.3}$$

where $BaseLvl$ is equal to 4 for the *abs_remainder* (Pass 2-1) and 0 for the *dec_abs_level* (Pass 2-2).

Finally, the third pass encodes the signs of the coefficients. We can notice that only the first pass relies on CABAC context coding and the last two passes perform bypass coding. The *abs_remainder* and *dec_abs_level* syntax elements are both binarized by a combination of TRp and in a special case EGk code. This binarization is presented in Section 5.3.2.3.



(a) (b)

Figure 5.3 – Transform Coefficients (TCs) scanning orders (5.3a) reverse diagonal scan order and (5.3b) regular diagonal scan order.

5.3.2.2 VVC Transform Skip (TS) coding mode

In Transform Skip mode, the coefficients of each sub-block are also encoded in three passes that process the coefficients in a simple diagonal scan order, as shown in Fig. 5.3b. The first pass mainly encodes all coefficients considered as significant (ie. $C \neq 0$) including its sign and parity. The second pass encodes more flags to check whether the coefficient is greater than a certain threshold. Finally, the third pass encodes the remainder coefficients greater than 10 using the TRp/EGk binarization of $abs\_remainderTS$

$$abs\_remainderTS = \left\lfloor \frac{|C| - 10}{2} \right\rfloor. \tag{5.4}$$

The local absolute sum in the case of TS mode $LocAbsSumTS$ is computed by (5.5) as follows

$$LocAbsSumTS = \left[\sum_{i \in S_2} |C_i| \right]_0^{31}. \tag{5.5}$$

It should be noted that the third pass relies on bypass coding.

5.3.2.3 Binarization process

Algorithm 5.1 gives the binarization process of the $abs\_remainder$. The dec_abs_level and $abs\_remainderTS$ syntax elements are also binarized by this algorithm. The TCs remainders are binarized using either a TRp code, introduced in Section 5.3.1, or an EGk code limiting the maximal length of a binarization to 32 bits as presented in Algorithm 5.2. The selection between the two binarizations depends on a threshold value β defined in the standard as

$$\beta = BinReduc\ 2^{cRiceParam}, \tag{5.6}$$

where $BinReduc$ is set to 5, and $cRiceParam \in \{0, 1, 2, 3\}$ is the rice parameter derived from a LUT $riceArr$ according to the saturated local absolute sum $LocSumAbs$ of previously coded coefficients computed by (5.3). Fig. 5.4a illustrates in green the set of coefficients S_1 used to derive the rice parameter of the current coefficient highlighted in yellow. Similarly, Fig. 5.4b presents the coefficients used in TS mode, where the rice parameter depends only on the top and left neighbor coefficients set S_2. When the remainder to encode is strictly below the threshold β, the TRp binarization is preformed with $p = cRiceParam$. Otherwise, the limited EGk coding is applied.

Algorithm 5.1: *abs_remainder* Binarization

Input: *abs_remainder* is the unsigned integer to binarize
cRiceParam is the rice parameter
log2TrRange is the *log2* of the TC range
$BinReduc \leftarrow 5$ is the value used to determine the threshold between TRp and *Limited_EGk*

$\beta \leftarrow BinReduc\ 2^{cRiceParam}$
if $abs\_remainder < \beta$ **then**
  TRp binarization with $p \leftarrow cRiceParam$
else
  $Limited\_EGk(abs\_remainder, cRiceParam,$
    $BinReduc, log2TrRange)$
end if

Algorithm 5.2 shows that the maximum length of the prefix $maxPrefixLen$ depends on the range of the transform coefficients $2^{log2TrRange}$ and $BinReduc$. To differentiate between the two binarizations at the decoder side, $BinReduc$ is added to the prefix length when *Limited_EGk* is used. Then, a classical EGk binarization starts. However, if the computed prefix length $prefixLen$ is equal to the maximal prefix length $maxPrefixLen$, the suffix length $suffixLen$ is set to $log2TrRange$. Both codes are composed of a variable-length $prefix$ and if exists, a fixed-length $suffix$. The prefix is coded using a U or TU code representation which implies that changing any bin will violate the decoder standard or change the bitrate. On the other hand, the suffix might be encrypted in format compliance and constant bitrate only when the $LocSumAbs$ does not change the $cRiceParam$ value of the neighbor coefficients.

5.4 Proposed Versatile Video Coding (VVC) selective encryption

This section presents a new selective encryption scheme for VVC standard. The proposed selective encryption fulfills two important features: standard format-compliant encryption (i.e. the bitstream must be decodable by any VVC decoder) and constant bitrate encryption (i.e. preserve the VVC compression efficiency).

The encryption is performed at the Context-Adaptive Binary Arithmetic Coding (CABAC) level of the encoder. Fig. 5.1 depicts in green the position of the selective

Algorithm 5.2: $Limited\_EGk(abs\_remainder, cRiceParam, BinReduc, log2TrRange)$

Input: $abs\_remainder$, $cRiceParam$, $log2TrRange$, $BinReduc$.

$maxPrefixLen \leftarrow 32 - BinReduc - log2TrRange$
$codeValue \leftarrow \left\lfloor \frac{abs\_remainder}{2^{cRiceParam}} \right\rfloor - BinReduc$
$prefixLen \leftarrow 0$
while $prefixLen < maxPrefixLen$
and $codeValue > 2^{prefixLen+1} - 2$ **do**
  $prefixLen \leftarrow prefixLen + 1$
end while
if $prefixLen = maxPrefixLen$ **then**
  $suffixLen \leftarrow log2TrRange$
else
  $suffixLen \leftarrow prefixLen + cRiceParam + 1$
end if
$totalPrefixLen \leftarrow prefixLen + BinReduc$
$bitMask \leftarrow 2^{cRiceParam}$
$prefix \leftarrow 2^{totalPrefixLen} - 1$
$suffix \leftarrow codeValue - 2^{prefixLen} - 1$
$suffix \leftarrow suffix\ 2^{cRiceParam} + abs\_remainder \mod bitMask$
//where $a \mod n$ gives the remainder of the euclidean divison of a by n

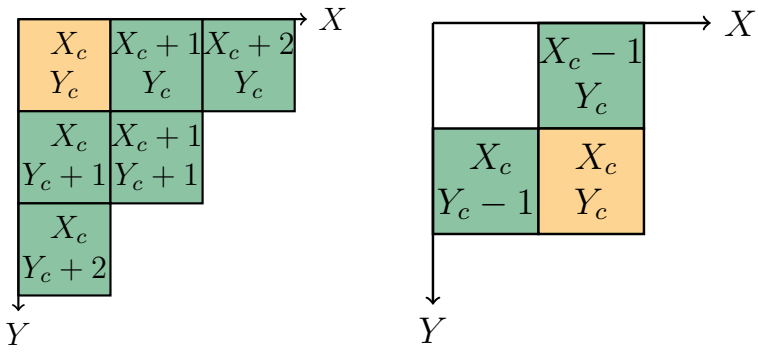
encryption in the CABAC engine. The encryption is performed after the binarization process, and only a set of selected syntax elements, listed in Table 5.1 are ciphered. The encryption involves syntax elements from different coding tools including transform block, intra and inter predictions, and in-loop filters. This ensures the encryption of both intra (Intra frame (I frame)) and inter (Predicted frame (P frame) and Bidirectional predicted frame (B frame)) coded slices included in the VVC video sequence.

The syntax elements, listed in Table 5.1, have been selected based on following two criteria:

— The syntax element is bypassed: this restriction preserves the VVC coding efficiency.

— Changing any bin will not change how the binstring is read by the decoder: this restriction ensures format-compliant encryption by excluding most of flags and syntax elements binarized by variable length codes.

The encryption of the most syntax elements listed in Table 5.1 is straightforward except the Transform Coefficients (TCs) that requires a specific processing to search for the



(a) (b)

Figure 5.4 – TCs dependencies: coefficients highlighted in green are used to compute the local absolute sum of the current coefficient in yellow for (5.4a) TC mode and (5.4b) TS mode. S_1 and S_2 are two sets of green coefficients in (5.4a) and (5.4b), respectively.

encryptable bins. In the next section, we describe the encryption of the TCs since it is the most challenging syntax element to encrypt. The coding of the TCs introduces dependencies that need to be carefully addressed to perform format-compliant and constant bitrate encryption.

5.4.1 Transform Coefficient encryption

This section presents how the TCs are encrypted. As explained in Section 5.3.2.3, the binarization of the TCs and especially the length of the suffix depends on the previously encoded TCs. Therefore, encryption that changes the value of the coefficients may introduce bitrate increase. Indeed, the binarization depends on a rice parameter $cRiceParam \in \{0, 1, 2, 3\}$ derived from previous TCs. This rice parameter defines the fixed length of the suffix and therefore it corresponds to the size of the encryptable bins. After an analysis of the binarization algorithm, multiple conditions ensuring constant bitrate have emerged and are presented below.

First, it is important to note that the coefficients are binarized in two different ways depending on whether they are processed by pass 2-1 or 2-2, as presented in Fig. 5.2.

— *The encryption **must not** change the parity of the coefficient:* changing the parity will result in changing the *state* of the CABAC context. The state is updated using the previous state value and the parity of the current coefficient.

— *The encryption **must not** change the rice parameter:* this will affect the bitrate.

Table 5.1 – Encrypted syntax elements in the proposed VVC selective encryption solution, all these syntax elements are bypass coded.

Coding block	Syntax elements	Binarization
Transform coefficients (TCs)	abs_remainder, dec_abs_level	Truncated Rice Code with context p (TRp), Exp-Golomb k-th order code (EGk)
	abs_remainderTS	TRp,EGk
	coeff_sign_flag	Fixed Length (FL)
	coeff_sign_flagTS	FL
Motion Vector (MV)	abs_mvd_minus2	EGk
	mvd_sign_flag	FL
ALF Filter	alf_luma_fixed_filter_idx	Truncated Binary (TB)
Inter Prediction	mmvd_direction_idx	FL
	merge_triangle_split_dir	FL
Sample-Adaptive Offset (SAO) Filter	sao_offset_sign	FL
	sao_band_position	FL
	sao_eo_class	FL
Intra Prediction	intra_chroma_pred_cand	FL

— *The encryption **must not** change the V value for coefficients processed by pass 2-2:* changing the value of this parameter can result in changing the parity, and thus the CABAC context.

Considering those conditions, Algorithm 5.3 is proposed to identify the bins within the TCs that can be encrypted in constant bitrate and format compliance. The rice parameter of each TC is derived from a saturated absolute sum of the local neighborhood of the current TC. Fig. 5.5a depicts in green the set $\bar{S}_1$ of affected TCs if the current TC in yellow is modified by encryption. Therefore, for each affected coefficient, Algorithm 5.4 checks whether the changes in the local absolute sum will affect the context, the rice parameter and the V value. To make sure that the parity is not changed, the encryption excludes the Less Significant Bits (LSB) of the suffix and will perform encryption only when the rice parameter is greater than 1 ($cRiceParam \in \{2, 3\}$).

The encryption of the coefficients in TS mode is similar. The main difference lies in how the rice parameter is derived. Fig. 5.5b shows the set $\bar{S}_2$ of affected coefficients in green when the current coefficient (in yellow) is modified by encryption. The current coefficient is binarized using a prediction based on the top and left coefficients. Therefore, the ciphered value of the current coefficient must remain lower or equal than the coefficient depicted in red in Fig. 5.5b according to the used prediction scheme.

Algorithms 5.3 and 5.4 are used to check that the encrypted bins of the current binarized coefficient are not affecting how the neighbor coefficients will be encoded. This

Algorithm 5.3: $nbEncryptable = isEncryptable(X_c, Y_c, CoeffArr, nbEncryptable, bypass, V)$

Input: (X_c, Y_c): the coordinate of the current pixel,
$CoeffArr[][]$: the array of coefficient value,
$nbEncryptable$: the number of encryptable bits to test,
$bypass$: **true** if the current coefficient is bypass,
V: of the current pixel, set to 0 if $bypass$ is **false**.
Output: the number of encryptable bits.

$absLevel \leftarrow |CoeffArr[X_c][Y_c]|$
if $absLevel \neq 0$ **and** $cRiceParam > 1$ **then**
  $absCMin, remMin \leftarrow computeMin(absLevel,$
    $nbEncryptable, bypass, V)$
  $absCMax, remMax \leftarrow computeMax(absLevel,$
    $nbEncryptable, bypass, V)$
  $encryptable \leftarrow$ **not** $(bypass$
    and $V \in [remMin, remMax])$
  for $p \in \bar{S}_1$ **do**
    $encryptable \leftarrow encryptable$ **and**
      $checkSumChange(X_p, Y_p,$
      $absLevel, absCMin, absCMax)$
  end for
  if not $encryptable$ **and** $nbEncryptable > 1$ **then**
    return $isEncryptable(X_c, Y_c, CoeffArr,$
      $nbEncryptable - 1, bypass, V)$
  else if $encryptable$ **then**
    return $nbEncryptable$
  else
    return 0
  end if
else
  return 0
end if

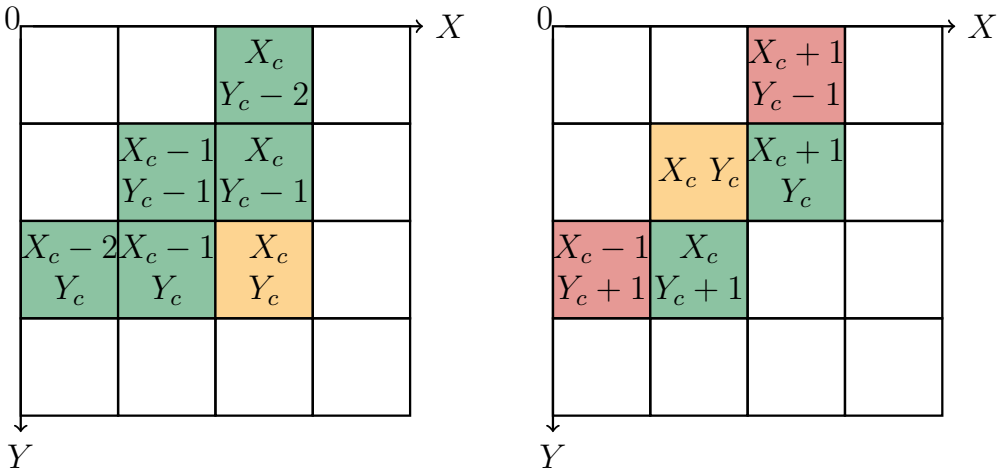
Algorithm 5.4: $Encryptable = checkSumChange(X_p, Y_p, absLevel, absCMin, absCMax)$;

Input: (X_p, Y_p): the coordinate of the tested coefficient ,
$absLevel$: the absolute value of the coefficient (X_c, Y_c) before encryption,
$absCMin$: the minimal possible encrypted value of the coefficient (X_c, Y_c),
$absCMax$: the maximal possible encrypted value of the coefficient (X_c, Y_c).

Output: **true** if ciphered value does not affect the context, the rice parameter and the C_0, **false** otherwise.

//Step 1
$AbsSumP1 \leftarrow \sum_{i \in S_1} \min(4 + |C_i| \mod 2, |C_i|)$
$numPos \leftarrow \sum_{i \in S_1} E(C_i)$ //where $E(x)$ returns 1 if $x \neq 0$, 0 otherwise
$AbsSumP1min \leftarrow AbsSumP1 - \min(4 + (absLevel \mod 2), absLevel)$
$+ \min(4 + (absCMin \mod 2), absCMin)$
$NoCtxChange \leftarrow \left\lfloor \frac{AbsSumP1+1}{2} \right\rfloor \geq 3$ **and** $AbsSumP1 - numPos \geq 4$
and $\left\lfloor \frac{AbsSumP1Min+1}{2} \right\rfloor \geq 3$ **and** $AbsSumP1Min - numPos \geq 4$
//Step 2
$AbsSumP21 \leftarrow \sum_{i \in S_1} |C_i|$
$AbsSumP21Min \leftarrow AbsSumP21 - absLevel + absCMin$
$AbsSumP21Max \leftarrow AbsSumP21 - absLevel + absCMax$
$TrAbsSumP21 \leftarrow [AbsSumP21 - 20]_0^{31}$
$TrAbsSumP21Min \leftarrow [AbsSumP21Min - 20]_0^{31}$
$TrAbsSumP21Max \leftarrow [AbsSumP21Max - 20]_0^{31}$
$RiceParP21 \leftarrow riceArr[TrAbsSumP21]$

$TrAbsSumP22 \leftarrow [AbsSumP21]_0^{31}$
$TrAbsSumP22Min \leftarrow [AbsSumP21Min]_0^{31}$
$TrAbsSumP22Max \leftarrow [AbsSumP21Max]_0^{31}$
$RiceParP22 \leftarrow riceArr[TrAbsSumP22]$
//Step 3
$NoRiceParChange \leftarrow$ **true**
if $TrAbsSumP21Min \notin I_R[RiceParP21]$ **or** $TrAbsSumP21Max \notin I_R[RiceParP21]$
or $TrAbsSumP22Min \notin I_R[RiceParP22]$ **or** $TrAbsSumP22Max \notin I_R[RiceParP22]$
then
$NoRiceParChange \leftarrow$ **false**
end if
//Step 4
$NoVChange \leftarrow$ **true**
for $i \leftarrow 0$ **to** 2 **do**
$currV \leftarrow VArr[i][TrAbsSumP22]$
if $TrAbsSumP22Min \notin I_P[i][currV]$ **or** $TrAbsSumP22Max \notin I_P[i][currV]$ **then**
$NoVChange \leftarrow$ **false**
end if
end for
//Step 5
return $NoCtxChange$ **and** $NoRiceParChange$ **and** $NoVChange$



(a) (b)

Figure 5.5 – The current coefficient (in yellow) is used to compute the local absolute sum of the coefficients highlighted in green for (5.5a) TC mode and (5.5b) Transform Skip (TS) mode. Coefficients highlighted in red are used along the current coefficient in the prediction of the parity of the coefficients in green. $\bar{S}_1$ and $\bar{S}_2$ are two sets of green coefficients in (5.5a) and (5.5b), respectively.

enables defining the bins that can be encrypted in format-compliant and constant bitrate. The proposed solution is carried out as follows:

— Algorithm 5.3 checks at the binarization process whether the TC can be encrypted or not. The encryption is possible only when the absolute value of the coefficient is different from 0 ($absLevel \neq 0$) and the value of the derived rice parameter is above 1 ($cRiceParam > 1$).

— The algorithm then computes the minimum and maximum values of the encrypted remainder ($remMin, remMax$) of the current coefficient. The minimum ($absCMin$) and maximum ($absCMax$) absolute values of the coefficient are derived from their respective remainders.

— Algorithm 5.4 checks for all coefficients in $\bar{S}_1$ depicted in green in Fig. 5.5a (when they exist) whether ciphering the current coefficient (X_c, Y_c) will affect its neighbor coefficients (X_p, Y_p), the rice parameter or the V value. This operation is performed in five steps as follows:

1. The algorithm computes a saturated absolute sum of the tested coefficient of coordinates (X_p, Y_p) ($AbsSumP1 = \sum_{i \in S_1} min(4 + |C_i| \mod 2, |C_i|)$) which

is used for the context computation, with S_1 the set of neighbor coefficients of (X_p, Y_p). This operation is performed at the first pass (P1) to check the CABAC context change and set the $NoCtxChange$ flag to **true** if the changes on the $AbsSumP1$ will not affect the context.

2. The local absolute sum $TrAbsSumP21$ is then computed by (5.3) for the coefficient of coordinates (X_p, Y_p). The minimum possible value $TrAbsSumP21Min$ and the maximum value $TrAbsSumP21Max$ are also computed by (5.3) with $BaseLvl$ equals to 4.
3. Then, the algorithm checks whether the rice parameter will be affected with the different computed sums in step 2 and sets a flag $NoRiceParChange$ to **true** if the rice parameter remains unchanged with all possible tested conditions. I_R is a Lookup Table (LUT) containing, for each rice value, the interval in which the the local absolute sum does not change the rice parameter.
4. The parameter V is computed only in pass 2-2. At this fourth step, the algorithm checks for the processed coefficients if the parameter V remains unchanged to set the flag $NoVChange$ to **true**. Similar to I_R, I_P returns, depending of the state and V values, the interval in which the local absolute sum does not change the V value.
5. Finally, Algorithm 5.4 returns **true** when $NoCtxChange$, $NoRiceParChange$ and $NoVChange$ are all equal to **true**.

The decoder performs inverse operations performed by the encoder for deciphering. The decoder first decodes the TCs and then it searches for the encryptable coefficients using Algorithms 5.3 and 5.4. Finally, the deciphering will process only the identified encrypted bins.

5.4.2 Encryption Method and Synchronisation

The syntax elements to cipher are now defined. To cipher the syntax elements of a variable length, a stream cipher is more suited for this application. As the minimum error propagation is one of the most desirable properties in video encryption, we use the Advanced Encryption Standard (AES) algorithm in CounTeR (CTR) mode as a Pseudo-Random Number Generator (PRNG) to encrypt the identified syntax elements. It is important to note that CTR counter value should not be reused, which is adopted in our

solution [93]. Meanwhile, other stream ciphers such as Rabbit [94], LightWeight Chaos-Based (LWCB) stream cipher [95], HC-128 [96] or even block ciphers like AES in Cipher FeedBack (CFB) mode, can be used as well. A stream cipher produces a cipher text C using a eXclusive OR (XOR) operation between the plain text P and the output steam X_g produced by a PRNG,

$$C(P) = P \oplus X_g. \tag{5.7}$$

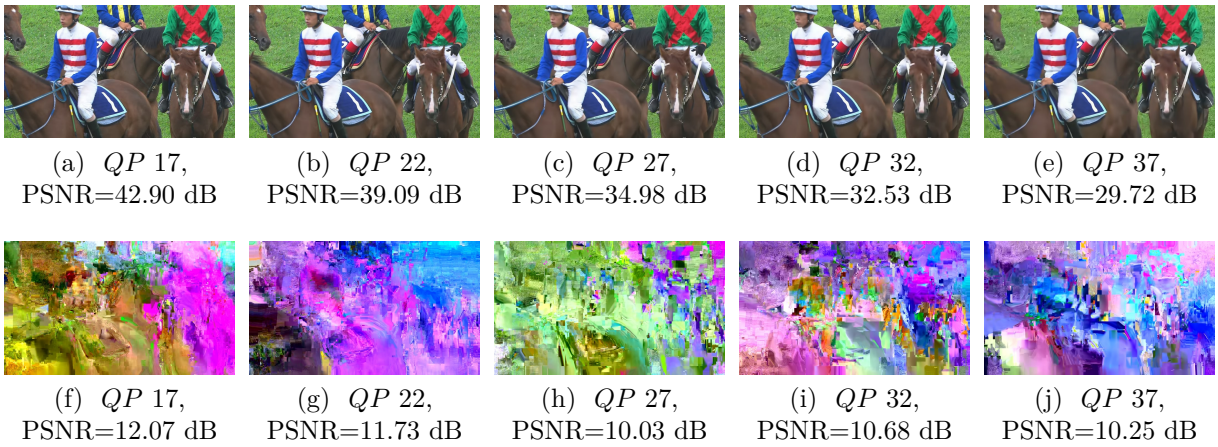
To revert the encryption, a XOR between the cipher text and the same PRNG output is performed. Thus, a perfect synchronization between the encoder and the decoder is required. Most of the syntax elements are systematically ciphered and do not dependant on the position or the context. However, the syntax elements associated to the TCs are ciphered only if they meet conditions previously described in Section 5.4.1. One of this conditions relies on the neighbor coefficients, implying that the last decoded coefficient needs to be deciphered first. To allow this behavior, for each significant coefficient, encryptable or not, the PRNG generates a sample equal to the size of the rice parameter, e.g. the maximum encryptable size. The unused samples are discarded to keep the encoder and the decoder perfectly synchronized. In CTR mode, one bit flipping caused by transmission errors will only affect one bit during the deciphering process which minimizes the error propagation.

5.5 Results and Discussions

In this section, we first present the experimental setup, followed by an assessment of the video degradation introduced by the selective encryption, then a security analysis will be presented, and finally a complexity evaluation is provided.

5.5.1 Experimental Setup

The experiments are carried-out under the Common Test Conditionss (CTCs) of the Versatile Video Coding (VVC) standard. The CTCs define several test video sequences of different resolutions, and five Quantization Parameters (QPs) are used $QP \in \{17, 22, 27, 32, 37\}$. The proposed encryption solution is implemented in the VVC Test Model (VTM) [97] version 6.0. VTM is the reference software implementation of both encoder and decoder of the VVC standard. The coding configuration without encryption is referred to as the Anchor. The video sequences are encoded with encryption in Random



(a) QP 17, PSNR=42.90 dB (b) QP 22, PSNR=39.09 dB (c) QP 27, PSNR=34.98 dB (d) QP 32, PSNR=32.53 dB (e) QP 37, PSNR=29.72 dB

(f) QP 17, PSNR=12.07 dB (g) QP 22, PSNR=11.73 dB (h) QP 27, PSNR=10.03 dB (i) QP 32, PSNR=10.68 dB (j) QP 37, PSNR=10.25 dB

Figure 5.6 – Visual illustration of Frame #10 of *RaceHorsesC* video decoded without encryption (5.6a - 5.6e) and with selective encryption (5.6f - 5.6j) at five QPs

Access (RA) coding configuration. This latter is the common coding configuration used in broadcast and Over-The-Top (OTT) applications with an Intra period of 32 frames. The complexity measurements are performed on a desktop computer equipped with an Intel i7-7700 processor running at 3.60 GHz on Ubuntu 18.04 OS.

5.5.2 Video Quality and Encryption Space

5.5.2.1 Video Quality

The distortion introduced by the proposed solution on the test video sequences is assessed in this section. Three full-reference objective image and video quality metrics are computed on the encrypted video sequences with respect to the original. PSNR is used to evaluate the video quality based on the mean squared error computed over the frame pixels [98]. The PSNR is computed as a weighted sum of the PSNR scores of the three color components. SSIM explores the structural similarity between the original and the decoded frame. It is important to note that a SSIM value close to 1 refers to decoded frame of a similar quality as the original frame [99]. Finally, VMAF is a video quality metric that predicts the perceived quality score of a video sequence [100], where a score of 100 indicates a good perceptual video quality and 0 refers to a very low perceived video quality.

Table 5.2 presents the PSNR performance over all video sequences at the five considered QPs. We can notice that the PSNR drops at QP 17 from 44.77 dB in average to

Table 5.2 – Peak Signal to Noise Ration (PSNR) performance of the proposed selective encryption for all video sequences at five QPs. Anchor and ciphered configurations correspond to the video decoded without encryption and with selective encryption, respectively.

		PSNR Scores (dB)									
		QP 17		*QP* 22		*QP* 27		*QP* 32		*QP* 37	
		Anchor	Ciphered	Anchor	Ciphered	Anchor	Ciphered	Anchor	Ciphered	Anchor	Ciphered
A1	Campfire	44.02	4.58	39.78	4.69	37.64	4.60	36.54	4.38	35.18	4.37
	FoodMarket4	46.21	10.99	44.29	10.79	42.96	10.62	41.13	9.76	38.84	10.83
	Tango2	42.37	8.66	40.40	9.05	39.73	10.40	38.88	8.16	37.58	8.68
A2	CatRobot1	42.84	9.13	40.48	9.55	39.56	9.73	38.37	8.72	36.70	9.08
	DaylightRoad2	41.76	8.93	38.25	11.16	37.33	9.34	36.44	9.74	35.12	10.00
	ParkRunning3	47.51	11.24	43.83	11.52	39.85	10.56	36.58	12.00	33.60	11.10
B	BasketballDrive	42.08	13.17	39.55	12.95	37.98	11.90	36.34	11.26	34.40	11.63
	BQTerrace	42.76	10.22	37.66	10.23	35.56	10.28	34.29	10.15	32.78	9.82
	Cactus	41.54	10.04	38.74	10.23	37.24	9.48	35.59	9.12	33.51	8.99
	MarketPlace	43.52	9.55	40.96	9.10	38.84	8.22	36.72	8.61	34.44	8.67
	RitualDance	47.02	9.43	44.79	10.44	41.71	10.11	38.71	9.28	35.76	9.56
C	BasketballDrill	44.19	13.40	41.69	12.83	38.57	12.88	35.71	11.94	33.09	11.22
	BQMall	42.70	11.94	40.73	11.12	38.34	10.79	35.81	11.16	33.14	10.65
	PartyScene	42.13	12.02	39.05	11.78	35.73	11.34	32.76	11.21	29.96	11.33
	RaceHorsesC	43.13	11.53	39.58	11.58	36.48	11.74	33.80	11.64	31.15	10.97
D	BasketballPass	45.22	13.54	41.58	13.44	37.54	13.15	34.27	13.41	31.39	13.79
	BlowingBubbles	41.75	11.20	38.81	11.69	35.60	11.06	32.69	11.44	29.87	11.57
	BQSquare	42.10	8.89	38.69	9.37	35.41	8.78	32.70	8.93	30.22	9.62
	RaceHorses	43.66	11.75	40.14	12.11	36.55	12.05	33.25	11.74	30.31	11.56
E	FourPeople	44.92	9.91	43.38	9.72	41.81	8.73	39.82	8.93	37.30	8.70
	Johnny	44.99	9.54	43.53	9.26	42.38	8.70	40.94	8.58	39.01	8.48
	KristenAndSara	45.42	9.50	43.86	9.15	42.40	7.60	40.64	7.73	38.42	7.62
F	ArenaOfValor	46.94	11.15	43.63	10.78	40.28	10.67	37.45	9.38	34.79	9.46
	BasketballDrillText	44.31	12.52	41.73	12.01	38.52	12.10	35.60	11.24	32.89	11.21
	SlideEditing	54.72	9.69	51.35	10.49	47.31	10.15	43.25	10.72	38.92	9.31
	SlideShow	56.12	1.94	52.43	3.16	48.61	4.28	45.17	3.98	41.83	4.35
Average		**44.77**	**10.17**	**41.88**	**10.32**	**39.38**	**9.97**	**37.05**	**9.74**	**34.62**	**9.71**

Table 5.3 – Average Structural SIMilarity (SSIM) performance of the proposed encryption solution for all video classes at five QPs

	SSIM Score									
	QP 17		*QP* 22		*QP* 27		*QP* 32		*QP* 37	
	Anchor	Ciphered	Anchor	Ciphered	Anchor	Ciphered	Anchor	Ciphered	Anchor	Ciphered
A1	1.00	0.27	1.00	0.22	1.00	0.23	0.99	0.21	0.99	0.22
A2	1.00	0.23	1.00	0.23	1.00	0.21	0.99	0.21	0.99	0.21
B	1.00	0.27	1.00	0.27	0.99	0.26	0.99	0.25	0.97	0.25
C	1.00	0.23	0.99	0.23	0.98	0.23	0.97	0.23	0.94	0.24
D	0.99	0.24	0.97	0.24	0.95	0.25	0.91	0.26	0.85	0.28
E	1.00	0.39	1.00	0.40	0.99	0.37	0.99	0.37	0.99	0.39
F	1.00	0.26	1.00	0.31	1.00	0.35	0.99	0.36	0.97	0.35
Average	**1.00**	**0.27**	**0.99**	**0.27**	**0.99**	**0.27**	**0.97**	**0.27**	**0.95**	**0.28**

Table 5.4 – Average Video Multimethod Assessment Fusion (VMAF) performance of the proposed encryption solution for all video classes at five QPs

	VMAF Score									
	QP 17		*QP* 22		*QP* 27		*QP* 32		*QP* 37	
	Anchor	Ciphered	Anchor	Ciphered	Anchor	Ciphered	Anchor	Ciphered	Anchor	Ciphered
A1	99.47	25.53	99.03	28.84	96.70	30.23	92.31	29.84	84.86	28.19
A2	99.88	23.79	99.22	23.88	97.20	25.15	92.49	25.27	85.02	25.22
B	99.80	2.78	99.08	2.89	96.35	3.24	89.46	3.52	77.84	3.67
C	99.88	7.16	99.59	7.72	96.91	7.78	89.84	7.63	77.55	7.88
D	99.42	6.66	98.78	6.57	95.68	6.76	87.96	6.75	75.68	6.29
E	97.37	0.16	96.54	0.51	95.00	0.62	92.04	0.71	86.52	1.27
F	98.88	6.20	98.65	7.83	96.72	8.57	92.47	8.66	85.43	6.86
Average	**99.30**	**9.32**	**98.76**	**10.10**	**96.37**	**10.64**	**90.73**	**10.66**	**81.27**	**10.25**

around 10.17 dB. The same PSNR values of encrypted videos are reached on different QPs values. This indicates that the proposed solution significantly decreases the objective quality of the encrypted video. *Campfire* and *SlideShow* encrypted video sequences have a very low PSNR values. For *Campfire*, it can be explained by its texture and complex shapes associated to high motion that increase the encryption space and thus improving the quality of the encryption. Concerning *SlideShow*, shapes are less complex however the encryption is able to flip the colors causing noticeable quality degradation with lower PSNR scores than the average.

Table 5.3 presents the average SSIM scores for seven video classes at different QPs. The proposed encryption solution enables to reduce the SSIM from around 1 to 0.25. The

obtained SSIM value confirms that the proposed solution introduces a drastic distortion on the structure information within the encrypted video frame. We can notice that SSIM scores of class E video sequences are higher than the average scores. These video sequences have low motion and less texture compared to other sequences. This improves the coding efficiency and decreases the performance of the selective encryption since less syntax elements are encrypted.

Finally, Table 5.4 presents the VMAF scores which also emphasize the large degradation of the subjective video quality as a result of using the proposed encryption solution.

Fig. 5.6 illustrates the frame #10 of *RaceHorcesC* video sequence decoded at five QP values with and without encryption. The visual quality of decoded encrypted video is very low making difficult to recognize objects and colors in the video frame at all QP values with PSNR scores around 11 dB.

5.5.2.2 Encryption space

The computational time of encryption mainly depends on the encryption space of any ciphering process. However, the robustness and security level will be enhanced by increasing the encryption space. In selective encryption, the somehow robust encryption algorithm and low computational overhead as outcome of the encryption is the target. Table 5.5 presents the encryption space of the proposed encryption solution as the percentage of encrypted bits by syntax element on the whole bitstream. The quality degradation of selective encryption is achieved by ciphering only 26.66% and 15.42% of the bitstream at high and low bitrates, respectively. We can notice that the largest encryption space is enabled by the encryption of the Transform Coefficients (TCs) while the part of other syntax element less present in the bitstream remains negligible ($< 2\%$).

5.5.3 Security Analysis

In the previous section we only assess the visual degradation achieved by the proposed encryption. In this section we focus on the quality of the proposed encryption and its robustness against different types of attacks.

5.5.3.1 Encryption Qualilty (EQ) analysis

The algebraic summation of differences between pixels distributions of the original frame $H(P)$ and the encrypted fame $H(C)$ is called EQ. This latter is computed as

Table 5.5 – Encryption space in percentage (%) per syntax element at five QPs

Syntax Elements	Encryption Space (%)				
	QP 17	QP 22	QP 27	QP 32	QP 37
alf_luma_filter_idx	0.01	0.02	0.04	0.06	0.06
sao_offset_sign	0.00	0.00	0.00	0.00	0.00
sao_band_position	0.01	0.01	0.01	0.01	0.01
sao_eo_class	0.01	0.02	0.01	0.01	0.00
mmvd_direction_idx	0.48	0.55	0.56	0.59	0.59
merge_triangle_split_dir	0.04	0.08	0.13	0.17	0.19
mvd_abs	0.26	0.48	0.76	0.98	1.14
mvd_sign	0.29	0.51	0.75	0.92	1.00
abs_remainder coeff_sign	25.44	21.39	17.15	14.33	12.22
abs_remainderTS coeff_signTS	0.06	0.06	0.05	0.03	0.01
intra_chroma _pred_candidate	0.07	0.10	0.14	0.16	0.18
Total	**26.66**	**23.21**	**19.61**	**17.24**	**15.42**

follows [84]

$$EQ = \frac{\sum_{Z=0}^{2^d-1} |H_Z(C) - H_Z(P)|}{2^d}. \tag{5.8}$$

Table 5.6 – Encryption Quality for CTCs video classes at five QP values

	Encryption Quality (EQ)					
	QP 17	QP 22	QP 27	QP 32	QP 37	EQ_{max}
A1	8 661	8 260	8 573	7 947	8 535	16 200
A2	9 235	6 791	8 058	8 871	7 971	16 200
B	1 620	1 805	1 923	1 920	1 911	4 050
C	231	245	224	236	270	780
D	61	69	81	70	83	195
E	682	748	1 057	1 150	1 096	1 800
F	907	1 225	944	1 240	1 298	2 108
Average	**2640**	**2407**	**2603**	**2680**	**2652**	**5199**

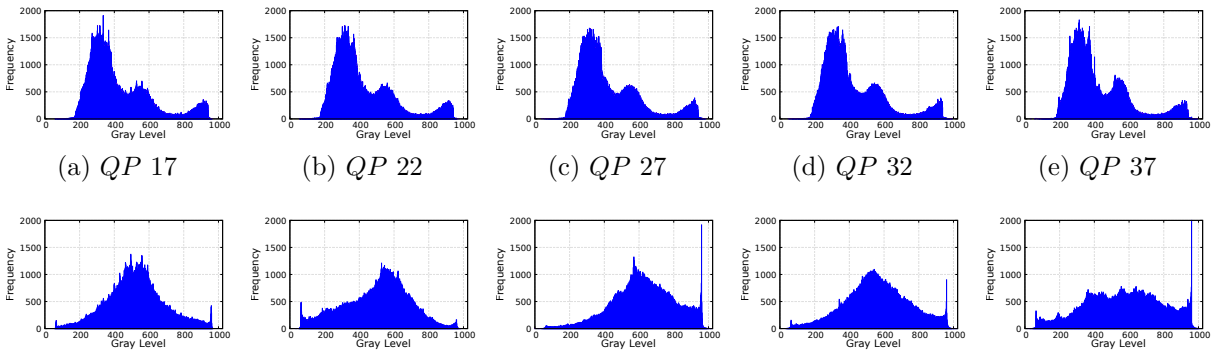
The higher EQ value is, the more secure is the selective encryption solution. Table 5.6 presents the EQ values for all video classes at the five considered QPs. The presented values are the average EQ over encrypted frames and video sequences of each class. The EQ does not have a relative point for comparison. A derivation from (5.8) is proposed to compute the upper bound value of the EQ [44] as follows

$$EQ_{max} = \frac{2\,W\,H}{2^d}, \tag{5.9}$$

where H and W are the video height and width, respectively and d is the bit depth. The upper bound value of the EQ is reached when the histograms of the two frames $H_Z(C)$ and $H_Z(P)$ are not overlapping.

The average EQ values are within the interval [2407, 2680] in average with a theoretical average upper bound of 5199.

The High Efficiency Video Coding (HEVC) selective encryption solution proposed in [44] achieved an EQ value for *Kimono* video sequence higher than 38.54% of its maximum EQ, and an EQ value for *PeopleOnStreet* video higher than 40.92% its maximum EQ. The proposed solution of different videos at different configuration ranges between 46.29% and 51.56% of the maximum EQ which confirms that the proposed solution has a high security level regarding the encryption quality metric.



(a) QP 17 (b) QP 22 (c) QP 27 (d) QP 32 (e) QP 37

(f) QP 17 (g) QP 22 (h) QP 27 (i) QP 32 (j) QP 37

Figure 5.7 – Histograms of frame #10 computed for the anchor (5.7a- 5.7e) and cipher (5.7f- 5.7j) of *RaceHorsesC* video at five QP values. The selective encryption solution significantly changes the pixels distributions as shown by the EQ metric.

5.5.3.2 Histogram analysis

Histogram of encrypted frame should be more uniform than original frame histogram in order to resist to statistical analysis based attacks [101, 102]. Fig. 5.7 illustrates the histograms of frame #10 of *RaceHorcesC* video sequence before and after selective encryption at five considered QPs. The histograms of the encrypted frames is completely different from the original frame histogram. In fact, the proposed encryption solution changes the distribution of the decoded pixels toward different pattern which is close to uniform distribution especially at lower bitrate (ie. high QP). We can also notice that in contrast to full encryption, it is difficult for constant bitrate and format compliant selective encryption

to reach the uniform distribution of the histogram at all coding configurations and video contents.

5.5.3.3 Edges and structural information protection

Edge detection enables assessing the ability of an encryption solution to hide the edge information in the encrypted frame. This section evaluates the ability of the proposed encryption solution to hide the edge in the encrypted video sequence. The Edge Differential Ratio (EDR) is computed by (5.10) [85, 103].

$$EDR = \frac{\sum_{i=1}^{H}\sum_{j=1}^{W}|P_{ED}(i,j) - C_{ED}(i,j)|}{\sum_{i=1}^{H}\sum_{j=1}^{W}|P_{ED}(i,j) + C_{ED}(i,j)|}, \tag{5.10}$$

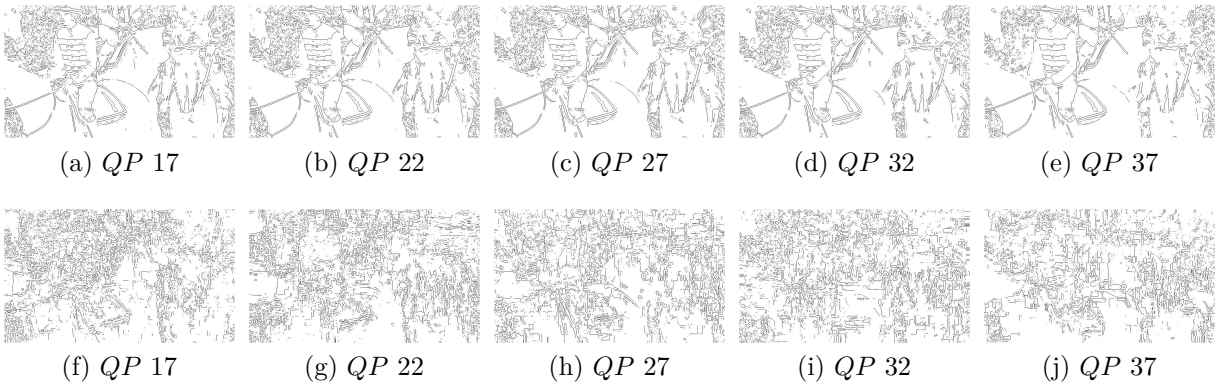
with P_{ED} and C_{ED} are the binary Laplacian of the decoded images with and without encryption, respectively. EDR takes values in the interval [0, 1], where a value close to 1 corresponds to a high edge hiding capability.

Table 5.7 presents the average values of EDR computed on the first 64 frames of the CTCs video sequences. The average EDR values are higher than 0.87 which shows the ability of the proposed selective encryption to hide edges and structural information in the encrypted frames. We can notice a lower EDR performance for class F video sequences. This class includes mainly screen content video sequences for which selective encryption is less effective to hide the structure of the edges.

Table 5.7 – Average EDR for CTCs video classes at five QP values

	EDR				
	QP 17	*QP* 22	*QP* 27	*QP* 32	*QP* 37
A1	0.919	0.923	0.928	0.932	0.934
A2	0.890	0.896	0.900	0.904	0.906
B	0.890	0.893	0.896	0.899	0.904
C	0.879	0.877	0.877	0.876	0.874
D	0.876	0.887	0.890	0.880	0.881
E	0.866	0.864	0.856	0.856	0.844
F	0.813	0.780	0.762	0.741	0.762
Average	**0.875**	**0.873**	**0.871**	**0.867**	**0.871**

Fig. 5.8 illustrates the edges of the decoded frame #10 of *RaceHorsesC* sequence without encryption (first row) and with encryption at the second row for five different QPs. This figure clearly shows that the ciphered frames are noisy caused by high frequency



(a) QP 17 (b) QP 22 (c) QP 27 (d) QP 32 (e) QP 37

(f) QP 17 (g) QP 22 (h) QP 27 (i) QP 32 (j) QP 37

Figure 5.8 – Edge Detection on Frame #10 computed for the anchor (5.8a-5.8e) and ciphered (5.8f- 5.8j) *RaceHorsesC* video at five QPs.

structure introduced by the selective encryption. Therefore, structural information including edges in the ciphered frames are hidden and can be hardly explored by the EDR based attacks.

5.5.3.4 Sensitivity to secret keys

Key sensitivity attacks are mainly based on the fact that the adversary tries to decipher the encrypted frames using a key close to the secret key used for encryption. The second adversary scenario is to guess the key if the encryption system provides information related to the used secrete key such as the sensitivity of the encryption regarding small change in the key. The proposed encryption algorithm should produce a completely different encrypted frame when a slight change (one bit change) on the used secrete key [104]. Evaluation of the system robustness against key sensitivity attacks can be assessed using many existing tools such as Unified Average Changing Intensity (UACI) and Number of Pixels Change Rate (NPCR) [52, 53]. To compute these metrics, one random key is generated K_1 and a key with only one bit difference K_2 is created. The two keys are then used to cipher the same frame of width W and height H and a bit depth d. The result will create a ciphered frame C_1 using K_1, and C_2 using K_2. The UACI and the NPCR are defined as follow

$$UACI = \frac{1}{H\,W\,2^d} \sum_{i=1}^{H} \sum_{j=1}^{W} |C_1(i,j) - C_2(i,j)|\ 100\%, \tag{5.11}$$

$$NPCR = \frac{1}{H\,W} \sum_{i=1}^{H} \sum_{j=1}^{W} D(i,j)\,100\%, \tag{5.12}$$

with:

$$D(i,j) = \begin{cases} 0, & \text{if } C_1(i,j) = C_2(i,j) \\ 1, & \text{if } C_1(i,j) \neq C_2(i,j) \end{cases}.$$

Table 5.8 – NPCR and UACI with two secret keys with 1-bit-difference

	UACI and *NPCR*									
	QP 17		*QP* 22		*QP* 27		*QP* 32		*QP* 37	
	UACI	*NPCR*	*UACI*	*NPCR*	*UACI*	*NPCR*	*UACI*	*NPCR*	*UACI*	*NPCR*
A1	25.52	99.62	29.76	99.63	24.68	99.77	33.00	99.83	31.18	99.90
A2	22.03	99.85	20.70	99.80	21.30	99.82	25.24	99.85	30.29	99.76
B	24.48	99.83	26.17	99.82	24.27	99.82	29.05	99.81	27.04	99.60
C	19.60	99.82	23.71	99.86	24.32	99.86	21.03	99.83	23.21	99.84
D	22.18	99.85	22.68	99.84	23.12	99.83	21.34	99.76	23.51	99.69
E	23.69	99.79	23.94	99.70	32.54	99.91	43.99	99.95	37.79	99.91
F	26.84	99.52	37.42	99.03	26.96	99.33	33.27	99.66	22.65	98.76
Average	**23.48**	**99.76**	**26.51**	**99.67**	**25.17**	**99.76**	**29.02**	**99.80**	**27.32**	**99.61**

The optimal NPCR and UACI values of a secure image encryption scheme against key sensitivity attacks are 99.58% and 33.46%, respectively [54].

Table 5.8 presents the obtained NPCR and UACI values on the CTCs for all video classes at three QPs. Here, it is important to note that the NPCR and UACI results of the selective encryption should not be analysed as in full image encryption. However, the obtained values can give an indication on the ability of the selective encryption to resist key sensitive and differential attacks. The average NPCR values at the three QPs for all classes are very close to the optimal value of a secure encryption scheme against key sensitivity attacks. Moreover, the average UACI values lie in the interval [19.60, 37.79] with an average value over all classes that converges to the optimal value of 33.64. This performance in terms of both NPCR and UACI proves the robustness of the proposed selective encryption solution with regards the key sensitivity and differential attacks.

5.5.3.5 Brute Force Attack

Brute force attack or exhaustive search attack performs testing all possible values of the used secret key in order to partially or completely break the cipher. [105] It is well-known, that any encryption algorithm with at least 128 bits as secret key is considered

Table 5.9 – Replacement Attack, average PSNR, SSIM and VMAF score on CTCs.

QP	PSNR		SSIM		VMAF	
	Anchor	Replaced	Anchor	Replaced	Anchor	Repla.
17	44.85	5.87	1.00	0.29	99.21	6.77
22	41.98	5.80	0.99	0.30	98.47	6.67
27	39.42	5.90	0.99	0.31	95.78	7.16
32	37.02	5.80	0.97	0.32	89.88	7.29
37	34.54	5.92	0.95	0.32	80.21	6.93

as resilient to brute force attack, which is the case for the used Advanced Encryption Standard (AES) algorithm. In selective encryption, the total number of tries to correctly guess the selected encrypted bits should be at least 2^{128} tries in order to resist to brute force attack [106, 107]. Our proposed selective encryption algorithm relies on AES in counter mode as stream cipher with a secret key size of 128 bits. Moreover, the size of the encryption space is very large.

5.5.3.6 Error Concealment Attack

The error concealment attack is a kind of attack based on guessing the encrypted bits based on some assumptions. However, since the encryption space of a VVC video is large, the only scenario that the adversary can follow is to try replacing all encrypted bits with the same value (zero or one) and decipher the modified encrypted frame [108, 109]. In order to evaluate our proposed solution regarding error concealment attacks, all encrypted bits are replaced by zero and then PSNR, SSIM and VMAF are calculated again under the same CTCs. Table 5.9 gives the average PSNR, SSIM and VMAF scores of video sequences deciphered with replacement attack at the five QPs. The obtained quality scores are similar or even worst compared to encrypted video with AES generator presented in Section 5.5.2. This confirms that the proposed selective encryption solution is robust against attacks based on replacement bits.

5.5.4 Complexity Analysis

The aim of this section is to assess the complexity of the proposed encryption solution. The complexity overhead is computed only for the decoder, since the encryption overhead is negligible with respect to the encoding time. The average decoding run time is computed based on 100 decodings without deciphering ($DecT_{Ref}$) and with deciphering ($DecT_{SE}$). The deciphering run time Δ_{SE} is computed as a difference between decoding times with

and without deciphering $\Delta_{SE} = DecT_{SE} - DecT_{Ref}$, while the percentage of deciphering complexity overhead CO_{SE} is derived as follows

$$CO_{SE} = \frac{\Delta_{SE}}{DecT_{Ref}} 100\%. \tag{5.13}$$

Table 5.10 gives the deciphering time Δ_{SE} in second and the deciphering complexity overhead in percentage for all video classes at three QPs. The deciphering time does not exceed 3 seconds even for high bitrate and high resolution 4K videos of classes A and B. This corresponds to less than 6% of the total decoding time. The average deciphering overhead remains lower than 4.23% observed at high bitrate presenting more TCs to cipher.

Table 5.10 – Deciphering time Δ_{SE} in second and deciphering overhead CO_{SE} in % on Intel i7-7700 processor at 3.6 GHz.

	QP 17		*QP* 22		*QP* 27		*QP* 32		*QP* 37	
	Δ_{SE}	CO_{SE}	Δ_{SE}	CO_{SE}	Δ_{SE}	CO_{SE}	Δ_{SE}	CO_{SE}	Δ_{SE}	CO_{SE}
A1	1.356	3.540	0.404	1.560	0.091	0.518	0.005	0.019	-0.013	-0.147
A2	2.514	5.605	0.625	2.448	0.156	0.674	0.025	0.118	-0.001	-0.047
B	0.473	4.579	0.160	2.641	0.039	0.976	0.016	0.489	0.001	0.041
C	0.116	5.136	0.054	3.244	0.025	2.101	0.008	0.924	0.006	0.791
D	0.026	4.476	0.013	2.846	0.005	1.384	0.002	0.656	0.001	0.450
E	0.072	3.205	0.016	1.110	0.004	0.387	0.002	0.193	-0.002	-0.198
F	0.086	2.933	0.038	1.755	0.023	1.331	0.010	0.716	0.008	0.701
Average	**0.581**	**4.236**	**0.167**	**2.305**	**0.045**	**1.111**	**0.010**	**0.485**	**0.001**	**0.261**

5.6 Conclusion

In this chapter, a new selective encryption solution for the Versatile Video Coding (VVC) standard was proposed. This solution encrypts at the Context-Adaptive Binary Arithmetic Coding (CABAC) level a set of VVC syntax elements in format-compliant and constant bitrate. The coding of the Transform Coefficients (TCs) in VVC introduces several dependencies making constant bitrate encryption more challenging. We have proposed an original algorithm that analyses the coding dependencies of the TCs to determine the number and positions of encryptible bins for each coefficient. The proposed encryption solution was integrated in both encoder and decoder of the VVC reference software VVC Test Model (VTM) 6.0. The quality of the encrypted video was assessed under the VVC Common Test Conditionss (CTCs) with three objective quality metrics including Peak

Signal to Noise Ration (PSNR), Structural SIMilarity (SSIM) and Video Multimethod Assessment Fusion (VMAF). The low obtained quality scores clearly show the quality degradation enabled by the encryption. Security analysis was also conducted to asses the robustness against several attacks including statistical, key sensitivity and brute force attacks. Finally, the complexity overhead of the deciphering at the decoder side is estimated and remains lower than 6% of the decoding time confirming the lightweight advantage of the proposed encryption solution.

CHAPTER 6

Conclusion

6.1 Research Contributions

This thesis presented three different contributions. The first contribution presents a new stream cipher based on the work of Taha [1]. The second proposed several implementation of the proposed cipher and outlined some physical vulnerabilities. The last contribution is a selective encryption scheme for the new video coding standard Versatile Video Coding (VVC).

6.1.1 A new LightWeight Chaos-Based (LWCB) Stream Cipher

Chapter 3 introduces a method of attack using Side-Channel Attacks (SCAs) for the stream cipher proposed by Taha [1]. The proposed method enables an attacker to recover the secret key completely.

From the vulnerabilities exposed, a new stream cipher based on the work of Taha [1] is proposed. This new stream cipher introduces a new chaotic map, the 4D map, to replace the Skew Tent (ST) map to add non-linearity to the system. It also introduces a soft coupling of the map cells increasing the interdependence of chaotic maps.

The new LightWeight Chaos-Based (LWCB) stream cipher is tested against statistical attacks using several statistical tests like the National Institute of Standards and Technol-

ogy (NIST) Statistical Tests Suite (STS), correlation, histogram analysis, key sensitivity and confusion analysis. Proposed cipher is passing all the statistical tests.

6.1.2 Hardware Design of Proposed Stream Cipher

In Chapter 4, several hardware designs of the stream cipher proposed in Chapter 3 are presented. The reference design, corresponding to an optimized implementation of the stream cipher without pipeline or additive masking. This implementation, using only 441 slices, achieves a throughput of 711.11 Mbps at an operating frequency of 22.22 MHz. The proposed 4-stage pipeline implementation operates at 80 MHz with a throughput of 2 560 Mbps and is using only 611 slices. This implementation approaches a speed up of 3.6, and is increasing the area by less than 40%. The last implementation is adding additive masking to the reference design. This addition slightly affects the speed performances, the throughput is only dropping to 673.98 Mbps.

Then, in a short study, the security against SCAs and the effect of the additive masking is assessed using the methodology proposed by Schneider and Moradi [60]. This study shows that the proposed implementations are not passing first order t-test and might be leaking information about the secret key.

6.1.3 Selective Encryption on the new Video Compression Standard Versatile Video Coding (VVC)

Chapter 5 presents the proposed selective encryption solution of VVC standard. This encryption is completely format compliant and Constant Bit Rate (CBR). It takes place at the Context-Adaptive Binary Arithmetic Coding (CABAC) level. This last chapter introduces a method mainly focusing on encrypting the Transform Coefficients (TCs). The TCs are challenging to encrypt due to its high dependency between each other. The proposed encryption scheme was integrated in both encoder and decoder of the VVC reference software VVC Test Model (VTM) 6.0.

The quality of the proposed scheme is evaluated under the Joint Video Experts Team (JVET) Common Test Conditionss (CTCs) with Peak Signal to Noise Ration (PSNR), Structural SIMilarity (SSIM) and Video Multimethod Assessment Fusion (VMAF) metrics. The degradation introduced by the encryption is notable for all metrics. The security of the proposed scheme is also evaluated through Encryption Qualilty (EQ), histogram

analysis, edge detection or sensitivity to the secret key. Finally, the encryption only shows a complexity overhead lower than 6% for the decoding process.

6.2 Prospects – Future Works

6.2.1 Countermeasures Against SCAs for Chaotic Maps

In proposed hardware implementations from Chapter 4, only the coupling matrix and the recursive cells are masked. From the results of the same chapter, the chaotic maps are leaking sensitive information. Chaotic maps used in encryption schemes are usually non-linear, or at least partly linear. To our knowledge, no work on this matter was published in the literature.

6.2.2 Gate-Level Implementation of Proposed LWCB Stream Cipher

From the conclusion made in Chapter 4 about the non-selective t-test, the proposed implementation should be improved. The possible leaks exposed need to be mitigated. Working to a lower-level implementation, at the gate level, could be a solution to suppress leakage. In the literature, proposed countermeasures are usually based on hiding sensitive data by adding extra gates to perform the opposite function at the bit level in order to have constant energy consumption.

6.2.3 Watermarking for VVC Standard

The solution proposed in Chapter 5 can have many applications such as Digital Rights Management (DRM) or watermarking. Watermarking is a technique used to identify a noise-tolerant signal by injecting markers into bits without a no visible onto subjective quality. By definition, watermarking is sort of the opposite of selective encryption. Same techniques can be implemented to VVC encoders.

APPENDIX A

French Summary

A.1 Contexte

Le besoin d'une communication sécurisée entre plusieurs entités a pratiquement toujours existé. Au début, la cryptographie n'était utilisée que par l'armée et les gouvernements. Aujourd'hui, ce besoin est devenu encore plus important avec l'apparition de moyens de communication plus modernes et le passage à un environnement entièrement numérique. Des canaux de communication sécurisés sont nécessaires, même pour des applications civiles telles que les applications bancaires, le télétravail ou encore la vidéo à la demande (VOD). Cette diversification d'utilisation, associée à la multiplication des appareils, des architectures ainsi que les ressources limitées des systèmes embarqués, ajoute de plus en plus de contraintes à la conception des algorithmes de chiffrement. En effet, le chiffrement et ses implémentations doivent être plus robustes à mesure que la capacité de calcul devient moins chère. Ils doivent également s'adapter à la pluralité des systèmes et à sa variété d'architectures, en tenant compte des ressources limitées telles que l'énergie, la mémoire ou les limitations de puissance de calcul des appareils.

En outre, au cours de la dernière décennie, l'utilisation des applications vidéo sur le réseau a augmenté de manière exponentielle. Les services VOD comme Netflix ou Amazon Prime par exemple, les réseaux de télévision payants doivent protéger leur contenu pour qu'ils ne puissent être regardés que par leurs abonnés. Toutefois, cette protection ne doit

pas affecter l'expérience de l'utilisateur en restant à faible délai et peu complexe pour continuer à assurer un décodage en temps réel.

Dans ce contexte, cette thèse vise à créer un lien entre ces deux différents domaines de recherche, tout d'abord en étudiant la mise en œuvre d'un schéma de cryptage sécurisé, puis en étudiant comment protéger une application vidéo sans créer de gêne pour l'utilisateur final.

A.2 Objectifs et contributions de la thèse

Les principaux objectifs de cette thèse sont les suivants :

1. Étudier la résilience contre les attaques par canaux auxiliairesSCAs d'un chiffrement par flux basé chaos existant.
2. Améliorer sa résilience en modifiant sa conception et en mettant en œuvre des contre-mesures comme le masquage.
3. Proposer une conception matérielle du chiffrement par flux amélioré.
4. Proposer une méthode de cryptage sélectif pour le contenu vidéo sans affecter le taux de compression.

Ces quatre objectifs sont regroupés dans trois contributions. La première contribution se concentre sur l'amélioration du chiffrement par flux basé chaos proposé par Taha [1]. Une méthode d'attaque, utilisant 25 SCAs consécutives, est proposée pour récupérer complètement la clé secrète. À partir de la méthode proposée, une nouvelle méthode de chiffrement par flux est présentée. Cette nouvelle conception vise à contrer, ou du moins à augmenter la complexité des attaques effectuées sur la conception précédente.

La deuxième contribution consiste à proposer une conception matérielle légère, sûre et efficace du nouveau système de chiffrement par flux. Plusieurs implémentations matérielles sont proposées, la première est une implémentation optimisée du nouveau chiffrement par flux sans masquage ni pipeline. Cette implémentation est la conception de référence. Ensuite, la seconde implémentation tire parti du pipeline pour augmenter le débit. Les troisième et quatrième implémentations intègrent une opération de masquage sur une partie de la conception de référence comme contre-mesure à SCAs.

La troisième contribution est un schéma de cryptage sélectif dans la toute nouvelle norme de compression vidéo VVCs. Ce schéma de cryptage sélectif est développé en respectant deux contraintes majeures : la solution doit être à débit constant, l'efficacité

de la compression doit rester inchangée, et conforme à la norme, la vidéo cryptée doit être lisible en utilisant n'importe quel décodeur VVCs standard.

A.3 Organisation du manuscrit

Cette thèse est organisée comme suit. La partie I est consacrée au contexte et à un bref examen de l'état de l'art concernant trois sujets principaux :

1. la cryptographie dans la Section 2.1, avec une présentation des propriétés de **confidentialité**, **d'intégrité des données** et **d'authentification**. La différence entre un système de cryptage **asymétrique** et **symétrique**, et leurs applications. Enfin, une présentation de quelques systèmes de chiffrement symétriques est proposée;
2. la théorie du chaos dans la Section 2.2, la définition d'une carte chaotique et leurs applications à la cryptographie;
3. enfin, dans la Section 2.3, un aperçu sur les SCAs et ses contre-mesures possibles.

La Partie II se concentre sur les contributions de cette thèse avec :

1. en premier lieu, après une évaluation de la sécurité du système de chiffrement par flux proposée par Taha [1], une nouvelle version de ce système est proposée dans le Chapitre 3;
2. le Chapitre 4 présente les implémentations matérielles du chiffrement par flux proposé avec des résultats en termes de vitesse et d'utilisation des ressources matérielles. Ce chapitre comprend également une courte étude sur la résistance des implémentations face aux SCAs;
3. Enfin, chapitre 5, une méthode de cryptage sélectif sur la toute dernière norme de codage vidéo VVC est présentée.

Cette thèse est ensuite conclue dans le Chapitre 6 suivi de quelques perspectives et travaux futurs.

A.4 Conclusion

Cette thèse a présenté trois contributions différentes. La première contribution présente un nouvel algorithme de chiffrement basé sur les travaux de Taha [1]. La seconde propose plusieurs implémentations du chiffrement proposé et souligne certaines vulnérabilités

physiques. La dernière contribution est un schéma de cryptage sélectif pour la nouvelle norme de codage vidéo VVC.

A.4.1 Un nouveau chiffrement par flux basé chaos

Le chapitre 3 présente une méthode d'attaque utilisant des SCAs contre le chiffrement par flux proposé par Taha [1]. La méthode proposée permet à un attaquant de récupérer complètement la clé secrète.

À partir des vulnérabilités exposées, un nouveau chiffrement par flux basé sur les travaux de Taha [1] est proposé. Ce nouveau chiffrement introduit une nouvelle carte chaotique, la carte 4D, pour remplacer la carte ST et ajouter de la non-linéarité au système. Il introduit également un couplage souple des cellules de la carte, augmentant l'interdépendance des cartes chaotiques.

Le nouveau chiffrement par flux basé chaos est testé contre les attaques statistiques à l'aide de plusieurs tests statistiques comme la suite de test NIST STS, test de corrélation, l'analyse de l'histogramme, l'analyse de la sensibilité des clés et de la confusion par exemple. Le chiffrement proposé passe tous les tests statistiques.

A.4.2 Conception matérielle du chiffrement proposé

Le chapitre 4 présente plusieurs conceptions matérielles du chiffrement par flux proposé dans le chapitre 3. La conception de référence, qui correspond à une implémentation optimisée du chiffrement par flux sans pipeline ni masquage additif. Cette implémentation, qui n'utilise que 441 tranches, permet d'obtenir un débit de 711,11 Mo/s avec une fréquence de fonctionnement de 22,22 MHz. L'implémentation pipeline à 4 étapes, proposée dans ce manuscrit, fonctionne à 80 MHz avec un débit de 2 560 Mo/s, et n'utilise que 611 tranches. Cette mise en œuvre approche une vitesse de 3,6, et augmente la surface de moins de 40 %. La dernière implémentation consiste à ajouter un masquage additif à la conception de référence. Cet ajout affecte légèrement les performances en ce qui concerne la vitesse, le débit ne diminuant qu'à 673,98 Mo/s.

Ensuite, dans une brève étude, la sécurité contre les SCAs et l'effet du masquage additif sont évalués en utilisant la méthodologie proposée par Schneider et Moradi [60]. Cette étude montre que les implémentations proposées ne passent pas le t-test de premier ordre et pourraient entraîner des fuites d'informations sur la clé secrète.

A.4.3 Cryptage sélectif sur la nouvelle norme de compression vidéo Versatile Video Coding (VVC)

Le chapitre 5 présente la solution de cryptage sélectif proposée de la norme VVC. Ce cryptage est entièrement conforme au format et à débit constant. Il s'effectue au niveau CABAC. Ce dernier chapitre présente une méthode principalement axée sur le cryptage des TCs. Les TCs sont difficiles à chiffrer en raison de leur grande dépendance les uns par rapport aux autres. Le schéma de cryptage proposé a été intégré dans le codeur et le décodeur du logiciel de référence VVC VTM 6.0.

La qualité du schéma proposé est évaluée sur des séquences vidéo suivant les Conditions de Test Communes CTCs avec les métriques PSNR, SSIM et VMAF. La dégradation introduite par le cryptage est notable pour sur toutes les métriques. De plus, la sécurité du système proposé est également évaluée au moyen d'une analyse de l'histogramme, d'une détection des contours ou de la sensibilité à la clé secrète. Enfin, le cryptage ne présente qu'une faible augmentation de complexité qui reste inférieure à 6 % pour le processus de décodage.

A.5 Perspectives et Travaux futurs

A.5.1 Contre-mesures contres les SCAs pour les cartes chaotiques

Dans les implémentations matérielles proposées au chapitre 4, seules la matrice de couplage et les cellules récursives sont masquées. D'après les résultats du même chapitre, les cartes chaotiques sont des informations sensibles aux fuites. Les cartes chaotiques utilisées dans les schémas de cryptage sont généralement non linéaires, ou du moins partiellement linéaires. À notre connaissance, aucun travail sur ce sujet n'a été publié.

A.5.2 Implémentation du chiffrement par flux proposé niveau porte logique

D'après la conclusion formulée au chapitre 4 sur le t-test non sélectif, l'implémentation proposée doit être améliorée. Les éventuelles fuites exposées doivent être atténuées. Travailler sur une implémentation à un niveau inférieur, au niveau des portes logiques, pourrait être une solution pour supprimer les fuites. Dans la littérature, les contre-mesures

proposées sont généralement basées sur la dissimulation de données sensibles en ajoutant des portes supplémentaires pour effectuer la fonction inverse au niveau du bit afin d'avoir une consommation d'énergie constante, quelle que soit la valeur de la clé.

A.5.3 Méthode d'insertion de filigrane pour le standard VVC

La solution proposée au chapitre 5 peut avoir de nombreuses applications telles que la gestion des droits numériques (DRM), ou l'insertion de filigrane. L'insertion de filigrane est une technique utilisée pour identifier un signal tolérant au bruit en injectant un marqueur dans les bits sans effet visible sur la qualité subjective. Par définition, l'insertion d'un filigrane est en quelque sorte l'opposé du cryptage sélectif. Cependant, les mêmes techniques peuvent être appliquées à l'encodeur VVC.

List of Figures

List of Tables

List of Algorithms

Glossary

AC Alternating Current p. 83

AES Advanced Encryption Standard pp. 12, 21, 22, 54, 60, 65, 66, 73–75, 81, 83, 97, 107, 108, 148

AVC Advanced Video Coding p. 81

B frame Bidirectional predicted frame pp. 83, 90

BRAM Block RAM p. 72

CABAC Context-Adaptive Binary Arithmetic Coding pp. 82, 83, 86–88, 90, 92, 96, 109, 112, 118, 123

CBC Cipher Block Chaining pp. 3, 19, 20, 121

CBR Constant Bit Rate pp. 12, 112

CFB Cipher FeedBack pp. 3, 19–21, 83, 97, 121

CPA Correlation Power Analysis pp. 4, 30, 38, 42, 43, 65, 70

CPU Central Processing Unit p. 59

CT Constant-Time pp. 47, 121

CTC Common Test Conditions pp. 82, 98, 103, 105, 107–109, 112, 119, 126, 148

CTR CounTeR pp. 3, 19, 21, 60, 66, 83, 97, 98, 121

DES Data Encryption Standard p. 21

DPA Differential Power Analysis pp. 4, 29, 30, 65, 70

DRM Digital Rights Management pp. 113, 120

DSP Digital Signal Processing pp. 72, 73

ECB Electronic Code Book pp. 3, 19, 20, 121

EDR Edge Differential Ratio pp. 82, 83, 105, 126
EGk Exp-Golomb k-th order code pp. 86, 89, 91
EQ Encryption Qualilty pp. 6, 82, 83, 102–104, 112, 123
FA Fault Attack pp. 4, 27
FF Flip-Flop pp. 72, 73
FL Fixed Length pp. 86, 91
FPGA Field-Programmable Gate Array pp. 65–67, 75
HD Hamming Distance pp. 4, 5, 28, 29, 54, 55, 57, 122
HEVC High Efficiency Video Coding pp. 81–83, 86, 87, 104, 131
HW Hamming Weight pp. 4, 28, 29, 38
I frame Intra frame pp. 83, 90
IIR Infinite Impulse Response pp. 5, 35, 38, 48, 50–52
IP Initial Permutation p. 21
IPM Intra Prediction Mode p. 83
ISO International Organization for Standardization p. 82
ITU International Telecommunication Union p. 82
IV Initial Vector pp. 18, 20, 21, 23, 36, 41, 52, 54, 67, 70, 71, 75, 76
JPEG Joint Photographic Experts Group p. 81
JVET Joint Video Experts Team pp. 112, 148
LFSR Linear Feedback Shift Register pp. 35, 46, 67, 70, 121
LSB Less Significant Bits p. 93
LUT Lookup Table pp. 72, 73, 75, 88, 89, 97
LWCB LightWeight Chaos-Based pp. 6, 7, 43, 54, 60, 65, 70, 71, 73, 75, 97, 111, 113, 122
MAC Multiply–ACcumulate pp. 51, 72
MPEG Moving Picture Experts Group p. 82
MV Motion Vector pp. 82, 83, 91
NCpB Number of Cycles per Byte pp. 47, 121
NIST National Institute of Standards and Technology pp. 5, 53, 61, 111, 118, 125
NPCR Number of Pixels Change Rate pp. 56, 57, 82, 83, 106, 107, 122, 126
OFB Output FeedBack pp. 3, 19, 21, 121
OTT Over-The-Top p. 98
P frame Predicted frame pp. 83, 90

PCNG Pseudo-Chaotic Number Generator pp. 5, 25, 35, 36, 38, 40, 43, 47, 52, 54, 60, 61, 67, 69–71, 70–73, 121, 122, 125, 127

PRNG Pseudo-Random Number Generator pp. 18, 19, 71, 97, 98

PSNR Peak Signal to Noise Ration pp. 82, 99, 102, 108, 109, 112, 119, 125, 126

PWLC PieceWise Linear Chaotic pp. 4, 5, 24–26, 35–38, 40, 43, 46, 47, 52, 67–70, 72, 121, 122

QP Quantization Parameter pp. 98, 99, 101–105, 107–109, 123, 125

QTC Quantized Transform Coefficient p. 83

ROI Region Of Interest pp. 81, 83

RTL Register-Transfer Level pp. 67–69, 122

SAO Sample-Adaptive Offset pp. 83, 91

S-Box Substitution Box pp. 19, 22, 30

SC Stream Cipher pp. 43, 65, 73

SCA Side-Channel Attack pp. 3–7, 12, 13, 17, 18, 20, 22, 24, 26–28, 30, 35, 38, 50, 61, 63, 65, 70, 75, 77, 79, 111–113, 116–119, 122, 148

SHVC Scalable High Efficiency Video Coding (HEVC) pp. 81, 83

SPA Simple Power Analysis pp. 4, 29, 43

SSIM Structural SIMilarity pp. 82, 99, 101, 108, 109, 112, 119, 125, 126

SSS Shamir's Secret Sharing p. 83

ST Skew Tent pp. 4, 24, 35–37, 43, 45, 46, 111, 118, 121

STS Statistical Tests Suite pp. 5, 53, 61, 111, 118, 125

SVC Scalable Video Coding p. 81

TB Truncated Binary pp. 86, 91

TC Transform Coefficient pp. 6, 82, 83, 86–91, 90–93, 96–98, 102, 109, 112, 118, 123

TRp Truncated Rice Code with context p pp. 86, 89, 91

TS Transform Skip pp. 6, 87–90, 93, 123

TU Truncated Unary pp. 86, 89

UACI Unified Average Changing Intensity pp. 56, 57, 82, 83, 106, 107, 122, 126

U Unary pp. 86, 89

VCEG Video Coding Experts Group p. 82

VMAF Video Multimethod Assessment Fusion pp. 82, 99, 102, 108, 109, 112, 119, 125, 126

VOD Video On Demand p. 11

VTM VVC Test Model pp. 98, 109, 112, 118, 142

VVC Versatile Video Coding pp. 6, 7, 12, 13, 81–84, 86–89, 88, 90, 91, 90–93, 91, 94–98, 100, 102, 104, 106, 108–113, 116–118, 120, 123, 125, 148

XOR eXclusive OR pp. 18, 20–23, 30, 35, 46, 50, 65, 66, 71, 83, 97, 98, 127

Bibliography

[1] Mohammad Abu Taha, « Real-Time and Portable Chaos-based Crypto-Compression Systems for Efficient Embedded Architectures », Chapter 3, PhD thesis, 2017 (cit. on pp. 12, 13, 35–37, 44, 54, 55, 61, 111, 116–118).

[2] Christof Paar, Jan Pelzl, and Bart Preneel, *Understanding cryptography: a textbook for students and practitioners*, en, 2nd corrected printing, OCLC: 845804174, Berlin: Springer, 2010 (cit. on pp. 22, 66).

[3] Martin Boesgaard, Mette Vesterager, and Erik Zenner, « The Rabbit Stream Cipher », *in*: *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2008, pp. 69–83 (cit. on pp. 22, 35, 55, 61).

[4] Matthew Robshaw, « The eSTREAM Project », en, *in*: *New Stream Cipher Designs*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2008, pp. 1–6 (cit. on p. 22).

[5] Hongjun Wu, « New Stream Cipher Designs », *in*: ed. by Matthew Robshaw and Olivier Billet, Berlin, Heidelberg: Springer-Verlag, 2008, chap. The Stream Cipher HC-128, pp. 39–47 (cit. on pp. 23, 35, 55, 61).

[6] Christophe De Canniere and Bart Preneel, « Trivium », *in*: *New Stream Cipher Designs: The eSTREAM Finalists*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 244–266 (cit. on pp. 23, 65, 66).

[7] Boris Hasselblatt and Anatole Katok, *A First Course in Dynamics: with a Panorama of Recent Developments*, Cambridge University Press, 2003 (cit. on p. 23).

[8] Kathleen T. Alligood, Tim D. Sauer, and James A. Yorke, *Chaos: An Introduction to Dynamical Systems*, Springer New York, 1996 (cit. on p. 23).

[9] Franz Pichler and Josef Scharinger, « Finite dimensional generalized baker dynamical systems for cryptographic applications », *in*: *Computer Aided Systems Theory — EUROCAST '95*, ed. by Franz Pichler, Roberto Moreno Díaz, and Rudolf Albrecht, Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 465–476 (cit. on p. 24).

[10] C. Manchein and M.W. Beims, « Gauss map and Lyapunov exponents of interacting particles in a billiard », *in*: *Chaos, Solitons & Fractals* 39.*5* (Mar. 2009), pp. 2041–2047 (cit. on p. 24).

[11] M. Hénon, « A two-dimensional mapping with a strange attractor », *in*: *Communications in Mathematical Physics* 50.*1* (1976), pp. 69–77 (cit. on p. 24).

[12] Colin Sparrow, *The Lorenz equations: bifurcations, chaos, and strange attractors*, vol. 41, Springer Science & Business Media, 2012 (cit. on p. 24).

[13] R Lozi, « Un attracteur étrange (?) du type attracteur de Hénon », *in*: *Le Journal de Physique Colloques* 39.*C5* (1978), pp. C5–9 (cit. on p. 24).

[14] T Yoshida, H Mori, and H Shigematsu, « Analytic study of chaos of the tent map: band structures, power spectra, and critical behaviors », *in*: *Journal of statistical physics* 31.*2* (1983), pp. 279–308 (cit. on p. 24).

[15] M. Hasler and Y. L. Maistrenko, « An introduction to the synchronization of chaotic systems: coupled skew tent maps », *in*: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 44.*10* (1997), pp. 856–866 (cit. on p. 24).

[16] J. Khan, J. Ahmad, and S. O. Hwang, « An efficient image encryption scheme based on: Henon map, skew tent map and S-Box », *in*: *2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, 2015, pp. 1–6 (cit. on p. 24).

[17] Shiguo Lian, Jinsheng Sun, and Zhiquan Wang, « A block cipher based on a suitable use of the chaotic standard map », *in*: *Chaos, Solitons & Fractals* 26.*1* (2005), pp. 117–129 (cit. on p. 24).

[18] K. Wong, Q. Lin, and J. Chen, « Simultaneous Arithmetic Coding and Encryption Using Chaotic Maps », *in*: *IEEE Transactions on Circuits and Systems II: Express Briefs* 57.*2* (2010), pp. 146–150 (cit. on p. 24).

[19] Shujun Li et al., « Statistical Properties of Digital Piecewise Linear Chaotic Maps and Their Roles in Cryptography and Pseudo-Random Coding », *in*: *Cryptography and Coding*, ed. by Bahram Honary, Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 205–221 (cit. on p. 25).

[20] K. Wong, Q. Lin, and J. Chen, « Simultaneous Arithmetic Coding and Encryption Using Chaotic Maps », *in*: *IEEE Transactions on Circuits and Systems II: Express Briefs* 57.*2* (2010), pp. 146–150 (cit. on p. 25).

[21] J. Peng et al., « A novel scheme for image encryption based on piecewise linear chaotic map », *in*: *2008 IEEE Conference on Cybernetics and Intelligent Systems*, 2008, pp. 1012–1016 (cit. on p. 25).

[22] Jiri Fridrich, « Image encryption based on chaotic maps », *in*: *1997 IEEE international conference on systems, man, and cybernetics. Computational cybernetics and simulation*, vol. 2, IEEE, 1997, pp. 1105–1110 (cit. on p. 25).

[23] Jiri Fridrich, « Symmetric ciphers based on two-dimensional chaotic maps », *in*: *International Journal of Bifurcation and chaos* 8.*06* (1998), pp. 1259–1284 (cit. on p. 25).

[24] Shiguo Lian, Jinsheng Sun, and Zhiquan Wang, « Security analysis of a chaos-based image encryption algorithm », *in*: *Physica A: Statistical Mechanics and its Applications* 351.*2-4* (2005), pp. 645–661 (cit. on p. 26).

[25] Huaqian Yang et al., « A fast image encryption and authentication scheme based on chaotic maps », *in*: *Communications in Nonlinear Science and Numerical Simulation* 15.*11* (2010), pp. 3507–3517 (cit. on p. 26).

[26] Mohamed L Barakat, Ahmed G Radwan, and Khaled N Salama, « Hardware realization of chaos based block cipher for image encryption », *in*: *ICM 2011 Proceeding*, IEEE, 2011, pp. 1–5 (cit. on p. 26).

[27] MS Azzaz et al., « Real-time FPGA implementation of Lorenz's chaotic generator for ciphering telecommunications », *in*: *2009 Joint IEEE North-East Workshop on Circuits and Systems and TAISA Conference*, IEEE, 2009, pp. 1–4 (cit. on p. 26).

[28] El-Habib Bensikaddour, Youcef Bentoutou, and Nasreddine Taleb, « Embedded implementation of multispectral satellite image encryption using a chaos-based block cipher », *in*: *Journal of King Saud University-Computer and Information Sciences* 32.*1* (2020), pp. 50–56 (cit. on p. 26).

[29] H. Bar-El et al., « The Sorcerer's Apprentice Guide to Fault Attacks », *in*: *Proceedings of the IEEE* 94.*2* (2006), pp. 370–382 (cit. on p. 27).

[30] L. Zussa et al., « Efficiency of a glitch detector against electromagnetic fault injection », *in*: *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2014, pp. 1–6 (cit. on p. 27).

[31] Sébastien Ordas, Ludovic Guillaume-Sage, and Philippe Maurine, « Electromagnetic fault injection: the curse of flip-flops », *in*: *Journal of Cryptographic Engineering* 7.*3* (2017), pp. 183–197 (cit. on p. 27).

[32] N. Moro et al., « Electromagnetic Fault Injection: Towards a Fault Model on a 32-bit Microcontroller », *in*: *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, 2013, pp. 77–88 (cit. on p. 27).

[33] Oliver Kömmerling and Markus G Kuhn, « Design Principles for Tamper-Resistant Smartcard Processors. », *in*: *Smartcard* 99 (1999), pp. 9–20 (cit. on p. 27).

[34] Paul C Kocher, « Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems », *in*: *Annual International Cryptology Conference*, Springer, 1996, pp. 104–113 (cit. on pp. 28, 29).

[35] Paul Kocher, Joshua Jaffe, and Benjamin Jun, « Differential power analysis », *in*: *Annual international cryptology conference*, Springer, 1999, pp. 388–397 (cit. on p. 30).

[36] Eric Brier, Christophe Clavier, and Francis Olivier, « Correlation power analysis with a leakage model », *in*: *International workshop on cryptographic hardware and embedded systems*, Springer, 2004, pp. 16–29 (cit. on p. 30).

[37] Suresh Chari et al., « Towards sound approaches to counteract power-analysis attacks », *in*: *Annual International Cryptology Conference*, Springer, 1999, pp. 398–412 (cit. on p. 30).

[38] Louis Goubin and Jacques Patarin, « DES and differential power analysis the "Duplication" method », *in*: *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, 1999, pp. 158–172 (cit. on p. 30).

[39] Yuval Ishai, Amit Sahai, and David Wagner, « Private circuits: Securing hardware against probing attacks », *in*: *Annual International Cryptology Conference*, Springer, 2003, pp. 463–481 (cit. on p. 30).

[40] Mehdi-Laurent Akkar and Christophe Giraud, « An Implementation of DES and AES, Secure against Some Attacks », *in*: *Cryptographic Hardware and Embedded Systems — CHES 2001*, ed. by Çetin K. Koç, David Naccache, and Christof Paar, Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 309–318 (cit. on p. 31).

[41] Emmanuel Prouff and Matthieu Rivain, « A generic method for secure sbox implementation », *in*: *International Workshop on Information Security Applications*, Springer, 2007, pp. 227–244 (cit. on p. 31).

[42] Erik Boss et al., « Strong 8-bit Sboxes with efficient masking in hardware extended version », *in*: *Journal of Cryptographic Engineering* 7.*2* (2017), pp. 149–165 (cit. on p. 31).

[43] Shiguo Lian et al., « A chaotic stream cipher and the usage in video protection », *in*: *Chaos, Solitons and Fractals* 34.*3* (2007), pp. 851–859 (cit. on p. 35).

[44] Wassim Hamidouche et al., « Real-time selective video encryption based on the chaos system in scalable HEVC extension », *in*: *Signal Processing: Image Communication* 58 (2017), pp. 73–86 (cit. on pp. 35, 82, 84, 103, 104).

[45] Charalampos Manifavas et al., « A survey of lightweight stream ciphers for embedded systems », *in*: *Security and Communication Networks* 9.*10* (Dec. 2015), pp. 1226–1246 (cit. on pp. 35, 61).

[46] Robert Nguyen et al., « Speed-up of SCA Attacks on 32-bit Multiplications », en, *in*: *Codes, Cryptology and Information Security*, ed. by Claude Carlet et al., vol. 11445, Cham: Springer International Publishing, 2019, pp. 31–39 (cit. on p. 42).

[47] T Geisel and V Fairen, « Statistical properties of chaos in Chebyshev maps », *in*: *Physics Letters A* 105.*6* (1984), pp. 263–266 (cit. on p. 45).

[48] Ons Jallouli, « Chaos-based security under real-time and energy constraints for the Internet of Things », Chapter 4, PhD thesis, 2017 (cit. on p. 48).

[49] L. E. Bassham et al., *A statistical test suite for random and pseudorandom number generators for cryptographic applications*, tech. rep., National Institute of Standards and Technology(NIST), 2010 (cit. on p. 53).

[50] Milton Abramowitz and Irene A Stegun, *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, vol. 55, Government Printing Office, 1964 (cit. on p. 54).

[51] E. Kreyszig, *"Introductory Mathematical Statistics"*, John Wiley, 1970 (cit. on p. 54).

[52] Yue Wu, Joseph P Noonan, Sos Agaian, et al., « NPCR and UACI randomness tests for image encryption », *in*: *Cyber journals: multidisciplinary journals in science and technology, Journal of Selected Areas in Telecommunications (JSAT)* 1.*2* (2011), pp. 31–38 (cit. on pp. 57, 106).

[53] Eli Biham and Adi Shamir, « Differential cryptanalysis of DES-like cryptosystems », *in*: *Journal of CRYPTOLOGY* 4.*1* (1991), pp. 3–72 (cit. on pp. 57, 106).

[54] Farhad Maleki et al., « An image encryption system by cellular automata with memory », *in*: *2008 Third International Conference on Availability, Reliability and Security*, IEEE, 2008, pp. 1266–1271 (cit. on pp. 57, 107).

[55] Umer Farooq and M. Faisal Aslam, « Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA », en, *in*: *Journal of King Saud University - Computer and Information Sciences* 29.*3* (July 2017), pp. 295–302 (cit. on pp. 65, 66, 74, 75).

[56] Paweł Chodowiec and Kris Gaj, « Very Compact FPGA Implementation of the AES Algorithm », en, *in*: *Cryptographic Hardware and Embedded Systems - CHES 2003*, ed. by Gerhard Goos et al., vol. 2779, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 319–333 (cit. on pp. 65, 66, 74, 75).

[57] Camel Tanougast, « Hardware Implementation of Chaos Based Cipher: Design of Embedded Systems for Security Applications », en, *in*: *Chaos-Based Cryptography*, vol. 354, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 297–330 (cit. on pp. 65, 67, 74, 75).

[58] S. Sadoudi et al., « Real-time FPGA implementation of Lü's chaotic generator for cipher embedded systems », *in*: *2009 International Symposium on Signals, Circuits and Systems*, 2009, pp. 1–4 (cit. on pp. 65, 67, 74, 75).

[59] 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)*, June 2010, pp. 76–87 (cit. on pp. 65, 71).

[60] Tobias Schneider and Amir Moradi, « Leakage Assessment Methodology », *in*: *Cryptographic Hardware and Embedded Systems – CHES 2015*, ed. by Tim Güneysu and Helena Handschuh, Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 495–513 (cit. on pp. 66, 75, 79, 112, 118).

[61] Martin Boesgaard, Mette Vesterager, and Erik Zenner, « The Rabbit Stream Cipher », en, *in*: *New Stream Cipher Designs*, vol. 4986, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 69–83 (cit. on p. 66).

[62] Deian Stefan, « Hardware Framework for the Rabbit Stream Cipher », en, *in*: *Information Security and Cryptology*, vol. 6151, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 230–247 (cit. on pp. 66, 74, 75).

[63] Daniel J. Bernstein, « The Salsa20 Family of Stream Ciphers », en, *in*: *New Stream Cipher Designs*, vol. 4986, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 84–97 (cit. on p. 66).

[64] Jarosław Sugier, « Implementing Salsa20 vs. AES and Serpent Ciphers in Popular-Grade FPGA Devices », en, *in*: *New Results in Dependability and Computer Systems*, vol. 224, Heidelberg: Springer International Publishing, 2013, pp. 431–438 (cit. on pp. 66, 74, 75).

[65] Kris Gaj et al., « Comparison of hardware performance of selected Phase II eSTREAM candidates », en, *in*: (), p. 11 (cit. on pp. 66, 67, 74, 75).

[66] David Hwang et al., « Comparison of FPGA-Targeted Hardware Implementations of eSTREAM Stream Cipher Candidates », en, *in*: (), p. 12 (cit. on pp. 66, 67, 74, 75).

[67] George Marsaglia, « Xorshift RNGs », *in*: *Journal of Statistical Software, Articles* 8.*14* (2003), pp. 1–6 (cit. on p. 71).

[68] Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al., « A testing methodology for side-channel resistance validation », *in*: *NIST non-invasive attack testing workshop*, vol. 7, 2011, pp. 115–136 (cit. on pp. 75, 79).

[69] *Side-channel AttacK User Reference Architecture. Sakura-X/Sasebo-GIII presentation page*, URL: http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-X.html (cit. on p. 75).

[70] *Specification for the Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publication 197, 2001 (cit. on p. 81).

[71] M. Van Droogenbroeck, « Partial encryption of images for real-time applications », *in*: *IEEE Signal Processing Symposium* 1.*2* (2004), pp. 11–15 (cit. on p. 82).

[72] Engel Dominik, Stütz Thomas, and Uhl Andreas, « A survey on JPEG2000 encryption », *in*: *IEEE Multimedia Systems* 15.*4* (2009), pp. 243–270 (cit. on p. 82).

[73] Z. Shahid, M. Chaumont, and W. Puech, « Fast Protection of H.264/AVC by Selective Encryption of CAVLC and CABAC for I and P Frames », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 21.*5* (May 2011), pp. 565–576 (cit. on p. 82).

[74] S. Park and S. Shin, « Efficient Selective Encryption Scheme for the H.264/Scalable Video Coding(SVC) », *in*: *2008 Fourth International Conference on Networked Computing and Advanced Information Management*, vol. 1, Sept. 2008, pp. 371–376 (cit. on p. 82).

[75] Zafar Shahid and William Puech, « Visual protection of HEVC video by selective encryption of CABAC binstrings », *in*: *ieee transactions on multimedia* 16.*1* (2013), pp. 24–36 (cit. on pp. 82, 83).

[76] Mousa Farajallah et al., « ROI encryption for the HEVC coded video contents », *in*: *2015 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2015, pp. 3096–3100 (cit. on pp. 82, 83).

[77] B. Boyadjis et al., « Extended Selective Encryption of H.264/AVC (CABAC)- and HEVC-Encoded Video Streams », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 27.*4* (Apr. 2017), pp. 892–906 (cit. on pp. 82, 84).

[78] Glenn Van Wallendael et al., « Encryption for high efficiency video coding with video adaptation capabilities », *in*: *IEEE Transactions on Consumer Electronics* 59.*3* (2013), pp. 634–642 (cit. on pp. 82, 84, 85).

[79] Vasileios A Memos and Kostas E Psannis, « Encryption algorithm for efficient transmission of HEVC media », *in*: *Journal of Real-Time Image Processing* 12.*2* (2016), pp. 473–482 (cit. on pp. 82, 84).

[80] Min Long, Fei Peng, and Xiaoqing Gong, « A Format-Compliant Encryption for Secure HEVC Video Sharing in Multimedia Social Network », *in*: *International Journal of Digital Crime and Forensics (IJDCF)* 10.*2* (2018), pp. 23–39 (cit. on pp. 82, 85).

[81] Ahmed I Sallam, El-Sayed M El-Rabaie, and Osama S Faragallah, « Efficient HEVC selective stream encryption using chaotic logistic map », *in*: *Multimedia Systems* 24.*4* (2018), pp. 419–437 (cit. on pp. 82, 85).

[82] N. Sidaty et al., « Compression Performance of the Versatile Video Coding: HD and UHD Visual Quality Monitoring », *in*: *2019 Picture Coding Symposium (PCS)*, Nov. 2019, pp. 1–5 (cit. on p. 82).

[83] C. E. Shannon, « Communication theory of secrecy systems », *in*: *Declassified Report, Bell Systems Technical Journal* 28 (Dec. 1949), pp. 656–715 (cit. on p. 82).

[84] H. E. H. Ahmed, H. M. Kalash, and O. S. F. Allah, « Encryption Efficiency Analysis and Security Evaluation of RC6 Block Cipher for Digital Images », *in*: *2007 International Conference on Electrical Engineering*, Apr. 2007, pp. 1–7 (cit. on pp. 82, 103).

[85] Nidhi Taneja, Balasubramanian Raman, and Indra Gupta, « Chaos based partial encryption of spiht compressed images », *in*: *International Journal of Wavelets, Multiresolution and Information Processing* 9.*02* (2011), pp. 317–331 (cit. on pp. 82, 105).

[86] Yue Wu et al., « NPCR and UACI Randomness Tests for Image Encryption », *in*: *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT*, 2011 (cit. on p. 82).

[87] Adi Shamir, « How to share a secret », *in*: *Communications of the ACM* 22.*11* (1979), pp. 612–613 (cit. on p. 84).

[88] V Vijayalakshmi, LM Varalakshmi, and G Florence Sudha, « Efficient encryption of intra and inter frames in MPEG video », *in*: *International Conference on Network Security and Applications*, Springer, 2010, pp. 93–104 (cit. on p. 84).

[89] Fei Peng et al., « A tunable selective encryption scheme for H. 265/HEVC based on chroma IPM and coefficient scrambling », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* (2019) (cit. on p. 85).

[90] Dawen Xu, « Data hiding in partially encrypted HEVC video », *in*: *ETRI Journal* (2020) (cit. on p. 85).

[91] S. Kim J. Chen Y. Ye, *Algorithm description for Versatile Video Coding and Test Model 10 (VTM 10)*, tech. rep. S2002, JVET, July 2020 (cit. on p. 86).

[92] V. Sze and M. Budagavi, « High Throughput CABAC Entropy Coding in HEVC », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 22.*12* (2012), pp. 1778–1791 (cit. on p. 86).

[93] Helger Lipmaa, Phillip Rogaway, and David Wagner, « Comments to NIST concerning AES modes of operations: CTR-mode encryption », *in*: *National Institute of Standards and Technologies*, Citeseer, 2000 (cit. on p. 98).

[94] Martin Boesgaard, Mette Vesterager, and Erik Zenner, « The Rabbit Stream Cipher », en, *in*: *New Stream Cipher Designs*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2008, pp. 69–83 (cit. on p. 98).

[95] Guillaume Gautier et al., « Hardware Implementation of Lightweight Chaos-Based Stream Cipher », *in*: *International Conference on Cyber-Technologies and Cyber-Systems*, CYBER 2019, Porto, Portugal, Sept. 2019, 5 pages, URL: https://hal.archives-ouvertes.fr/hal-02184571 (cit. on p. 98).

[96] Hongjun Wu, « The Stream Cipher HC-128 », en, *in*: *New Stream Cipher Designs*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2008, pp. 39–47 (cit. on p. 98).

[97] *Git repository of the VTM*, URL: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM (cit. on p. 98).

[98] Stefan Winkler and Praveen Mohandas, « The evolution of video quality measurement: From PSNR to hybrid metrics », *in*: *IEEE transactions on Broadcasting* 54.*3* (2008), pp. 660–668 (cit. on p. 99).

[99] Zhou Wang et al., « Image quality assessment: from error visibility to structural similarity », *in*: *IEEE transactions on image processing* 13.*4* (2004), pp. 600–612 (cit. on p. 99).

[100] R. Rassool, « VMAF reproducibility: Validating a perceptual practical video quality metric », *in*: *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, June 2017, pp. 1–2 (cit. on p. 99).

[101] Rawan Qumsieh, Mousa Farajallah, and Rushdi Hamamreh, « Joint block and stream cipher based on a modified skew tent map », *in*: *Multimedia Tools and Applications* 78.*23* (2019), pp. 33527–33547 (cit. on p. 104).

[102] Michael Lewis-Beck, *Data analysis: An introduction*, 103, Sage, 1995 (cit. on p. 104).

[103] Nidhi Taneja, Balasubramanian Raman, and Indra Gupta, « Selective image encryption in fractional wavelet domain », *in*: *AEU-International Journal of Electronics and Communications* 65.*4* (2011), pp. 338–344 (cit. on p. 105).

[104] Narendra K Pareek, Vinod Patidar, and Krishan K Sud, « Diffusion–substitution based gray image encryption scheme », *in*: *Digital signal processing* 23.*3* (2013), pp. 894–901 (cit. on p. 106).

[105] Behrouz A Forouzan, *Cryptography & network security*, McGraw-Hill, Inc., 2007 (cit. on p. 107).

[106] Amir Said, « Measuring the strength of partial encryption schemes », *in*: *IEEE International Conference on Image Processing 2005*, vol. 2, IEEE, 2005, pp. II–1126 (cit. on p. 108).

[107] SuGil Choi, Jong-Wook Han, and Hyunsook Cho, « Privacy-Preserving H. 264 Video Encryption Scheme », *in*: *ETRI Journal* 33.*6* (2011), pp. 935–944 (cit. on p. 108).

[108] Thomas Stütz and Andreas Uhl, « On JPEG2000 error concealment attacks », *in*: *Pacific-Rim Symposium on Image and Video Technology*, Springer, 2009, pp. 851–861 (cit. on p. 108).

[109] Frederic Dufaux and Touradj Ebrahimi, « Scrambling for privacy protection in video surveillance systems », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 18.*8* (2008), pp. 1168–1174 (cit. on p. 108).

List of my publications

Conferences

1. **Guillaume Gautier**, Safwan El Assad, Olivier Déforges, Sylvain Guilley, Adrien Facon, Wassim Hamidouche, «Enhanced Software Implementation of a Chaos-Based Stream Cipher», in: *SECURWARE 2018*, Venise, Italy, Sept. 2018, pp. 128–133.
2. Robert Nguyen, Adrien Facon, Sylvain Guilley, **Guillaume Gautier**, Safwan El Assad, «Speed-up of SCA Attacks on 32-bit Multiplications», in: *International Conference on Codes, Cryptology And Information Security, C2SI 2019*, Rabat, Morocco, Apr. 2019, pp. 31–39.
3. **Guillaume Gautier**, Maguy Le Glatin, Safwan El Assad, Wassim Hamidouche, Olivier Déforges, Sylvain Guilley, Adrien Facon, «Hardware Implementation of Lightweight Chaos-Based Stream Cipher», in: *International Conference on Cyber-Technologies and Cyber-Systems, CYBER 2019*, Porto, Portugal, Sept. 2019, 5 pages.

Journal

1. Mousa Farajallah, **Guillaume Gautier**, Wassim Hamidouche, Olivier Déforges, Safwan El Assad, «Selective Encryption of the Versatile Video Coding Standard», in: *IEEE Access*.

INSA de RENNES
Service des Formations

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:
Conception de Solutions Matérielles de Chiffrement Basées Chaos pour des Applications Vidéo Sécurisées

Nom Prénom de l'auteur : GAUTIER GUILLAUME

Membres du jury :
- Monsieur HAMIDOUCHE Wassim
- Monsieur LOZI René
- Monsieur LEVEUGLE Régis
- Madame BOCHARD Nathalie
- Monsieur PUECH William
- Monsieur EL ASSAD Safwan
- Monsieur DEFORGES Olivier

Président du jury : William PUECH

Date de la soutenance : 09 Décembre 2020

Reproduction de la these soutenue

☒ Thèse pouvant être reproduite en l'état
☐ Thèse pouvant être reproduite après corrections suggérées

Fait à Rennes, le 09 Décembre 2020

Signature du président de jury

William PUECH

Le Directeur,

INSTITUT NATIONAL des SCIENCES APPLIQUÉES - RENNES -

Abdellatif MIRAOUI

DOCTORAT BRETAGNE LOIRE / MATHSTIC

INSA RENNES

Titre : Conception de Solutions Matérielles de Chiffrement Basées Chaos pour des Applications Vidéo Sécurisées

Mot clés : Cryptographie Basée Chaos, Attaques par Canaux Auxiliaires, Versatile Video Coding, Chiffrement Sélectif.

Résumé : De par de la prolifération des objets connectés, le développement de systèmes cryptographiques sécurisés et à faibles ressources est devenu un véritable défi. De nombreux algorithmes de cryptage n'ont pas été conçus initialement pour être mis en œuvre sur des plateformes embarquées aux ressources informatiques, de mémoire et énergétiques limitées. De plus, les méthodes d'Attaque par Canaux Auxiliaires (SCAs) sont de plus en plus efficaces, le coût de leurs applications diminue, de sorte que le besoin d'un chiffrement sécurisé, léger et orienté matériel est considérable.

Dans cette thèse, après avoir évalué la sécurité d'un chiffrement par flux basé chaos, une conception avancée de ce chiffrement par flux est proposée. Une implémentation matérielle de ce nouveau chiffrement est introduite et évaluée par rapport aux SCAs. Dans une dernière partie, une méthode de cryptage sélectif pour la prochaine norme de codage vidéo Versatile Video Coding (VVC) est présentée. Elle est appliquée sur des séquences vidéos suivant les Conditions de Test Communes (CTC) du standard, en utilisant un chiffrement standardisé Advanced Encryption Standard (AES).

Title: Lightweight Hardware Design of a Chaos-Based Stream Cipher for Secure Video Applications

Keywords: Chaos-based Cryptography, Side-Channel Attacks, Versatile Video Coding, Selective Encryption.

Abstract: Due to the proliferation of connected devices, the development of secured and low-resource cryptographic systems has become a real challenge. Many encryption algorithms were not designed to be implemented on embedded platforms with limited computing, memory and energy resources. Moreover, Side-Channel Attack (SCA) are more and more efficient, the cost of their applications is decreasing so the need for a secure lightweight hardware-friendly cipher is considerable.

In this thesis, after assessing the security of an exciting chaos-based stream cipher, an enhanced design of this stream cipher is proposed. Then, a hardware implementation of this new cipher is introduced and assessed against SCA. Finally, a selective encryption method for the next video coding standard Versatile Video Coding (VVC) is proposed and assessed under the Joint Video Experts Team (JVET) Common Test Conditions (CTC) sequences using Advanced Encryption Standard (AES).