Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή
Στην Ασφάλεια Υπολογιστών και Δικτύων

Μελέτη της Ασφάλειας των Αυθεντικοποιημένων Κρυπτογραφικών Αλγόριθμων

Χρυσόστομος Κωνσταντίνου

Επιβλέπων Καθηγητής
Κωνσταντίνος Λιμνιώτης

Μάιος 2019

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μελέτη της Ασφάλειας των Αυθεντικοποιημένων Κρυπτογραφικών Αλγόριθμων

Χρυσόστομος Κωνσταντίνου

Επιβλέπων Καθηγητής
Κωνσταντίνος Λιμνιώτης

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στην Ασφάλεια Υπολογιστών και Δικτύων

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Μάιος 2019

Περίληψη

Οι αυθεντικοποιημένοι κρυπτογραφικοί αλγόριθμοι έχουν προσελκύσει την προσοχή της κρυπτογραφικής κοινότητας τα τελευταία χρόνια. Η ανάπτυξη τέτοιων αλγορίθμων και η μελέτη τους έχουν αυξηθεί κατακόρυφα λόγω της υψηλής ασφάλειας που προσφέρουν. Προκειμένου λοιπόν η διατριβή να συμβαδίζει με την εποχή και να συνεισφέρει στην κρυπτογραφική κοινότητα, επιλέχθηκε να μελετηθεί η ασφάλεια των αυθεντικοποιημένων κρυπτογραφικών αλγόριθμων ροής.

Στη διατριβή αναλύονται αρχικά οι αυθεντικοποιημένοι κρυπτογραφικοί αλγόριθμοι ροής. Έπειτα περιγράφεται μια πρόσφατα ανεπτυγμένη τεχνική για την εύρεση προσεγγίσεων λογικών συναρτήσεων, η οποία μπορεί να ενισχύσει γνωστές κρυπταναλυτικές επιθέσεις. Η εν λόγω τεχνική αναλύεται διεξοδικά ώστε να μπορεί να έχει το καλύτερο δυνατό αποτέλεσμα. Στη συνέχεια περιγράφονται τρεις αυθεντικοποιημένοι κρυπτογραφικοί αλγόριθμοι ροής, οι συναρτήσεις των οποίων υποβάλλονται στην τεχνική που προαναφέρθηκε.

Η συνεισφορά της διατριβής μπορεί εν συντομία να συνοψιστεί σε τρία σημεία. Πρώτον, καταγράφεται η ανάλυση της ασφάλειας τριών αυθεντικοποιημένων κρυπτογραφικών αλγορίθμων ροής, οι δύο εκ των οποίων έχουν απασχολήσει αρκετά την ερευνητική κοινότητα τα τελευταία χρόνια, ενώ ο τρίτος είναι πολύ πρόσφατος και δεν έχει ακόμη μελετηθεί εκτενώς. Δεύτερον, στη διατριβή αυτή γίνεται ανάλυση μιας πρόσφατης τεχνικής εύρεσης προσεγγίσεων, με απόδειξη νέων αποτελεσμάτων που επιτρέπουν πιο αποτελεσματική εφαρμογή αυτής. Η τεχνική εύρεσης προσεγγίσεων μπορεί να αποτελέσει εργαλείο για κατασκευή ισχυρών κρυπτογραφικών συναρτήσεων που να είναι ανθεκτικές σε προσεγγίσεις αλλά και να χρησιμοποιηθούν σε συνδυασμό με άλλες τεχνικές κρυπτανάλυσης για τη δημιουργία επιθέσεων εναντίων κρυπτογραφικών αλγορίθμων. Τέλος, η τεχνική αυτή εφαρμόζεται στους τρεις αλγορίθμους οι οποίοι αναφέρονται προηγουμένως με τα αποτελέσματα να δίνουν αρκετές πληροφορίες για τη δραστικότητα της.

Summary

Authenticated stream ciphers have attracted great attention of the cryptographic community last years. The development of such ciphers and their study have increased dramatically due the high level of security they offer. Therefore, this thesis studies the security of authenticated ciphers, as a contemporary research topic.

Initially, authenticated stream ciphers are analysed in the thesis. After this, a recently developed method for approximating cryptographic functions is described, which can be subsequently used to enhance known cryptanalytic attacks. This technique is further analysed so as to be improved in terms of efficiency. Subsequently, three authenticated stream ciphers are being studied, as case studies for exploring the strengthness of this technique.

The contribution of this thesis can be summarized in three points. Firstly, a security analysis of three authenticated stream ciphers is performed, whereas two of them have been greatly studied by the research community during the last years and the third is a recent cipher, not having yet studied and evaluated to a great extent. Secondly, an approximation technique is analysed, with the aim to reach conclusions that allow an increase of its effectiveness. The approximation technique can be used as a tool for the construction of cryptographic functions that are resistant to approximations and also be combined with other cryptanalytic techniques for the creation of attacks against ciphers. Finally, the technique is applied to the three ciphers that are mentioned before and the results provide us with information about its potency.

Table of Contents

Chapter 1
Introduction

Authenticated ciphers have been around only for the last decades but there is already a massive amount of research that concerns them due to their necessity. So, this thesis will not examine these ciphers' s security from a surface level because it has been done. Instead, it will focus on a specific kind of authenticated ciphers and it will present how and why these ciphers react to a newly developed approximation technique[1]. Specifically, the authenticated stream ciphers will be examined and the technique that will be used is able to find approximation functions that depend on less variables. Taking advantage of the opportunity, I would like to thank my supervisor because this thesis wouldn't be possible without his help. In the next paragraphs, there is a short description of how authenticated stream ciphers came to be, a paragraph that summarizes the work done in this thesis regarding the approximation technique and in the last paragraph is stated what is included in the following chapters.

The world of cryptography has vastly expanded the last decades, although it has been around for centuries. New ciphers are being designed nonstop because of the swift increase of the resources in computer systems and the new cryptanalysis methods that are being exposed (to the public). So, algorithms are needed that can provide resistance to at least conventional known attacks like algebraic, distinguishing or cube attacks. From this perspective higher security is the purpose.

There are also ciphers that aim for speed. Another reason for the necessity of new ciphers are the low resources devices. Most electronic devices these days are connected to the internet and there must be a secure channel through encryption. So, ciphers that utilize fewer resources are designed. This large amount of ciphers has been divided in some categories.

In general, the modern ciphers are being categorised to symmetric and asymmetric with the symmetric being divided to block and stream ciphers. There are of course more subcategories depending on the features and operation of each cipher. One of those, are the authenticated stream ciphers and their security is the main subject of this thesis.

Stream ciphers are very common and important in the world of cryptography. The reasons behind this are their speed, simplicity and ease of implementation on hardware. That's why stream ciphers are preferred in specific sectors, like GSM. In stream ciphers, typically plaintexts bits are encrypted through an operation (usually XOR) one at a time with the corresponding keystream generator's bits, resulting in ciphertext bits – in the contrary to block ciphers that work on fixed blocks of bits. The keystream of an ideal stream cipher would be random, but because the sequences of each cipher are produced from the same operations, the keystream can't be absolutely random and is called pseudorandom. These operations are typically shift registers that use random unique initial values which are called seeds. The same values are also used in the decryption process. This in a way is the philosophy behind stream ciphers.

From that point, each stream cipher has different designs and specifications. Of course, each cipher has its own flaws too. Some known attacks on stream ciphers are the reused key, the bit-flipping and the chosen-*IV* attacks. In response to the attacks, there are counter measures and as concerns the bit-flipping attack, it can be prevented with a message authentication code (MAC). Surely, that's not the only reason a MAC is used, because nowadays message authentication is an integral part of all modern cryptosystems and not just to stream ciphers.

Message authentication is a property that increases security because it ensures that the data of the message has not been modified during transit (integrity) and that the receiving entity can verify the sending one. This is typically achieved by using authenticated encryption (AE), message authentication codes (MACs), and digital signatures, with the latest being used for asymmetric ciphers. Stream ciphers use MACs as mentioned before. At first the encryption and the generation of the MAC were entirely different processes and this was the source for some problems which are described in *Chapter 2*. So, the need for the unification of these processes popped up. As a result, we

have the authenticated encryption which simultaneously assures the confidentiality and the authentication of the data. This kind of structure to a cipher, led to the creation of authenticated stream ciphers.

In this thesis, a new method for analysing weaknesses in cryptographic Boolean functions is being studied, with emphasis on analysing relevant cryptographic properties in authenticated ciphers lying in the class of stream ciphers. More precisely, the thesis focus on a very recent technique to find out, in an efficient way, how well a cryptographic function can be approximated by another function with fewer number of variables. Such approximations could possibly be the starting point for subsequently mounting successful attacks and, thus, they are of high importance. The aforementioned technique is based on appropriately using a known algorithm for computing the error linear complexity spectrum of sequences, namely the Lauder-Paterson algorithm, via uniquely associating each truth table of a Boolean function on n variables with a well-determined sequence of period 2^n. Since this technique strongly depends on a proper ordering of the input variables in the function, the thesis also further elaborated towards proving results that allow for efficiently choosing the optimal such ordering – i.e. the ordering that is the most probable to reveal whether the corresponding function is weak or not, under this cryptographic criterion.

The thesis is organised as follows. In *Chapter 2* there is a detailed analysis of the authenticated stream ciphers. Later on, in *Chapter 3* the Error Linear Complexity Spectrum is explained, followed by an analysis of Boolean functions. At the end of the chapter, a link is established between the two that forms the base of the approximation technique [1] alongside an examination of how the technique works and how the technique can be used in the most favourable way. In *Chapter 4*, an overview of the ciphers that will be used to test the approximation technique and in *Chapter 5* the technique applied to the ciphers and the results are presented. For the end, *Chapter 6* includes the conclusions and suggests new studies that can be conducted based on this thesis.

Chapter 2
Authenticated Stream Ciphers

Before analysing authenticated stream ciphers, it is required to make a reference separately for stream ciphers and authenticated encryption. For both of them a detailed analysis follows that present how the stream ciphers that offers AE emerged and developed.

2.1 Stream Ciphers

Modern stream ciphers are designed to be computationally – and not unconditionally – secure. That means that if an attacker had infinite resources to attack the cipher, it would break. The unconditional security means that even if the attacker had infinite resources, the cipher would be unbreakable. The inspiration for creating stream ciphers was given by an unconditionally secure cipher, the one-time pad (OTP).

2.1.1 Foundations

The OTP, also known as Vernam cipher [2], was basically a stream cipher, in which the random secret key is the same size, or longer than the message and is generated by a true random number generator (TRNG). The key is securely distributed to the legitimate parties. Note that it is necessary for the key stream to be truly random or else the cipher is not perfectly secure. Thus, we have a perfect cipher but we can see that is not used in modern technology. That's because the OTP is impractical. The reasons are the need of TRNG, which most PCs and smartphones don't have, the secure transport of the keystream from the one party to the other and the most important is the need of one key bit for each plaintext bit. These problems led to the creation of the modern practical stream ciphers which replaced the truly random keystream with a pseudorandom keystream that uses a key as a seed.

Stream ciphers are symmetric-key algorithms that are used to provide confidentiality, which ensures that the message is only disclosed to authorized entities. In order to do this, they produce a sequence of elements over the finite field $GF\ (2)\ =\ \{0,1\}$ that depends on the secret key. For the encryption process, a group operation combines each plaintext symbol with the corresponding keystream symbol and the ciphertext is computed. This ciphertext is transmitted to the receiver via insecure channels and with the use of the secret key the receiver decrypts it. Stream ciphers are superior in speed compared to block ciphers due to their lower hardware complexity, but if used incorrectly, they are susceptible to serious security problems.

2.1.2 Random Number Generators

The number generators play a major role in the modern stream ciphers. The security of the ciphers highly depends on the randomness of the numbers that are generated. The number generators can be truly random or pseudorandom. The main feature of the TRNGs is the uniqueness in each sequence it produces. This happens because the generator depends on physical processes, like semiconductor noise. This feature makes TRNGs ideal for producing session keys in cryptography, but impractical in generating a keystream, because two parties will not be able to generate the same keystream. On the other hand, pseudorandom number generators (PRNGs) initially take a seed value and through computations, having the seed as a starting point, sequences of numbers are generated. Cryptographically good PRNGs are those having good statistical properties, which means that their outcome is close to the one of a true random number generator. However, these number generators can't be used because even with a small part of the plaintext the ciphertext can be decrypted very easy with a simple attack [3].

The ones that can be used in stream ciphers are a special type of PRNGs, the cryptographically secure pseudorandom number generators (CSPRNGs), which have the feature of being unpredictable. That means that given n number of bits of the sequence, it isn't possible to compute the following or the preceding bits with better chance of success than 50%. These deterministic number generators are suitable for the generation of keystreams in stream ciphers.

2.1.3 Feedback Shift Registers

The right theoretical framework was found, but the practical one was missing. There was the problem of how the stream ciphers will generate such sequences with few CPU instructions for software implementation and easily adapt on hardware operations for hardware implementation. There are many proposals in the literature, but the most prominent is the shift registers with feedback. They are being used in stream cipher designs because they offer large periods, efficiency and good statistical properties.

The pseudorandom sequences that will be generated in stream ciphers must have certain properties. Linear Feedback Shift Registers (LFSRs) are widely used and studied [4] in cryptography because of their useful properties in generating pseudorandom sequences. Since their mathematical properties are well-understood to the research community, it is easy to find LFSRs that produce max period 2^n, given n. Although LFSRs produce sequences with good statistical properties, they are cryptographically weak because they are completely linear. If $2c(s)$ (where $c(s)$ is the linear complexity) of the output are leaked, the sequence can be computed using the Berlekamp-Massey algorithm [5]. Therefore, the stream ciphers need to have high nonlinearity to avoid this weakness.

Nonlinearity is the criterion that determines the minimum distance of a function f from any affine/linear function. There were many design attempts to add nonlinearity to ciphers. Some attempts were based only on LFSRs, like using a nonlinear function to combine outputs of LFSRs or with a nonlinear filter of the LFSR state [6]. However, these types of approaches don't always offer the desired security level to the cipher [7]. There are limitations to the LFSRs, in contrast with the Nonlinear Feedback Shift Registers (NFSRs), which are used in most cases of recently designed stream ciphers [8], [9]. Consequently, NFSRs can offer high nonlinearity, but there are not well-understood like LFSRs. For example, there is not an efficient way to find such functions with a maximum period 2^n. Golomb [4] presented a method of creating maximum period NFSRs but their corresponding feedback functions have low nonlinearity and they can be approximated with affine

functions. Someone can find more on the properties of the NFSRs in these studies [10], [11]. The optimum solution to the construction of a stream cipher for security, is considered to be the use of both LFSRs and NFSRs.

2.1.4 Synchronous and Asynchronous

In the design of a stream cipher there are two essential procedures. The one is the update of the state of the cipher and the other is the interaction of the plaintext with the state that creates the ciphertext. The latter is mostly a bit-wise exclusive-or of the function of the state (keystream) and the plaintext. The former is a little more complicated and it divides the stream ciphers into two types, *synchronous* and *asynchronous*.

If the state is updated independently, without the use of the plaintext or the ciphertext, then the stream cipher is called *synchronous*. Since the state isn't affected, the corruption of one bit in the ciphertext will not affect the next ciphertext bits. As a result, the cipher has no *error-propagation*. This seems to be desirable, but it has its drawbacks, such as the possibility of an attacker that will make controlled changes to some bits of the ciphertext, knowing well the corresponding plaintext. From a practical perspective, it is important for encryption and decryption units to be in step because the encrypt and decrypt processes must be synchronized. To accomplish this, *marker positions* are usually used in the transmission.

On the opposite, the *asynchronous* stream ciphers compute the next state using previously generated ciphertext bits. This way, if the synchronization between the encryption and the decryption is lost, the decryption process is resumed correctly. This type of stream cipher has limited error propagation. If a bit is incorrect, then the following bits that are affected by that bit may be incorrect. This may seem as an advantage considering the attack that was described in the previous paragraph, but Rueppel [12] managed to deliver two disadvantages when using self-synchronizing stream ciphers. The first is that an attacker can identify some of the variables that are being used by the generator and the second is the inability to fully analyse those generators because the keystream depends on the plaintext.

2.1.5 Design

Until this point, it was remarked that stream ciphers need high nonlinearity, but that's not the only consideration that the designers keep in mind. A very important factor is the period, which must be of great length. If the period is short, then some identical parts of the plaintext may be encrypted in the same way. It is not stated in the literature the exact length that is required for a period, but it would be optimum if the same part of the keystream is not reused during the encryption. It is also necessary for the sequences of stream ciphers to have good statistical properties. This can happen by following Golomb' s randomness postulates [4]. Of course, these alone are not enough to offer good pseudorandom sequences and there are various statistical tests [12]–[14] that can be applied to sequences to assess the randomness. Another important consideration is the complexity of the sequence, which needs to be high. The complexity indicates how hard the sequence can be reproduced. The most popular technique to measure that complexity is the linear complexity [15]. Thus, the design and the construction of stream ciphers must be based on these considerations or else the cipher will be cryptographically weak.

2.1.6 Initialization Vector

Almost all stream ciphers nowadays have two and not one inputs: a secret key k and the initialization vector IV. The secret key is used for the encryption and decryption, like in all other symmetric cryptosystems. The IV, which takes a new value for each encryption process, is used as a randomizer. Arbitrary numbers that are used once, like the values of IV, are called *nonces* in cryptography. The primary purpose of IV is to ensure that an attacker can't reuse old communications in replay attacks. In brief, replay attacks can be done when an attacker has the plaintext of an encryption and he can compute the corresponding keystream. If another encryption uses the same keystream, then the attacker can easily decipher the ciphertext. It is also important to note about the IV, that it doesn't have to be secret like the key.

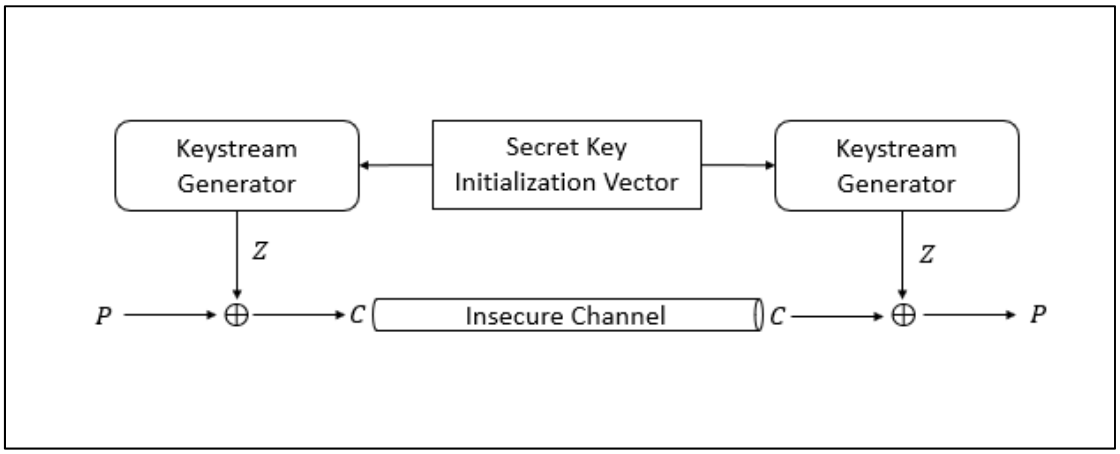

Figure 2.1: Stream cipher

2.2 Authenticated Encryption

Most people think that the most important goal in cryptography is confidentiality (or privacy as it is usually called), but usually message authentication has a more significant role. Just thinking about it, it's not always of great significance for the message to stay private, but it's always essential to know the source of the message. This is being done through message authentication, by detecting a message that has been modified en route from the sender to the receiver. It is important to note that the detection is not absolutely certain. Data integrity is also a property of message authentication. This means that the message is received by the authorized party as it was sent out, without any modification. This significance of message authentication gave birth to the idea of creating authenticated encryption schemes that would provide both confidentiality and authenticity. Authenticated encryption is the process of transforming plaintext to ciphertext by an encryption algorithm and attaching an authentication tag that was generated by an authentication algorithm.

2.2.1 Early work

The idea of creating algorithms that combine message authentication and confidentiality was first introduced by Bellare and Namprempre [16]. They described the beneficial conditions of a symmetric encryption scheme combined with a MAC algorithm and identified the security notions for a generic scheme of Authenticated Encryption (AE). In the same paper, there are also various definitions, including the ones of privacy and authenticity. About a year later, Katz and Yung also published a study on AE [17]. The latter also describes parametres and identify security notions of a generic scheme of AE that combine a symmetric encryption with a MAC algorithm. Based on both

previously mentioned papers [16], [17], this new scheme needs two keys to function, one for each algorithm. The construction of this kind of scheme has a major drawback. The data need to pass from an algorithm twice, one time for confidentiality and another for authenticity and this is a major problem for efficiency. This would be the initial point of many design attempts to build a secure and efficient authenticated encryption scheme.

Some early design attempts to create cryptosystems that provided confidentiality and authenticity did not meet the expectations of the security standards. Specifically, Gligor and Linsday [18] proposed a mechanism that was based on various cryptographic techniques and data redundancy. The authentication in this mechanism can be achieved with a signature and it is stated in the study that «*The more redundant bits included in the signature, the more reliable is the authentication procedure*». A similar cryptographic technique that only provides authenticity is also found in the literature [19]. It is based on cryptographic check digits which can be likened with data redundancy to the message. These proposals were found vulnerable to chosen plaintext and chosen ciphertext attacks due to the redundancy [20]. This was a major weakness for the mechanisms that were based on data redundancy, so the designers decided to construct new mechanisms that would provide authenticated encryption without security flaws.

2.2.2 Schemes

AE schemes had been studied for more than 20 years, but only recently there is so much interest, due to the recognition of the significance of AE [16], [17], [21]. This happened because problems were emerging from the combination of privacy-only encryption and message authentication code (MAC) [16], [22], [23] leading to the creation of the first class of AE schemes [24], [25]. All of the schemes can have efficient constructions and offer sufficient security. These are shared-key mechanisms that through processing transform a message M into a ciphertext C that can be sent safely to the receiver because both confidentiality and authenticity are protected. Each one of them has important practical applications. As it was mentioned in the introduction, message authentication in asymmetric cryptography is done with digital signatures. In the case of symmetric ciphers, the sender shares a secret key with the receiver to authenticate their message. So, the following approaches to message authentication are based on a shared key relationship between the sender and the receiver.

There are three approaches that will be described in brief:

1. Let's suppose that a sender S has encrypted a message M and sent it to the receiver R. For the R to be able to authenticate M, S has encrypted the key K with some encryption algorithm E and its outcome is concealed in the ciphertext C. The C is transmitted to R, but because the transmission channel is insecure, R will receive the ciphertext C'. R will apply the decryption algorithm D to C'. There are two possible outcomes from the decryption: (a) the message M' which is the original M or (b) an indication $\perp$, that indicates that C' wasn't authentic. This approach of message authentication is based on an encryption process. This encryption is not obligated to protect the privacy of the message. Sometimes the term *authenticated encryption* is used for this method.

2. Privacy is not necessary for a message to be authenticated. So, for this method a ciphertext C is transmitted from the sender to the receiver and it includes the message M with a tag T. T is generated by a tag-generation algorithm TG using a key K and $M, T \leftarrow TG_K(M)$. TG can be probabilistic or stateful. When the ciphertext is received, M' and T' are run in a tag-verification algorithm with the key K and a bit B is produced, $B \leftarrow TV_K\ (M', T')$. If the bit is 1, then M' is accepted as the original message, but if the bit is 0, then M' is rejected. Due to the form of the ciphertext, this mechanism is called message-authentication scheme.

3. The most common approach is when the tag-generation algorithm TG is stateless and deterministic. In this case, the whole scheme is called message authentication code, or just MAC. The process starts with the sender computing the tag – that is $T = MAC_K(M)$. It is important to mention that when MAC is used, there is no need for a tag-verification algorithm, because the receiver after the receipt of ciphertext C', which contains M' and T', can compute $T'' = MAC_K(M')$. If T'' equals T', then M' is authentic; if not, the message is disregarded.

One other way to classify AE schemes is the order in which the processes of encryption and authentication are performed. These two processes are very important for these primitives, so a different order can cause massive changes. There are three compositions which are analysed in the literature: Encrypt-and-MAC (*E&M*), Encrypt-then-MAC (*EtM*), MAC-then-Encrypt (*MtE*). These three compositions were named and studied first by Bellare and Namprempre [16] and because of their cryptographic value are also studied (sometimes with different names) in later studies [23]. The following paragraph describes in brief how each composition works and the security it offers.

In the Encrypt-and-MAC scheme, the plaintext and the key are the inputs to the encryption and MAC algorithms. The MAC tag and the ciphertext are the outputs of the two algorithms and they are sent together to the receiver. This scheme lacks in security compared to the other two [23]. The Encrypt-then-MAC scheme computes firstly the ciphertext which is used as an input in the MAC algorithm. Then the MAC is attached to the ciphertext and they are sent together. Between the three compositions, this is the one that can reach the highest level of security. The reverse procedure happens in the MAC-then-Encrypt, where the MAC is produced first and then the MAC tag is attached to the plaintext and they are together used as input in the encryption algorithm. The ciphertext is sent alone in this case. For this approach, it is proven that the security is not guaranteed [23].

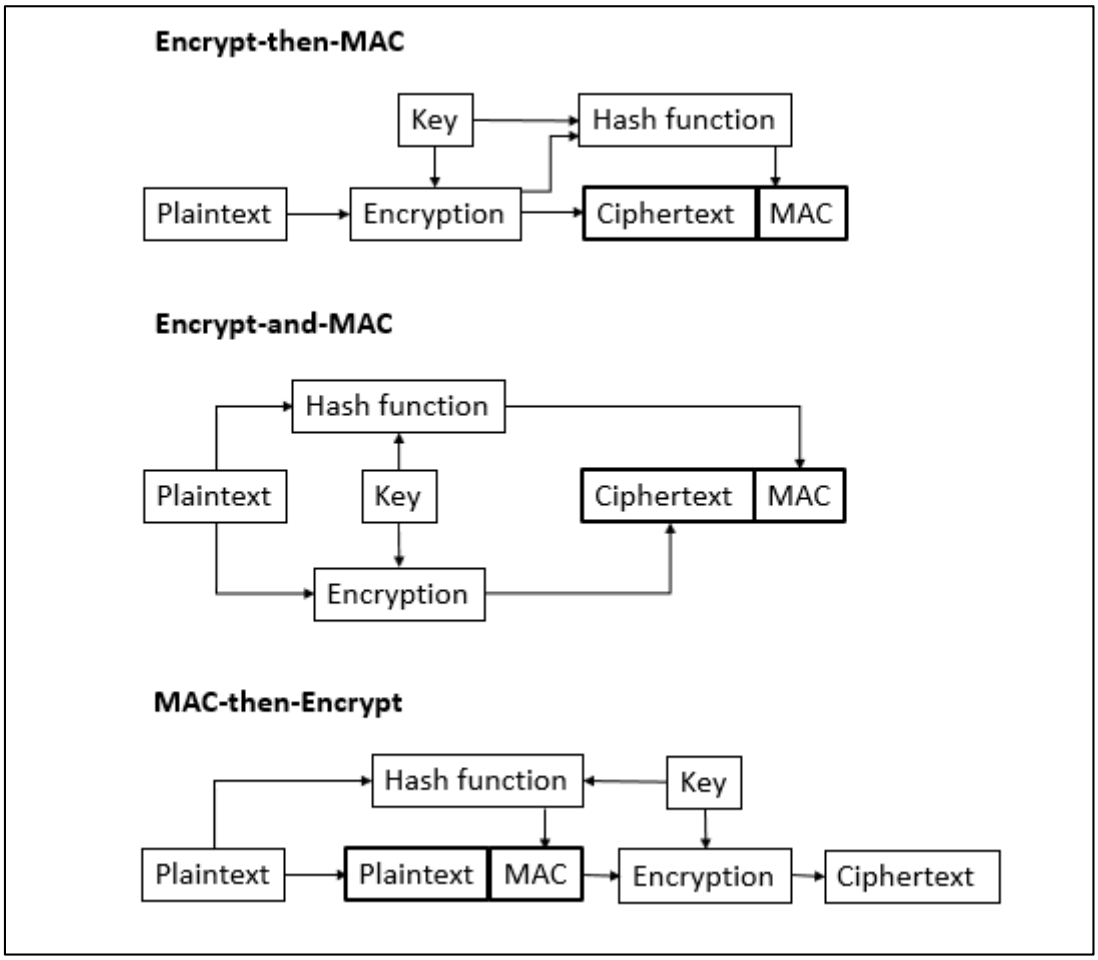

Figure 2.2: EtM, E&M, MtE from Top to Bottom Respectively

2.2.3 MAC

Message Authentication Code (MAC) has a major role in the cryptographic world. It is also referred to as a *cryptographic checksum* or a *keyed hash function*. MAC is similar to digital signature because

both of them provide message authentication and message integrity. The difference lies to the use of symmetric-key schemes from MACs and asymmetric-key schemes from digital signatures. It is also important to mention that MACs don't provide non-repudiation, unlike digital signatures (since, in MACs, there are two parties that can generate the same "signature", that is the same MAC; namely, two parties sharing the same secret key). The advantage of MACs is their speed because they are based on hash functions or block ciphers. The input of a MAC algorithm is a message of arbitrary length and it produces a fixed-size authentication tag that is cryptographically secure and it is attached to the message. A secret key is necessary between the two legitimate parties, since MACs are based on symmetric schemes. For a MAC to be considered as unforgeable and thus secure, it shouldn't be computationally possible to compute the generated MAC of a message without knowing the key.

An option for building MACs, as it was mentioned in the previous paragraph, is with the use of hash functions as building blocks. There are two approaches to the construction of hash-based MAC, the *secret prefix* MAC and the *secure prefix* MAC. Both of them generate strong cryptographic checksums, but have weaknesses [3]. A very popular among all the types of constructions of hash-based MAC is the HMAC [3] that was introduced in 1996, because it doesn't show the weaknesses of both approaches. Except that being provable secure, HMAC is also efficient and that's way is being used in the Transport Layer Security (TLS) and the IPsec protocol suite. In 1999 UMAC was proposed [26] and the designers aimed to achieve extreme speed and guarantee security. UMAC can achieve them because it uses NH, which is a universal hash-function family. These are the prominent MACs of this kind, but there are even more new similar MACs in the literature.

2.2.4 Block Ciphers

It is worth mentioning that a huge step into authenticated encryption was first made with the use of block ciphers to build MACs. Jutla presented in 2001 the Integrity Aware CBC which does not require two passes to provide authenticated encryption [24]. Instead, this mode of operation requires $m + 2$ block encryptions, assuming that the message has a length of m blocks. In the same study, a highly parallelizable mode (IAPM) for encryption and message integrity is introduced. IAPM was used as a base scheme for the construction of OCB [25]. The latter was a refined scheme of the former and has additional features. Other important block cipher modes that provide authenticated encryption is the Counter with CBC-MAC mode [27] and Galois /Counter mode (GCM) [28]. These ciphers paved the way for the following block and stream ciphers offering AE.

2.2.5 AEAD and DAE(AD)

Studies of the first AE schemes shown that there are some data that don't need to be encrypted. So, it was decided that for better results in efficiency a mixture of encrypted and unencrypted data would be optimum. This of course is not supposed to decrease the security of the cryptosystem. For this new scheme, privacy is necessary for the encrypted data and authenticity for all the data. This scheme is called authenticated-encryption with associated-data (AEAD) [29]. The unencrypted data are called associated data. It is very useful in the case of network packets because their header doesn't have to be encrypted, in contrast to the payload. Both of them though need authentication. Later works [30]–[32] describe methods to provide AE and simultaneously authenticate the associated data.

A special kind of AE schemes is the Deterministic Authentication Encryption (DAE) [33]. The main difference with the other schemes is that DAE doesn't use a nonce. DAE can be used with associated data (DAEAD), thus there is also a header used in the cryptosystem. Security of these schemes is determined as for other AE schemes. In other AE schemes the same nonce can't be used twice, but in the case of DAE schemes the same messages can't be used twice. This is a drawback for DAE schemes because they lack in efficiency because of that.

2.3 Stream Ciphers providing AE

Authenticated stream cipher is a stream cipher algorithm that can provide both authenticity and confidentiality. This kind of primitives have been designed and developed relatively recently. This means that their security is not absolutely guaranteed because not enough time has passed for getting scrutiny from the research community. That's why there are not enough studies in the literature related to the security of authenticated stream ciphers. Of course, these primitives are most probably checked with known attacks like algebraic and are able to withstand them. However, because this is a new field, maybe there are some other attacks that can be applied only to authenticated stream ciphers and specifically to those primitives. In the following years, some primitives will be proved to be secure and some won't.

The construction of an authenticated stream cipher is a mixture of the constructions of stream cipher and authenticated encryption. As in stream ciphers, a secret key k is shared to the legitimate parties and most times it is combined with an initialization vector (IV). At first, a plaintext P is

inserted into the authenticated stream cipher algorithm and the output is the ciphertext and a MAC tag (MT). The way the MT is going to be combined with the ciphertext depends on the order in which the MAC and encryption process are being done like it is described in subchapter *2.2.2*. The encrypted message EM is then transmitted over the insecure channel from the sender to the receiver. The latter gets the encrypted message EM' which is the input data to the authenticated stream algorithm that computes the decrypted message P' with the corresponding MAC tag MT'. If the values of MT and MT' are the same then the integrity of the message is validated and the message is show, otherwise the message is considered as falsified and is disregarded.

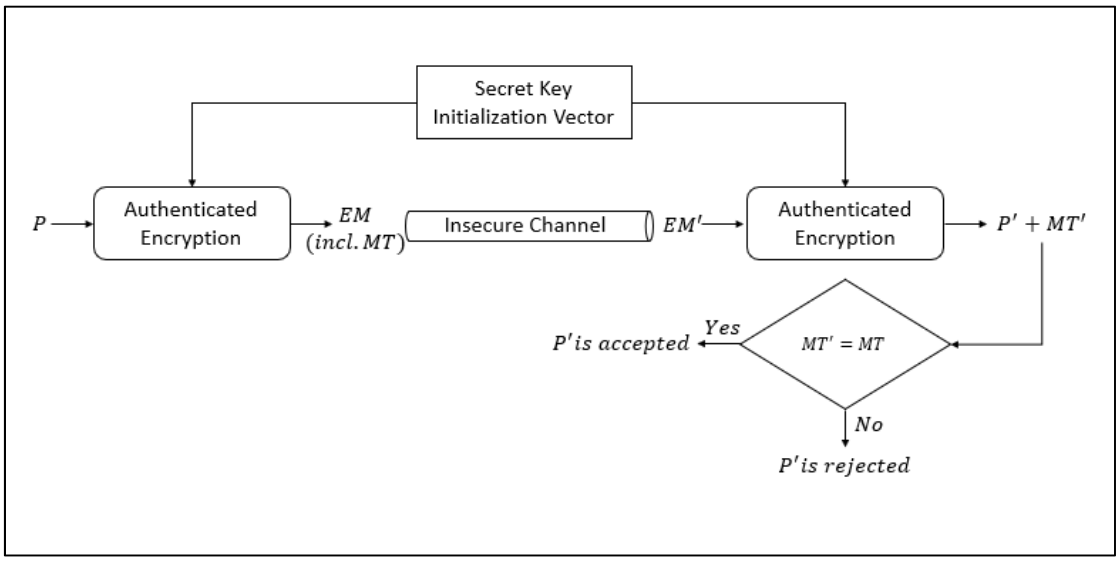

Figure 2.3: Authenticated Stream Cipher

The stream ciphers providing AE attract the attention of the cryptographers and the designers. That's why there are many submitted designs of constructions of such algorithms. Some examples of such ciphers are SOBER-128[34], Helix [35] and Phelix [36], ZUC [37], Grain-128a [38]. ZUC is the only one (of the mentioned ciphers), that was designed for telephony application. It was considered to be secure until it was found susceptible to timing attacks [39]. SOBER-128 is one of the SOBER family of ciphers and it was designed with a built-in MAC. Many attacks were tried on this cipher, but the one that's staying on top of them and it indicates a weakness of SOBER-128 is the MAC forgery attack [40] .Phelix is an advanced version of the Helix cipher. Both of them were found vulnerable to differential attacks – Helix to a differential attack published by Muller [41] and Phelix to a differential attack if the nonces are reused published by Wu and Preneel [42]. In contrast, Grain-128a, which is of the Grain family of ciphers, is showing good results against known attacks and it will be used later in this study.

Authenticated stream ciphers have some similarities in their construction except from the standard parts of a stream cipher. They usually depend on LFSRs that interact with NFSRs or non-linear functions to generate the keystream and the MAC tag. It's also very common to use an Initialization Vector (*IV*) and associate data to strengthen the security. In general, authenticated stream ciphers use techniques that don't require much resources because except from encryption this cipher is also designed to authenticate the message too.

Apart from the stream ciphers that were built from the start to provide authentication, there are the cases where a stream cipher can be reformed to provide authentication. This can happen by combining the stream cipher with a MAC like it was previously described. A common MAC that is combined with stream ciphers is *Poly1305* provided by Bernstein [43]. The problem is that there is not enough research on the optimum way to combine a stream cipher with a MAC algorithm. Another way of making stream ciphers provide authentication is by using an initialisation vector (*IV*) [33]. In general, the study describes «*a systematic framework for using a stream cipher supporting an initialisation vector (IV) to perform various tasks of authentication and authenticated encryption*». The big advantage of the constructions based on this study is a keyed hashed function that it was proven to have low collision and differential probabilities. Both methods are widely used to stream ciphers that have proved their security. Of course, security must be preserved and after the reform of the cipher.

2.4 Caesar Competition

The significance of the authenticated stream ciphers can be seen in the ongoing competition CAESAR (Competition for Authenticated Encryption: Security, Applicability and Robustness)[1]. The organisers of the competition are asking cryptographic designers to submit shared secret-key authenticated ciphers that have better features over AES-GCM and are suitable for widespread application. The specific requirements of the ciphers are provided in the competition's website[2].

At the first round, there were 57 ciphers that were put to test for security and efficiency. As a result, the ciphers that were not qualified to the standards of the competition were withdrawn. This happened at each round with more requirements as the number of the rounds was increasing. At present, the competition is at round 4 with 7 ciphers. The three main types of ciphers that were

[1] https://competitions.cr.yp.to/caesar.html
[2] https://competitions.cr.yp.to/caesar-call.html

submitted are block, stream and sponge. Of the 7 finalists two are authenticated stream ciphers, ACORN [44] and MORUS [45]. Both of them showed good results in all the known attacks and efficient tests. A bit more interesting is the ACORN cipher, that will be used later in the study.

Chapter 3
Approximation technique

The approximation techniques can used in combination with other cryptanalytic techniques to compromise a cipher. The approximations are until now useful in attacks against combination and filter functions. However, the approximation technique that is described in this Chapter is also applied to feedback functions in Chapter 5. This is done for two reasons: 1) Evaluation of the strengthness of the approximation technique 2) for future use in case a cryptanalytic technique is developed that can use approximations of feedback functions.

In the beginning of this chapter the Error Linear Complexity Spectrum (*ELCS*) is explained and analysed. Later on, there is a detailed analysis on the Boolean functions and after that, the relationship between binary sequences and Boolean functions is explained. The latter leads to the conclusion that the ELCS can be useful in finding approximations of Boolean functions. The method of finding approximations is analysed and it is also mentioned how it is possible to achieve the best possible approximations in some cases.

3.1 Error Linear Complexity Spectrum

Stream ciphers produce keystreams that are combined with the plaintext to create the ciphertext. These produced keystreams are binary sequences that must have good pseudorandomness properties like complexity [6], [46]. The most common way to measure the complexity of a *sequence s* with *period N* is via its *linear complexity* $c(s)$, which is the length of the shortest *LFSR* that is able to generate s. A strong-secure sequence must have high $c(s)$. Nevertheless, the linear complexity is not the only cryptographic criterion that is being used to evaluate the pseudorandomness properties of a sequence.

The notion of pseudo-random sequences [4] refers to a periodic binary sequence that satisfies three randomness postulates. These postulates, that are proposed by Golomb, correspond to the properties that need to be satisfied by a sequence to resemble a random one. However, despite their importance, these Golomb's pseudorandomness criteria are not sufficient and thus, a cryptosystem is not secure only by applying these postulates to create a cryptographic sequence. For example, the linear complexity of a sequence is not being explicitly described by Golomb but it constitutes an important cryptographic criterion since $2c(s)$ of consecutive bits are needed by an attacker to fully determine the whole sequence. This can happen by using the Berlekamp – Massey algorithm that takes advantage of the low linear complexity of the sequence. The algorithm needs up to $\mathcal{O}(N^2)$ operations to be completed, for N being the known part of the sequence. This algorithm computes the linear complexity but also the feedback polynomial of the shortest LFSR that generates the sequence; this minimum-length LFSR is unique if and only if the linear complexity of the sequence is less than the half of the length of the whole sequence (that's why knowledge of $2c(s)$ consecutive bits is adequate).

In the case that period N equals 2^n, then there is a more efficient way to compute the linear complexity of a sequence. The Games-Chan algorithm (GCA) [47] can compute $c(s)$ using $\mathcal{O}(N)$ operations but it has the drawback that it needs the whole period of the sequence (whilst the Berlekamp-Massey requires $2c(s)$ bits). Thus, this algorithm is not applicable to modern ciphers that have large periods, but it reveals properties that can be used to construct sequences with specific properties [48], [49].

In short, the GCA works by performing the following steps:

Let $c(s) = 0$. At first, the sequence s that has length l, is decomposed in $L = (s_1, \dots s_{l/2})$ and $R = (s_{\frac{l}{2}+1}, \dots, s_l)$, which are the left are right halves of the sequence respectively. If the two halves are not identical ($L \oplus R \neq 0$), then $c(s)' = c(s) + l/2$ and $s' = L \oplus R$. Otherwise, $c(s)' = c(s)$

and $s' = L$. After that in both cases $l' = l/2$ and the process starts again from the point of decomposition. When $l = 1, c(s)' = c(s) + 1$ if $s = 1$ and $c(s)' = c(s)$ if $s = 0$. The value of $c(s)$ is equal to the linear complexity of s.

Linear complexity is valuable as a measure for the randomness of finite sequences. Rueppel noticed this and introduced *linear complexity profile* [50], which describes the growth of linear complexity as the length of the sequence increases. Moreover, criteria to evaluate the randomness of generated sequences are analysed. This profile is an important tool for the assessment of finite binary sequences such as the keystreams of stream ciphers.

It was previously stated that sequences must have high linear complexity for security. Except that, the sequence also needs to keep its linear complexity at a high level even if some of bits are changed to be cryptographically strong. If this doesn't hold, then the knowledge of some consecutive bits of a keystream can lead to the creation of a sequence that closely approximates the original. This observation is of high cryptographic value and led to the introduction of the *k-error linear complexity* of sequence [51]. As the definition implicates, it is related with the variation of linear complexity depending on the number of changed bits. Specifically, as it is stated in the last mentioned paper, the k-error linear complexity (denoted as $c_k(s)$) of a sequence s that has period N and linear complexity $c(s)$, is defined as the lowest possible $c(s)$ of s when k or fewer bits are changed in every period of the sequence.

The Error Linear Complexity Spectrum (ELCS) is based on the k-error linear complexity and it indicates how linear complexity decreases as the number of the changed bits increases. This means that when we have 0 errors the linear complexity of s is $c(s)$ and when k equals $wt(s)$ then $c(s)$' equals 0. Besides these 2 points, any pair $(k, c_k(s))$ for any k lies in the aforementioned spectrum. ELCS is defined as k-error linear complexity profile by Martin and Stamp [51]. It is highlighted that the same definition was also given by Niederreiter [52] with the difference on the way the linear complexity changes. Etzion [53] presented a formula of the minimum k that is needed to reduce the $c(s)$ of sequences with $N = 2^n$ and is shown as an explicit function of the Hamming weight of $c(s)$. There are also several other properties of the ELCS that are studied and proved in other works [54].

Based on the Games-Chan algorithm, Martin and Stamp managed to create an efficient algorithm that is able to compute, for any fixed k, the linear complexity $c(s)$ for a binary periodic sequence of period $N = 2^n$. The algorithm [51] computes the entire ELCS using $\mathcal{O}(N^2 logN)$ operations.

Subsequently, Lauder and Paterson [55] created a generalized version of the algorithm to compute the entire ELCS of sequences. This algorithm can also be used as a soft-decoding method for a specific class of linear subcodes of Reed – Muller binary codes [56].

The Lauder and Paterson algorithm (LPA) takes as input a sequence s of period $N = 2^n$ and the output presents the points where there is a decrease of the linear complexity along with the number of bits that need to be changed for that decrease. These points are called *critical points* (CPs), the number of bits is denoted by k and the linear complexity of the sequence with k changed bits is denoted by $c_k(s)$. Clearly, the critical points constitute a subset of ELCS and is being called Critical Error Linear Complexity Spectrum (CELCS). In the output, the critical points are presented as $CP: (k, c_k(s))$. All sequences have at least 2 CPs, which are the points $(0, c(s))$ and $(wt(s), 0)$. The values of the in between CPs (if there are any) depend on each sequence. For any given k, a *critical error sequence*, which is denoted by e, is a sequence with period N and weight k for which $c(s \oplus e) = c_k(s)$.

As stated above, given a binary sequence s of period $N = 2^n$, the Games-Chan algorithm requires the full sequence to compute its linear complexity, while Berlekamp-Massey algorithm requires only $2c(s)$ of bits. A modified version of the Games-Chan algorithm [57], was designed that needs only $2c(s)$ of bits to compute the complexity. In the same article, the Lauder-Paterson algorithm is also modified, so for a given constant c, it computes the minimum number of errors and their position needed for bringing the complexity below c over a period.

3.2 Boolean Functions

Boolean functions are one of the most important tools in cryptography. They are used in various ways and have the most times a prominent role in the efficiency and the security of cryptosystems. Their most important applications rest with their usage as building blocks in symmetric cryptosystems. Specifically, they are used in the analysis and design of s-boxes in block ciphers and for the construction of filter or combining functions in stream ciphers [6]. The widespread use of Boolean functions is mainly attributed to their simplicity in hardware and software application and because when they are properly used in cryptosystems, they can withstand various known cryptanalytic attacks.

3.2.1 Nonlinearity

It was mentioned in the previous chapter, that just using an LFSR with good statistical properties and high linear complexity as a pseudorandom bit generator, is not adequate in terms of security because of the Berlekamp-Massey algorithm (even if using an LFSR of huge size with adequately high linear complexity that can resist to Berlekamp-Massey, the system that would use such an LFSR would be impractical). A solution to this problem is to use a nonlinear Boolean function in the process of generating the sequence. The nonlinearity of a function f can prevent linear cryptanalysis attacks [58] and best affine approximation attacks [59]. Thus, a Boolean function of high nonlinearity must be used for the generation of the sequence or a system that can increase nonlinearity at a sufficient level.

Such systems are the nonlinear combination generators and the nonlinear filter generators. The former takes the outputs of n LFSRs and use them as inputs in an n-vector nonlinear Boolean function $f(x_1, x_2 \ldots . x_n)$. The function f then generates a sequence of bits that serves as the keystream. For the latter, a nonlinear function f is required to be used at a fixed number of stages of the outputs of a single LFSR. Both of them are widely used in modern stream ciphers. More on these generators can be found in the literature [60].

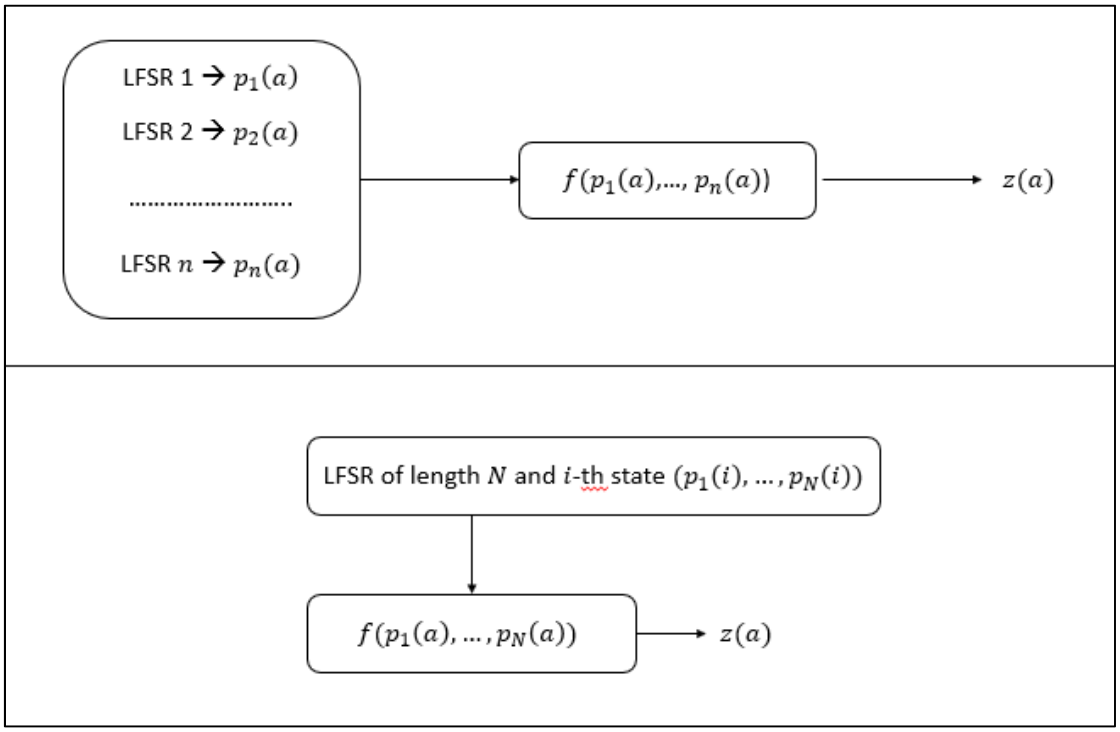

Figure 3.1: Nonlinear Combination Generator (Top) and Nonlinear Filter Generator (Bottom)

3.2.2 Properties

Let $F_2 = \{0,1\}$ and B_n be the set of all Boolean functions that consist of n variables. From the previous assumptions, it is considered that $f \in B_n$ and $f: F_2^n \rightarrow F_2$. The most common way to express f is by a multivariate polynomial, called *algebraic normal form* (ANF) and it's given by:

$$f(x_1, \dots, x_n) = \sum_{i \in F_2^n} a_i x_1^{i_1} \dots x_n^{i_n} \quad a_i \in F_2$$

where sum (Σ) is performed modulo 2. Each monomial of a polynomial is composed by a number of variables.

The *algebraic degree* of f, denoted by $deg(f)$, is the maximum number of variables that are presented in one of the monomials with a nonzero coefficient. In case that $\deg(f) = 1$, then the function f is *affine*. Furthermore, if the constant term in the ANF is zero, the f is *linear*. The value of the degree affects the weight of f [61]. If the Boolean functions have only terms with the same degree, then they are called *homogeneous*. The opposite function or complement of function f is f'; thus,$f' = f \oplus 1$. It is important to mention that cryptosystems shouldn't use Boolean functions with maximal degree because their output distribution is biased.

Another important representation of a Boolean function is its *truth table*, which present the values of each element in F_2^n and the value vector of f. For example, Table 3.1 illustrates the truth table of the Boolean function $f = x_1 + x_2x_3$. A Boolean function can be identified and it is defined by its value vector. McWilliams and Sloane [62] describe a method to build the ANF of a Boolean function f by its truth table.

x_1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1
$x_1 + x_2x_3$	0	1	0	1	0	1	1	0

Table 3.1: N Truth table of $f = x_1 + x_2x_3$

The hamming weight of a function f, denoted by $wt(f)$, is the number of $1's$ in the truth table of f. If $wt(f) = 2^{n-1}$, then the f is called *balanced*.

Balanced functions are used by most cryptosystems because they offer better randomness compared to unbalanced functions. The latter have an unbalanced distribution of binary digits or a

statistical bias as it's called and they are subject to various cryptanalysis attacks like correlation attacks [63]. However, almost balanced functions may also be acceptable, if they simultaneously satisfy other cryptographic criteria.

Hamming distance is the number of differences in the truth tables of two functions. It is defined as:

$$d(f,g) = wt(f \oplus g)$$

regarding that $f, g : Vn \rightarrow F2$. Thus, the hamming distance of a function f to all affine/linear functions is its nonlinearity.

Let $f, g \in B_n$, A be a non-singular matrix $(n \times n)$ over F_2 and b an n-vector over F_2 . It is said that $f(x)$ and $g(x)$ functions in n variables are *affine equivalent* if $g(x) = f(Ax \oplus b)$. When two Boolean functions are affine equivalent, then $wt(f) = wt(g)$ and $NL_f = NL_g$, The weight (wt) and nonlinearity (NL) are affine invariant cryptographic properties of Boolean functions.

The Walsh transform (also called the Hadamard transform) is of great importance for Boolean functions and therefore for cryptography. It is a generalized class of Fourier transforms and it performs a symmetric, orthogonal, involutive, linear operation on 2^n real numbers. By using the Walsh transform the nonlinearity of a function f can be easily computed [64], like many other properties of Boolean functions.

The *bias* or *correlation* or *imbalance* of a Boolean function f is denoted by Ɛ is and can be calculated by:

$$\mathcal{E}(f) = \sum_{x \in F_2^n} (-1)^{f(x)} = 2^n - 2wt(f)$$

It is understood that a function f is balanced only if $\mathcal{E}(f) = 0$.

The Walsh spectrum of $f : F_2^n \rightarrow F_2$ is computed by:

$$\mathcal{F}(f + \phi_a) = \sum_{x \in F_2^n} (-1)^{f(x)+a*x}, a \in F_2^n$$

where ϕ_a is the linear function $x \mapsto \alpha * x$ ($*$ indicates the inner product). Thus, the hamming distance of f to $\{\phi_a + \varepsilon,\ \alpha \in F_2^n, \varepsilon \in F_2\}$, which is the nonlinearity of f can by computed by:

$$NL_f = 2^{n-1} - \frac{1}{2}\mathcal{L}(f) = 2^{n-1} - \max_a |\mathcal{F}(f + \varphi_\alpha)|$$

A useful property of Boolean function is the *Strict Avalanche Criterion* (SAC), which was first introduced by Webster and Tavares in a study dedicated to the design of S-boxes [65]. A Boolean function $f(x)$ of n variables that satisfies SAC means that changing one bit of the input x will result to the change of exactly half of the 2^{n-1} vectors to the output of the function. SAC can be used in various cryptographic applications [66]. It is widely used because of the big change to the output that occurs even in case of a slight change in the input. This way, a Boolean function input is more difficult to be computed by its output, which is essential for cryptosystems. More details on the SAC and how to construct SAC functions are described by Cusick and Stanica [67].

When the values of $f \in B_n$ are statistically independent of a subset of k variables ($1 \leq k \leq n$), then the Boolean function f is *correlation immune of order k*. Namely, if the subset of k variables is called W and $W = \{x_{i_1}, \dots, x_{i_k}\}$, then:

$$I\ (f(x)|W) = 0$$

Correlation immunity determines the minimum number of LFSRs that must be used in a correlation attack on a combination generator, which is $k + 1$ [61]. However, there is a trade-off between several desired cryptographic properties. It is indicated [68] that even with the introduction of one bit of memory into the generator, the trade-off of degree and correlation immunity can be avoided. In case that a *k-th order correlation immune* function is *balanced*, is called *k-resilient* [69].

Another important cryptographic property that a Boolean function needs to possess is that it should not be well approximated by another function with fewer number of variables. Not many results are currently known on this criterion. For t-resilient functions it is possible to compute the minimum distance of an approximation function with less variables. Let $f \in B_n$ be a t-resilient function. Then, the hamming distance $d_H(f, B_n(k))$ of f from the set $B_n(k)$ of all functions depending on k input variables satisfies [70]:

$$d_H(f, B_n(k)) \geq 2^{n-1} \frac{\mathcal{L}(f)}{2} \left(\sum_{i=t+1}^{k} \binom{k}{i}^{\frac{1}{2}} \right) \quad (1)$$

Let us consider a Boolean function $f(x_1, \dots, x_n)$ depending on n variables. An important representation of Boolean functions is the Boole's expansion or as it is also called, the Shannon's expansion, which is defined, for any input variable $x_i, i = 1, \dots, n$ as follows:

$$f(x_1, \dots, x_n) = (1 + x_i) f_0(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) + x_i f_1(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$$

where both functions f_0, f_1 depend on $n - 1$ variables (actually on all variables except x_i). More precisely, f_0 (resp. f_1) is the function determined by the initial function f if we fix the value of x_i being equal to 0 (resp. 1).

Example:

Recalling the previous example with the function $f(x_1, x_2, x_3) = x_1 + x_2 x_3$, the Shannon's Expansion Formula for each of the three variables are given as follows:

$f(x_1, x_2, x_3) = (1 + x_1)(x_2 x_3) + x_1(1 + x_2 x_3)$ (for the variable x_1)

(indeed, by setting $x_1 = 0$, f becomes $x_2 x_3$, whereas by setting $x_1 = 1, f$ becomes 1+$x_2 x_3$)

$f(x_1, x_2, x_3) = (1 + x_2)(x_1) + x_2(x_1 + x_3)$ (for the variable x_2)

$f(x_1, x_2, x_3) = (1 + x_3)(x_1) + x_3(x_1 + x_2)$ (for the variable x_3)

3.2.3 Special Types

Boolean functions are called *symmetric* if their outputs depend only on the Hamming weights of their inputs. This means that any permutation of the input bits will not result in any change of the function's value. An extensive study on Symmetric Boolean Functions and specifically their cryptographic properties is given by Canteaut and Videau [71]. Except that, there is also a detailed analysis of symmetric functions of even variables with maximum algebraic immunity [72].

A bent function is another special type of Boolean functions [73]. The feature that separates bent functions from all other is that they have maximum difference (or distance) from all linear and affine functions – i.e. they achieve the maximum possible nonlinearity. This feature is desirable in cryptography because it makes them hard to approximate. That's the reason they are widely used and studied [74]. However, bent functions have some drawbacks: i) they exist only for an even number of variables, ii) they are non-balanced.

3.3 Relationship between Binary Sequences and Boolean Functions

The relationship of binary sequences and Boolean functions can be found in many studies in the literature. It is proved [75] that there exists an $1-1$ correspondence between binary sequences of period 2^n and Boolean functions of n variables. However, a slightly different approach to the relationship [1], that leads to some desirable conclusions, is used in this thesis.

If $s = (s_1, \dots, s_{2^n})$ is a periodic binary sequence of period 2^n, there is a Boolean function f of n variables, whose truth table denoted by $s_f = (s_1, \dots, s_{2^n})$ corresponds to $s, s \leftrightarrow s_f$. Thus, for any $f \epsilon B_n$ of n variables, there is a 2^n periodic sequence s that $s \leftrightarrow s_f$. This definition makes the Boolean function $f \epsilon B_n$ subject to all cryptographic criterions that can be applied to the binary sequence s. This means that as for the sequence s, the linear complexity and the ELCS can also be calculated for s_f.

3.3.1 Computation of Approximation Functions

It is proved [1] that for a 2^n periodic binary sequence s with linear complexity $c(s)$ that $2^{n-l-1} < c(s) \leq 2^{n-l}$ for some $1 \leq l < n-1$, if the Boolean function $f(x_1, \dots, x_n)$ depends only on $x_1, \dots, x_{n-l}$. This proposition establishes a relationship between the linear complexity of sequence s with the number of variables of the corresponding Boolean function f. Thus, if the $c(s)$ is decreased enough, the number of variables of the corresponding function is decreased too. This leads to the conclusion that by reducing $c(s)$, we can compute approximation functions that depends on fewer number of variables.

Approximating a Boolean function f with another that depends on fewer variables falls into the category of *correlation attacks* and can also be considered as divide-and-conquer algorithm. The attack that is presented by Siegenthaler [63] can be prevented by using correlation-immune functions [69]. It is highlighted that correlation immunity is an affine invariant criterion. Canteaut [70] describes how an approximation function can be used for cryptanalysis.

The LPA can be useful in computing approximations of functions with fewer variables. This is because LPA like it was previously described, finds the ELCS of a sequence, which includes the points where the $c(s)$ is decreased. This approach is described in the following paragraph:

Let s be a binary sequence of period 2^n with a CP $(k, c_k(s))$ satisfying $2^{n-l-1} < c(s) \leq 2^{n-l}$ for $l \geq 1$ and f be a Boolean function for which $s_f \leftrightarrow s$. It is proved [1, Theorem 2] that the function h that can be built by the sequence $s_f \oplus e$ depends on the first $n - l$ variables and there are no other sequences for which $wt < k$ and can lead to the creation of functions that depend on the first $n - l - 1$ variables. It is also stated that if k is the least possible of all the values of k in CPs in the spectrum of $2^x < c(s) \leq 2^{x+1}$ then there are no other sequences whose weight wt satisfies $wt < k$ and can lead to the creation of functions that depend on the first $n - l$ variables.

As it was previously mentioned, the LPA always finds two CPs for all sequences. However, only the in between CPs have cryptographic value for the purpose of finding approximations of functions and from these (in between CPs) only the ones with the least k for a certain number of variables are used. Thus, we will call these CPs, *Significant Critical Points* (SCPs). It should be stated that the algorithm doesn't present in the output the appropriate changes that need to be made to the sequence for each CP. The algorithm can be appropriately modified to also compute the critical error vectors [55]. This modification is needed in the process of building the approximation function. It is also important to mention that the LPA doesn't compute the best approximations of functions with functions of fewer variables, but at least it sets the upper bound limits [1].

It seems preferable for sequences to have only two CPs, so that they don't have any approximations. However, this kind of sequences lack of cryptographic strength [54]. A brief description of the weakness follows:

Let the truth table of function f be a sequence that have only two CPs be represented as $s_f = [L\ R]$ with L representing the left half of s_f and R the right half. These sequences have one of the two following form and it is explained beneath of each form why they lack of cryptographic strength:

i) $L = R = \hat{s}\ \ OR\ \ L \oplus R = 0$

In case the (i) implies, then the truth table of f is of the form $s_f = \hat{s}||\hat{s}$ (where || indicates concatenation), which means that the s_f of function $f(x_1, \dots, x_n)$ doesn't depend on the last variable x_n. Immediately, since the s_f depends only on $x_1, \dots, x_{n-1}, x_n$ is not taken into account. Consequently, we can calculate approximations of the function using the sequence $\hat{s}$, which uses one less variable that the sequence s_f.

ii) $L = R \oplus 1 = \hat{s}\ \ OR\ \ L \oplus R = \mathbf{1}$

In case that (ii) implies, it means that the truth table of f is of the form $s_f = \hat{s}||\hat{s}\oplus 1$ (where || indicates concatenation). It is well known that the value of the last variable x_n is 0 in the first half of the truth table and 1 in the second half of the table. Thus, this kind of sequences can be produced by a Boolean function whose ANF includes the variable x_n in a monomial of $\deg(f) = 1$. This leads to the conclusion that:

$$s_f(x_1, \dots x_n) = \hat{s}(x_1, \dots, x_{n-1})||\hat{s}(x_1, \dots, x_{n-1})\oplus x_n = \hat{s}(x_1, \dots, x_{n-1})||\hat{s}(x_1, \dots, x_{n-1})\oplus 1$$

The previous shows that again, the variable x_n is not taken into account.

Theorem 3 [1] is an extension of Theorem 2 and it includes all the permutations matrices of Boolean function f over F_2, which is denoted by P_n. A Boolean function have $n!$ possible matrices and they should be used to find approximations with fewer variables. This means that better approximations can be found by using the LPA. It is also stated that Theorem 3 is applicable to functions that are *affine equivalent* to f and not only to the functions that are obtained by permuting the input variables.

It is proved [1] that for a Boolean function f of $\deg(f)$, the LPA can be used to efficiently compute approximations of degree lower than $\deg(f)$. As it was previously mentioned, the LPA computes the ELCS of a sequence and it can also present the corresponding critical error sequences. This was used to create an algorithm in the same paper for the computation of lower degree approximations of functions. Although, it is not ensured that the algorithm always computes the best low degree approximations, experiments indicate that in some cases, such approximations are indeed efficiently computed.

3.3.2 Analysis of the Computation Method

As it was stated, the LPA can compute CPs that can lead to approximations of function $f(x_1, \dots, x_n)$ with fewer variables. The approximation functions depend on $n - x$ variables for $1 \leq x \leq n - 2$. The way the variables are reduced is from the latter (x_n) to the first (x_1).

Let $s = [L\ R]$ be a sequence of period 2^n with L representing the left half of s and R the right half. A simple way of explaining how the LPA works is that it compares L and R and it finds the bits that need to be changed in one of them, to make them identical. If the approximation function depends on $n - 1$ variables, then a number of bits must be changed so that $s = [\dot{s}\ \dot{s}]$. If the approximation depends on $n - 2$ variables, then $s = [\dot{s}\ \dot{s}\ \dot{s}\ \dot{s}]$ and so on.

Based on the properties of the LPA, a way to compute higher approximations through proper permutation is examined. For this, the notion of $\mathcal{BA}$ is introduced. The bits affected in the truth table by a variable is denoted by $\mathcal{BA}$. The $\mathcal{BA}$ of a variable x depends on the positions of x in the polynomial. This means that the monomials that include x should be taken into account to measure $\mathcal{BA}$. The basic factors that determine $\mathcal{BA}$ are 1) the degree of the monomials and 2) the other variables that are included in each monomial.

More precisely, the $\mathcal{BA}$ is given by the following:

Theorem 1

Let f_0, f_1 be the sub-functions (depending on $n - 1$ variables) obtained by applying the Shannon Expansion Formula to f with respect to the variable x_i for any i. Then the $\mathcal{BA}$ for x_i is equal to the weight of the function $f_0 + f_1$ (being considered as functions on $n - 1$ variables).

Proof:

Since f_0 equals to f under the assumption that $x_i = 0$ and f_1 equals to f under the assumption that $x_i = 1$, we get that x_i affects an output of f if and only if, $f_0(v) \neq f_1(v)$, where v is the corresponding input vector on $n - 1$ variables (all except x_i). Hence, the claim follows.

Example:

Let f be a Boolean function for which the ANF is:

$$f(x_1, \dots, x_5) = x_1 + x_1x_2 + x_3x_4x_5$$

The s_f of the above function is: 01000100010001000100010001001011

1st half: 0100010001000100

2nd half: 0100010001001011

Recalling the properties of Boolean functions, the difference between the two halves is made by the variable x_5, which means that $\mathcal{BA}_{x_5} = 4$.

$\mathcal{BA}_{x_5}$ can be calculated using the Shannon expansion formula:

$$f(x_1, \dots, x_5) = x_5(x_1 + x_1x_2 + x_3x_4) + (1 + x_5)(x_1 + x_1x_2)$$

The $\mathcal{BA}_{x_5}$ equals the $wt()$ of the sum of $f_0(x)\ and\ f_1(x)$:

$$\mathcal{BA}_{x_5} = wt(f_0(x) + f_1(x))$$

$$\mathcal{BA}_{x_5} = wt((x_1 + x_1x_2 + x_3x_4) + (x_1 + x_1x_2))$$

$$\mathcal{BA}_{x_5} = wt(x_3x_4)$$

$$\mathcal{BA}_{x_5} = 4$$

It is also understood that if variables x_3, x_4 were last, they would also affect 4 of the 32 bits in contrast with the other two that affect more bits. The variables have different $\mathcal{BA}$ values because they exist in monomials of different degrees.

Corollary 1

Let f be a Boolean function depending on n variables $x_1, x_2, \dots x_n$. Then the $\mathcal{BA}$ for x_i for any i, is equal to the weight of f_{x_i}, where f_{x_i} denotes all the monomials in the ANF of f that contain x_i.

Proof:

The proof is straightforward from Theorem 1, since – with the notation therein - f_{x_i} is equal to $x_i * (f_0 + f_1)$.

The $\mathcal{BA}$ of a variable x_i has a crucial role in determining how well a function can be approximated by a function with the same number of variables minus x_i (i.e. with one less variable). Indeed, the $\mathcal{BA}$ provides direct information on the existence of a function with $n-1$ variables whose hamming distance from the initial function is equal to $\mathcal{BA}$; this approximation is the function that is obtained by f by simply removing all the monomials in the ANF that depend on the variable x_i (the weight of these monomials equals $\mathcal{BA}$). Consequently, the smallest value of $\mathcal{BA}$ amongst all the possible variables constitutes the best choice to compute the highest possible approximation for a function that depends on $n-1$ variables.

The $\mathcal{BA}$ can be trivially obtained for some cases. For example, we prove the following result.

Proposition 1

If a variable x_i lies only in one monomial in the ANF of f, with degree $deg(monomial)$, then the $\mathcal{BA}$ is $2^{n-deg(monomial)}$; therefore, there exists an approximation on $n-1$ variables that approximates f $\left(1-\frac{1}{2^{\deg(monomial)}}\right)\times 100\%$. This implies that the highest possible approximation that can be calculate by LPA for a function that depends on $n-1$ variables equals $\left(1-\frac{1}{2^{\deg(f)}}\right)\times 100\%$; this is the upper bound limit of approximations that are calculated by LPA. Moreover, if all the variables in this monomial are also present only once in the ANF of f, that is they do not appear to any other monomial, then again there exists an approximation depending on $n-deg(monomial)$ variables that approximates $f\left(1-\frac{1}{2^{\deg(monomial)}}\right)\times 100\%$.

Proof:

Amongst the 2^n rows in the truth table of f, there exist $2^{n-deg(monomial)}$ rows in which all the variables in the monomial have the value "1"; recalling the definition of $\mathcal{BA}$ as well as Corollary 1, the first claim follows. Clearly, if we remove this monomial from the ANF of f, we get another function on $n-deg(monomial)$ variables whose Hamming weight from f is also $2^{\mathrm{n}-\deg(monomial)}$ and, thus, the second claim also follows.

Example:

$$f(x) = x_1 \oplus x_2x_3 \oplus x_1x_3 \oplus x_4x_5 \oplus x_1x_2x_3$$

It is observed that variables $x_4\ and x_5$, which are the last variables in the function only exist once in the same monomial. Since the monomial of the last variables ($x_4\ and\ x_5$) is of degree 2 the function that depends on two less variables approximates $f(x)$ by 75%. The approximation function is $x_1 \oplus x_2x_3 \oplus x_1x_3 \oplus x_1x_2x_3$.

Therefore, if a variable appears only once, its $\mathcal{BA}$ is trivially obtained. However, the same $\mathcal{BA}$ can be also computed even if the variable x_n exists more than once in the polynomial, as is shown in the next example. In case that x_n exists only in one monomial, the LPA will yield an approximation whose distance from f is exactly $2^{\mathrm{n-deg}(monomial)}$; this is an important result which is subsequently proved.

Example:

$$f(x_1, \dots, x_5) = x_1 \oplus x_2x_3 \oplus x_1x_4 \oplus x_3x_5 \oplus x_3x_4x_5$$

In the function f the highest approximation that can be calculated by LPA for 4 variables is computable if the last variable is x_5. This is because it's in a monomial whose degree equals $\deg(f)$ and it's in another monomial with the variable x_3 that already exists in the previously mentioned monomial. It is easy to compute by using Shannon formula the $\mathcal{BA}$ and see that $\mathcal{BA}_{x_4} = 2 \times \mathcal{BA}_{x_5}$. Thus, if x_4 was used as the last variable, the approximation function will not be as good. These are verified by the LPA. Therefore, indeed we get the best result through LPA by putting as last – under a permutation of variables – the variable with the smallest $\mathcal{BA}$.

Corollary 2

If the variable x_n has a corresponding value $\mathcal{BA}$, then the LPA will yield an approximation of f with $n - 1$ variables whose distance from f is exactly $\mathcal{BA}$.

Proof:

The $\mathcal{BA}$ of x_n is given, according to Theorem 1, by the weight of $(f_0 + f_1)$ where these are the sub-functions on $n - 1$ variables obtained by applying the Shannon Expansion formula to f. Recalling how the Lauder-Paterson works, this weight is the minimum number of bits required to be changed

in order to reduce the linear complexity of the corresponding sequence below 2^{n-1} (since f_0 and f_1 correspond to the left L and right half R of the sequence respectively). Hence, the claim follows.

Note that, in some cases, the value of $\mathcal{BA}$ is trivially computed and, thus, we may know exactly which the approximation that we could get by the LPA. For example, if x_n appears only in one monomial with degree $deg(monomial)$, then its $\mathcal{BA}$ is $2^{\mathrm{n-deg}(monomial)}$. Actually, in such cases there is no need to execute the LPA to compute approximations on $n-1$ variables.

Therefore, towards computing the best possible approximations, is seems that putting – under a permutation of variables - as last the variable with the less $\mathcal{BA}$ is the right option to get the best approximation results with the LPA. Such an observation is of high practical importance, since restrict the entire space of all possible permutations on variables that can be applied is restricted.

In case that x_n exists in a monomial of $deg \geq 2$ but also in another monomial of degree 1, then it is questionable whether x_n is the best choice in order to find out the best approximations depending on $n-1$ variables. That's because the $\mathcal{BA}_{x_n}$ is equal to $2^{n-1} - 2^{n-deg}$. For instance, if we have $x_1 + x_1x_2x_3$, then x_1 affects $2^{n-1} - 2^{n-3}$ bits in the truth table and it doesn't lead to such a good approximation if x_1 is placed last, compared to the case that x_3 or x_2 are put last.

As the number of the excluding variables increases, then the relationship between the $\mathcal{BA}$ and the outputs of the LPA gets more complicated. For approximations that depends on less than $n-1$ variables things get more complicated and the interaction of the $\mathcal{BA}$ by the excluding variables should also be taken into account.

Proposition 2

When variables exist only once in the polynomial and they are in monomials of degree 1, then they don't contribute to the process of computing approximating functions and they can be ignored.

Example:

$$f(x) = x_1 \oplus x_2 \oplus x_3 \oplus x_4x_5 \oplus x_6x_7 \oplus x_4x_6x_8$$

The SCPs of $f(x)$ are (32,97), (64,25), (96,5) which correspond to the approximations 87,5%, 75%, 62,5% respectively.

The approximations of function $f(x)$ can be computed without the use of variables x_1, x_2, x_3. Thus, the remaining part of the function will be used:

$$x_4x_5 \oplus x_6x_7 \oplus x_4x_6x_8$$

The CPs for this function are (0,22), (4,13), (8,4), (12,0). A reference to all CPs is done because the last one, even if its not a SCP, is used for the approximation of $f(x)$. The CP (4,13) correspond to an approximation of 87,5%, (8,4) to a 75% approximation and (12,0) to a 62,5% approximation. The approximations are the same but the distance is different. To calculate the distance of the approximations for $f(x)$ k is multiplied by 2^x,where x is the number of variables that are ignored.

This subchapter provides us details on the limits of the approximations found by LPA, where the algorithm is not necessary and how it is possible to find better approximations through the permutation of variables.

Chapter 4
Overview of the Authenticated Stream Algorithms

In this chapter, there is a brief description of the algorithms that are used in the next Chapter to test the approximation technique that was explained in Chapter 3. Alongside that, there is also a security analysis for each authenticated stream algorithm.

4.1 ACORN

ACORN is a lightweight authenticated stream cipher that is submitted in the CAESAR competition and it's one of the leading candidates. So far, there are three versions of the cipher with some modifications-improvements in each new version. ACORN-v3 [44] is the current version and until now it has been proved to have very good cryptographic properties. For the purpose of this thesis, ACORN-128 will be used, which is suitable for lightweight and high-performance applications.

ACORN-128 uses a 128-bit key, 128-bit nonce (IV) and 128-bit tag. The associated data length and the plaintext length are less than 2^{64} bits. It has a small 293-bits state that is consisted of 6 LFSRs.

It has a sequential design and at each step, only one message bit is processed. ACORN allows parallel computation that offer high speed of hardware and software implementation.

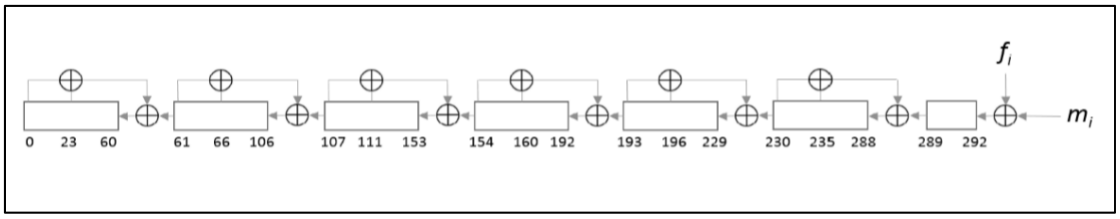

Figure 4.1: The concatenation of the 6 mentioned LFSRs. f_i is the overall feedback bit for the ith step and m_i is the message bit for the ith step.

There are three functions in ACORN-128: the function to generate keystream bit from the state, the function to compute the overall feedback bit and the function to update the state.

The keystream bit is computed by using:

$$ks_i = S_{i,12} \oplus S_{i,154} \oplus maj(S_{i,235}, S_{i,61}, S_{i,193}) \oplus ch(S_{i,230}, S_{i,111}, S_{i,66})$$

Initialization

- Key and IV are injected into the state bit by bit.
- It consists of 1792 steps.

Padding of associated data

- Padding is fixed at 256 steps.
- One bit at each step

Padding of the plaintext

- Padding is fixed at 256 steps.
- One bit at each step

Finalization

- The cipher runs for 768 steps.

- The last 128 bits of keystream are the tag.

After the processing of the associated data, one plaintext bit p_i is used to update the state and is also encrypted to c_i. The encryption is as follows: $c_i = p_i \oplus ks_i$

The decryption and verification are similar to the encryption and tag generation. If verification fails, the ciphertext and the authentication tag are not given as output to avoid known plaintext or chosen ciphertext attacks.

Changes from v1 to v3

- The number of steps was changed for initialization, padding of associated data, padding of plaintext and finalization from $1536, 512, 512, 512$ to $1792, 256, 256, 768$ respectively. This offers better protection to the key if the nonce is reused.

- In initialization, the key bits are used in 1664 steps in contrast with version 1 that were used only in 128 steps. By this modification the cipher was strengthened against nonce reuse attacks.

- The function $ch(S_{i,230}, S_{i,111}, S_{i,66})$ was moved from the feedback function f_i to the output filtering function ks_i. This was made to prevent guess-and-determine attacks.

4.1.1 Security Analysis

In general, ACORN is considered to be a secure cipher until now. Despite this, the attacks that follows present that ACORN can be compromised under some specific circumstances.

ACORN can have state collisions in the internal state [76]. State collisions happen when different sets of inputs have the same internal state at some point of cipher's operation. In order to achieve collisions in ACORN, the IV or associated data or plaintext should be modified properly.

Cube attack [77] is one powerful cryptanalytic tool designed for symmetric key ciphers and is especially effective at stream ciphers. The conventional attack considers the symmetric cryptosystem as a blackbox polynomial which is analysed experimentally. Therefore, it is not possible to evaluate the security of a cipher because the experimental size can't be exceeded. If the

cube attack is developed by the division property [78], it is possible to exploit a large cube size. This was the first time the division property was applied to stream ciphers. Compared to the previous best key-recovery attacks, this attack was more efficient and it updated the maximum number of initialization rounds that are needed for the recovery from 503 to 704.

Cube attack was applied on ACORN. This kind of attack is effective against ciphers of low algebraic degree or against ciphers of high degree but a sparse system of nonlinear equations. There are 6 LFSRs and 14 taps in the initialization process and it is expected and maybe this could make the system of equations dense. A cube attack was performed [79] to a reduced version of the initialization phase that consists of 477 steps. The attack could successfully recover the key in the reduced version but not in the normal version.

An attack framework based upon cube testers and d-monomial tests is proposed against ACORN-v3 [80]. Specifically, some high degree monomials are considered in the Boolean functions of the keystreams using cube testers and a statistical test is carried out on the outputs of the truth tables of the functions. The specific statistical test is called d-monomial [81]. The d-monomial test compares the output of a symmetric cipher with that of a random Boolean function. Normally, the focus is on the frequency of the special monomials in the ANF of the Boolean functions, but for the attack, the focus is on the truth table. The framework distinguishes between random sequences and keystreams of ACORN-v3 and for up to 676 initialization rounds with a time complexity of 200×2^{33}. A big advantage of this proposed framework is that the accuracy of the test can be adjusted to the available computing power. It is stated that it is the best practical attack on ACORN-v3 so far.

A differential cryptanalysis of initialization is presented by the designers [44]. From the results, ACORN is proved to be secure against differential cryptanalysis.

A fault differential attack is proposed against ACORN-v3 [82]. The fault attack [83], [84] is categorised as a side channel attack meaning that it works on physical implementations and the cipher can be weakened by injecting a fault. The cipher had to be tested to this kind of attacks too, because it's an algorithm that maybe will be used in reality. The attack is composed mainly of two procedures: fault locating and equation solving which are detailed explained in the work. Then a guess-and-determine method is used to obtain the initial state. It is stated in the work that with n fault experiments ($26 < n < 43$), the initial state can be recovered with time complexity $c \times 2^{146,5-3,52\times n}$, where c is the time complexity of solving linear equations. It is also shown that ACORN-v3 is more vulnerable to this attack than ACORN-v2. Of course, this doesn't mean that this

attack is a problem for the security of ACORN-v3 since it requires many preconditions and high time complexity.

Another fault attack was conducted against ACORN-v3 [85], that requires 9 faults for cryptanalysis to be possible. A successful attack means that the secret state is compromised and thus the secret key is obtained too. The method of the attack is similar to that of a differential fault attack on Plantlet [86]. The problem of implementing this method to ACORN-v3 was its large state with the complicated update. To overcome this, some bits are fixed for the differential fault attack to be possible with the drawback of increasing the time complexity. The big contribution of this work is the small number of faults that are required for the cryptanalysis.

4.2 Grain Family

Grain-128a is the cipher that offers authentication but first there must be a reference to Grain-v1 and Grain-128 that are former versions of the cipher since the basic structure remains the same.

4.2.1 Grain-v1

The cipher was submitted to the eSTREAM project and it stood out for its easiness in hardware implementation and because is able to be applicated in very limited hardware environments. It also has the feature of increasing its speed in case of extra resources in the hardware. The original version (*v0*) was weak and after some observations and changes, the final version was proposed that is known as Grain-v1 [9].

Grain is a bit oriented synchronous stream cipher. The structure is based in two shift registers of 80 bits, one LFSR and one NFSR. The LFSR guarantees the lower bound of the *period* and provides *balance* to the output and the NFSR adds the *nonlinearity*. The secret key is 80 bits and the *IV* is specified to be 64 bits.

The cipher has three main building blocks, the LFSR $f(x)$, the NFSR $g(x)$ and a nonlinear filter function $h(x)$. The content of the LFSR is denoted by $s_i, s_{i+1}, \dots, s_{i+79}$ and the content of the NFSR is denoted by $b_i, b_{i+1}, \dots, b_{i+79}$.

The feedback polynomial of the LFSR is defined as:

$$f(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80}$$

The feedback polynomial of the NFSR is defined as:

$$\begin{aligned} g(x) = 1 &+ x^{17} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{65} + x^{71} + x^{80} + x^{17}x^{20} \\ &+ x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} \\ &+ x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} \\ &+ x^{28}x^{35}x^{43}x^{47}x^{52}x^{59} \end{aligned}$$

The contents of LFSR and NFSR represent the state of the cipher. Out of this state, 5 variables are used as input to a Boolean function $h(x)$. This function is balanced, correlation immune of the 1^{st} order, has algebraic degree 3 and $NL_{h(x)} = 12$.

The filter function is defined as:

$$h(x) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$$

where the variables x_0, x_1, x_2, x_3 and x_4 correspond to $s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}$ and b_{i+63} respectively. The output of the filter function produces the keystream of the cipher.

4.2.2 GRAIN-128

Due to the increase of the processing capabilities of computers Grain-v1 was not considered secure because it was vulnerable to exhaustive attacks (it required approximately 2^{80} of computational complexity). Grain-128 [87] was designed and proposed to meet the requirements of security of the time and still possessed the advantages of Grain-v1. Grain-128 supports a 128-bits key and a 96-bits IV. It is stated by the designers that at that time there was not another 128-bit cipher that could offer such security and ease in hardware implementation.

The cipher, as Grain-v1, consists of an LFSR, an NFSR and an output function with some differences. The content of the LFSR is denoted by $s_i, s_{i+1}, \dots, s_{i+127}$ and the content of the NFSR is denoted by $b_i, b_{i+1}, \dots, b_{i+127}$. The 256 bits of both of these shift registers represent the state of the cipher.

The LFSR is defined as:

$$f(x) = 1 + x^{32} + x^{47} + x^{58} + x^{90} + x^{121} + x^{128}$$

The NFSR is defined as:

$$g(x) = 1 + x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{60} + x^{61}x^{125} + x^{63}x^{67} + x^{69}x^{101} + x^{80}x^{88} + x^{110}x^{111} + x^{115}x^{117}$$

From the state of the cipher, 9 variables (2 from the NFSR and 7 from the LFSR) are used as input in the Boolean function $h(x)$ as follows:

$$h(x) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8$$

where $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$ correspond to the tap positions $b_{i+12}, s_{i+8}, s_{i+13}, s_{i+20}, b_{i+95}, s_{i+42}, s_{i+60}, s_{i+79}, s_{i+95}$ respectively.

The output function, which gives the keystream, is defined as:

$$z_i = \sum_{j \in A} b_{i+j} + h(x) + s_{i+93}$$

where $A = \{2,15,36,45,64,73,89\}$.

4.2.3 GRAIN-128a

This is a new version of Grain-128 that offers authentication and is strengthened against all known attacks. The modifications to this new version did not affect the basic structure of Grain-128 that, as it previously mentioned, is based on Grain-v1. Thus, the hardware performance of Grain-128a is close to that of the former versions.

The cipher supports two modes of operations, with and without authentication. When $IV_0 = 1$, the authentication is mandatory and when $IV_0 = 0$, it's forbidden. The authentication tag, denoted by w is up to 32 bits in size and it doesn't affect the keystream of Grain-128a. If there is no authentication, the cipher can be more efficient due to its construction.

Most of the parametres of the cipher are the same as the previous version. The key and IV size remain the same. The polynomials of the LFSR $f(x)$ and the Boolean function $h(x)$ also remain the

same. The output function becomes a pre-output function and a new output function is presented. There is an addition in the NFSR $g(x)$ to strengthen the cipher against known attacks as it is stated by the designers.

The LFSR is defined as:

$$f(x) = 1 + x^{32} + x^{47} + x^{58} + x^{90} + x^{121} + x^{128}$$

The NFSR is defined as:

$$\begin{aligned} g(x) = 1 &+ x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{60} + x^{61}x^{125} + x^{63}x^{67} + x^{69}x^{101} \\ &+ x^{80}x^{88} + x^{110}x^{111} + x^{115}x^{117} + x^{46}x^{50}x^{58} + x^{103}x^{104}x^{106} \\ &+ x^{33}x^{35}x^{36}x^{40} \end{aligned}$$

The last three monomials are the addition to $g(x)$ of Grain-128.

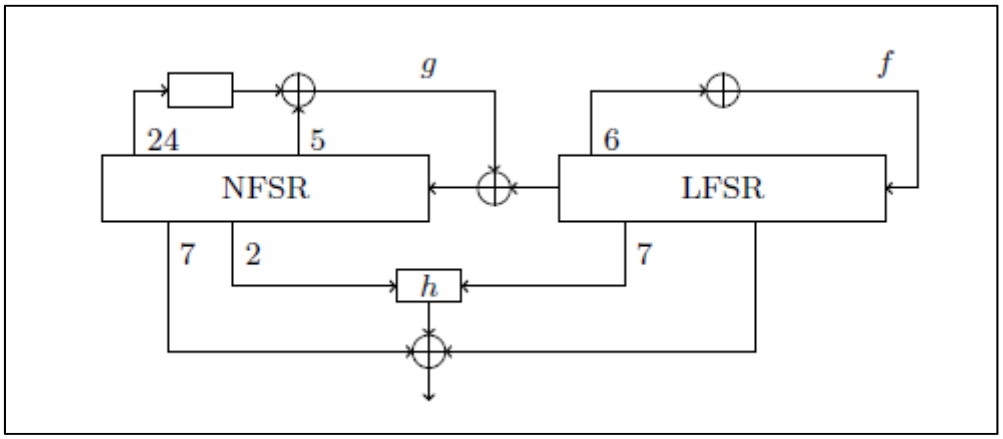

Figure 4.2: An overview of the pre-output generator

The pre-output function is defined as

$$y_i = \sum_{j \in A} b_{i+j} + h(x) + s_{i+93}$$

where $A = \{2,15,36,45,64,73,89\}$.

For the generation of the keystream, y_i is used but the is a small difference in the generator that depends on the mode of operation. If $IV_0 = 1$ the output function is defined as

$$z_i = y_{64} + 2i$$

If $IV_0 = 0$, then

$$z_i = y_i$$

meaning that without authentication, the keystream of the cipher is generated as in Grain-128.

4.2.4 Security Analysis

A related-key chosen IV attack was conducted on Grain-v1 and Grain-128 [88] that is an extended version of the slide resynchronization attack [89]. The attack takes advantage of the similarity of the setup mode of the ciphers with the keystream generation mode which is common for all ciphers in the Grain family. The attack on Grain-v1 needs $2^{22.59}$ chosen IVs, $2^{26.29}$ bits of keystream sequences and $2^{22.90}$ computational complexity to recover the secret key. The same attack on Grain-128 needs $2^{26.59}$ chosen IVs, $2^{31.39}$ bits of keystream sequences and $2^{27.01}$ computational complexity to recover the secret key.

The attack described in the previous paragraph is possible due to the symmetric padding that is used in Grain-v1 and Grain-128. Grain-128a has asymmetric padding, thus the specific chosen IV related key attack is not applicable to this new version of Grain. A functionable chosen IV related key attack for Grain-128a [90] was later published and its goal is to recover the key. In the paper it is proved that using around 2^{40} related keys and 2^{72} chosen IVs, it's possible to obtain 32×2^8 simple nonlinear equations that gives the secret key when solved. The complexity for this attack is better than exhaustive search. A countermeasure for this attack, that would make its complexity higher than a brute force would be to extend the padding to be at least equal to half the length of the secret key [90].

An attack based on differential cryptanalysis and targets cryptosystems that include NLFRs in their constructions was also used against Grain [91]. What is requested is to obtain deterministic differential characteristics for large number of rounds by identifying conditions on the internal state. Depending on these conditions, distinguishing and partial key recovery attacks are derived. The technique is applied to Grain-v1 and on Grain-128. Grain-128 can be distinguished for up to 215 of its 256 rounds and some parts of the key can be recovered for up to 213 rounds.

Standard cube attacks obtain the key by solving linear equations in the key bits. The *dynamic cube attack* [92] recovers the secret key by exploiting distinguishers obtained by cube testers. The attack can recover the key of Grain-128 if the number of initialization rounds is reduced to 207 for a feasible time complexity. The attack is also done with 250 initialization rounds and it's shown that this method is faster than exhaustive search. The attack on the full version of the cipher can be successful if the key belongs to a subset 2^{-10} of possible keys. It is stated that in this paper it was the first time that a cube attack is effective against a well-known, considered secure cipher.

A single key attack is used against Grain-128 [93] which can recover the secret key by an algorithm significantly faster than exhaustive search. The paper states that for 7,5% of keys, there is an improvement factor of 2^{38} over exhaustive search. There are no assumptions taken into account for this attack. The only restriction to the attack is that it needs dedicated hardware, due to high complexity and hardware-oriented nature. The attack may be infeasible but it presents better results than other methods.

The attack framework that was mentioned for ACORN-v3 [80], was also applied to Grain-128a in the same paper. The framework distinguishes between random sequences and keystreams of Grain128a and for up to 171 initialization rounds with a time complexity of 200×2^{28}. As in ACORN, the attack has good results, but the security of the cipher is not contained.

The newly introduced MAC in the Grain family is used to authenticate the message. However, it can be also be used by cryptanalysts to compromise the cipher. This scheme was already described [94] and it uses a differential fault attack on the cipher to recover the key by observing the correct and faulty MACs that are produced for certain chosen messages. This attack is functionable due to specific properties of Boolean functions and corresponding choices of the taps from the LFSR. The attack requires less than 2^{11} fault injections and invocations of less than 2^{12} MAC generation routines to find the secret key.

Not many cryptanalytic studies had been published about Grain-128a because not many years has passed since the introduction of the cipher in 2011. However, Grain-128a is considered to be a cipher of high security level until now and because of this it has been standardized for radio frequency identification (RFID) devices.

4.3 PALS

This cipher was introduced by Ashouri [95] and it is designed to resist all known conventional attacks. It is a clock-controlled stream cipher with a mechanism of altering steps. The main key's size is 256 bits and the message key's size is 32 bits. Important criteria of the cipher are maximum period, high linear complexity and good statistical properties. The base structure is a clock-controlled combination generator with memory.

The main and the message keys are used to generate a session key of 256 bits that is extended to 1600 bits and becomes the initial vector.

Message key generator

In PALS, a message key is produced by an LFSR of 32 bits, whose feedback function is represented by the following polynomial:

$$C(x) = x^{32} + x^{29} + x^{24} + x^{23} + x^{21} + x^{19} + x^{17} + x^{16} + x^{14} + x^{13} + x^{11} + x^{9} + x^{6} + x^{3} + 1$$

Session key generator

The message key bits should be altered by 50% (Avalanche effect) to be used on the main key (Mk) and generate the session key (S_k). For this, a permutation and a substitution box are used for the 32 bits message key to obtain good diffusion that produce a 32-bit sequence. This operation is repeated eight times and the diffused sequence (D)

$$Sk_i = Mk_i \oplus D_i$$

Initial Vector Generator

For the production of the IV, an LFSR of 256 bits, a polynomial of degree 256 and four S-boxes are used. The session key is used as the initial state of the LFSR and generates 8 bits at any clock. For the diffusion, the S-boxes are used and the first 320 generated bits are discarded. The next 1600 bits are used as the IV.

Keystream generator

The generator is based on 8 LFSRs with different lengths that are clocked irregularly using one of the four S-boxes of the IV generator. 8 bits that are selected in different stages of the LFSRs are used as input to eight nonlinear Boolean functions of $NL = 6$ and correlation immunity of the 2^{nd} order. Each function has 9 input variables and the 9^{th} variable's value is a bit of the S-box's output. The output of these 8 functions enters the function g. The 9^{th} variable of the function g is taken by the output of the nonlinear function h. The output of the function g is the keystream.

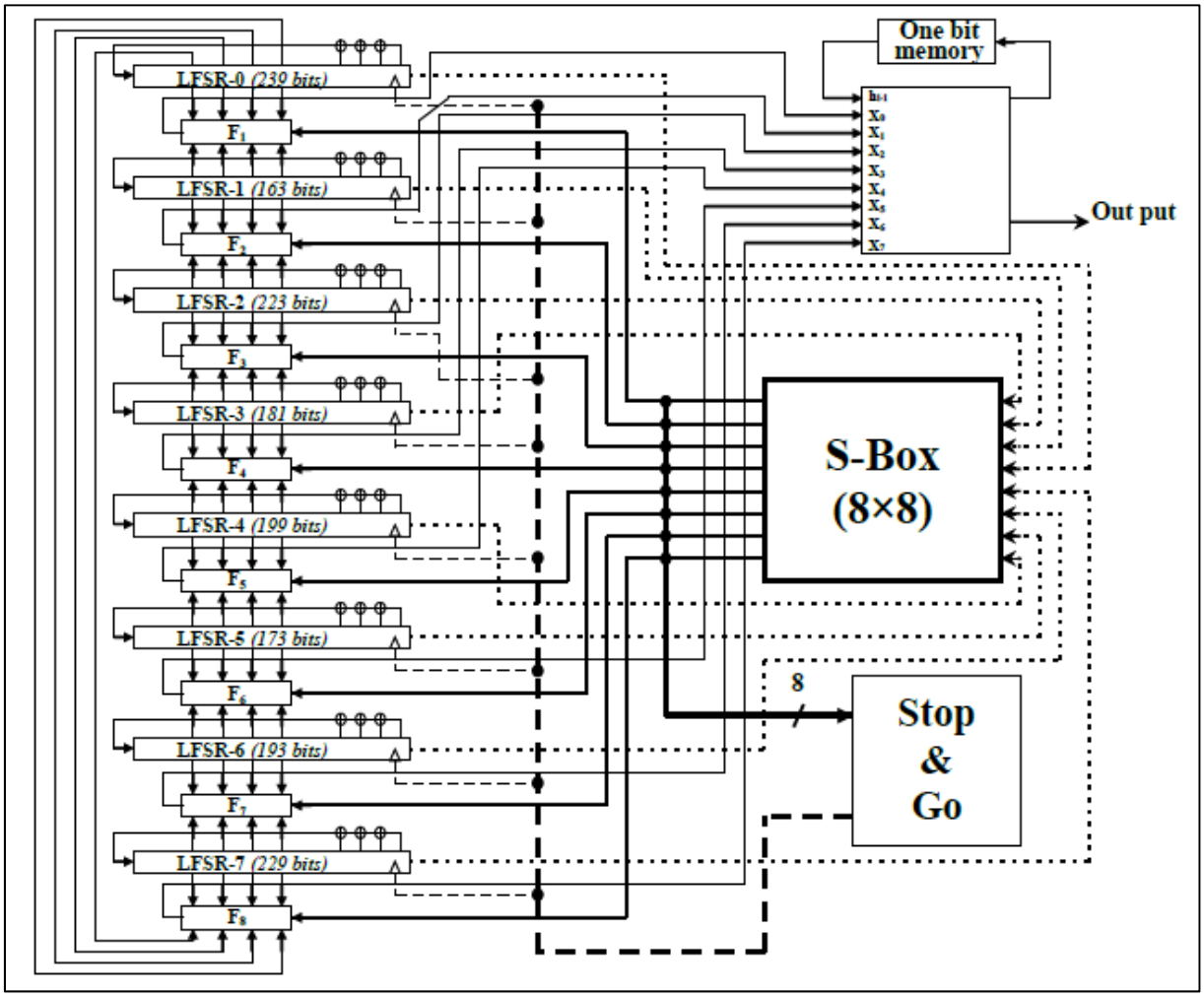

Figure 4.3: Keystream generator

The polynomial of the output combiner (h and g functions) is as follows:

$$h_i = x^1 + x^2 + x^5 + x^5x^3 + x^6x^4 + x^7x^0 + x^7x^1 + x^7x^5 + x^8x^0 + x^8x^2 + x^8x^7x^0 + x^8x^7x^1 + x^8x^7x^3x^2 + x^8x^7x^4x^2 + x^8x^7x^4x^3x^2 + x^8x^7x^5x^2 + x^8x^7x^5x^3x^2 + x^8x^7x^5x^4x^2 + x^8x^7x^5x^4x^3x^2 + x^8x^7x^6x^2 + x^8x^7x^6x^3x^2 + x^8x^7x^6x^4 + x^8x^7x^6x^4x^2 + x^8x^7x^6x^4x^3 + x^8x^7x^6x^4x^3x^2 + x^8x^7x^6x^5 + x^8x^7x^6x^5x^2 + x^8x^7x^6x^5x^3 + x^8x^7x^6x^5x^3x^2 + x^8x^7x^6x^5x^4 + x^8x^7x^6x^5x^4x^2 + x^8x^7x^6x^5x^4x^3 + x^8x^7x^6x^5x^4x^3x^2$$

$$g = x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + h_{i-1}$$

4.3.1 Security Analysis

PALS is a new algorithm that was submitted in 2018. Therefore, there are not yet published attacks on the cipher except the ones that were performed by the designer in [95], in order to display its high level of security against known attacks.

Chapter 5

The Approximation Technique applied to the Algorithms

In this chapter, the process of finding approximations with fewer variables that was described in Chapter 3 is applied to the ciphers of Chapter 4. Specifically, the LPA is used on the sequences of the truth tables of Boolean functions that are used in the algorithms of the ciphers[3].

To achieve this, the Boolean functions are expressed in their ANF[4]. For better results in the approximation functions the variables may be permuted. Then, the truth table of the Boolean function is generated and the LPA is adjusted to respond to the sequence of the truth table. The sequence is set as input to the algorithm and after that the algorithm is executed. The output

[3] For the generation of the truth table, an application that is available on the web was used that was developed by *Southwestern Adventist University.* http://turner.faculty.swau.edu/mathematics/materialslibrary/truth/

[4] For simplicity reasons, characteristic polynomials are associated with Algebraic Normal Forms (ANFs) in some occasions in this thesis, since this leads to permutation equivalent functions

includes the CPs. The SCPs are written down and present the distance of each approximation function.

For the creation of the approximation function, as it is mentioned in Chapter 3, the LPA must be modified to present which bits must be changed. After that, the truth table of the approximation sequence is built and the function is built based on that sequence.

5.1 ACORN

The keystream bit is generated as follows:

$$ks_i = S_{i,12} \oplus S_{i,154} \oplus maj(S_{i,235}, S_{i,61}, S_{i,193}) \oplus ch(S_{i,230}, S_{i,111}, S_{i,66})$$

We know that

$$maj(x, y, z) = xy \oplus xz \oplus yz$$

$$ch(x, y, z) = xy \oplus \big((\sim x)z\big) = xy \oplus z \oplus xz$$

Thus, the keystream generator can be written as:

$$ks_i = S_{i,12} \oplus S_{i,154} \oplus S_{i,235}S_{i,61} \oplus S_{i,235}S_{i,193} \oplus S_{i,61}S_{i,193} \oplus S_{i,230}S_{i,111} \oplus S_{i,66} \oplus S_{i,230}S_{i,66}$$

Replacing the S_i with x_n, we have the following Boolean function:

$$kx_n = x_1 \oplus x_5 \oplus x_8x_2 \oplus x_8x_6 \oplus x_2x_6 \oplus x_7x_4 \oplus x_3 \oplus x_7x_3$$

According to Proposition 1, the highest possible approximation that depends on $n - 1$ variables for a function of $deg = 2$ is:

$$\left(1 - \frac{1}{2^2}\right) \times 100\% = 75\%$$

The critical points are (0,161), (64,73), (96,17), (128,0). The SCPs are (64,73), (96,17). Like it was mentioned in Chapter 3, the first number displays the number of bits that need to be changed, thus the distance between the two functions and the second number displays the corresponding linear

complexity which is used to find out the number of variables. Based on the previous, Table 5.1 displays at each row the number of variables, the distance and the approximation of each SCP.

Variables	Distance	Approximation (%)
7	64	75
5	96	62,5

Table 5.1: ACORN I

The approximation that depends on $n-1$ variables is the highest possible, although x_8 appears in two monomials. That is because x_8 affects $\frac{2^8}{2^2}$ (64) bits even if it exists in two monomials. Like it was shown in *subchapter 3.3.2*, it is not necessary for variables to only exist once in an ANF to find the best approximation functions.

The next approximation function depends on $n-3(=5)$ variables. Apparently, the distance of the function that depends on $n-2(=6)$ variables has the same distance as the one that depends on one less variable from the function kx_n. This means that the distance between kx_n and the function that depends on $n-2$ is also 96. However, the LPA only showed the one with the less linear complexity, which is the one that depends on $n-3$ variables.

Variable x_7 affects $\frac{2^8}{2^2}$ (64) bits in the function kx_n as variable x_8 but because x_7 is the second variable that needs to be removed and there would be only 6 variables left in the function, other things should be taken into account – the interaction between the $\mathcal{BA}$ by variables x_7 and x_8.

As concerns the variable x_6, it also affects $\frac{2^8}{2^2}$ (64) bits of function kx_n. The function that depends on $n-3$ variables has the same distance as the one that depends on $n-2$ variables because of the interaction of the $\mathcal{BA}$.

As it is presented, only two approximations are computed by this order of variables. There are no more approximations because x_5 only appears once in a monomial of degree 1. However, more approximations may be found because there are more variables that appear in monomials with $deg > 1$ that can be permitted with x_5.

By permitting the variables the new order is as follows:

$S_{i,12}$	$S_{i,61}$	$S_{i,66}$	$S_{i,111}$	$S_{i,154}$	$S_{i,193}$	$S_{i,230}$	$S_{i,235}$
x_1	x_2	x_3	x_5	x_4	x_6	x_7	x_8

Table 5.2: ACORN II

Thus, the ANF of the new permuted equivalent function is:

$$kx_n = x_1 \oplus x_4 \oplus x_8 x_2 \oplus x_8 x_6 \oplus x_2 x_6 \oplus x_7 x_5 \oplus x_3 \oplus x_7 x_3$$

The sequence of the truth table of the above function is used as input to the LPA that gives the following CPs: (0,161), (64,81), (96,9), (128,0). The approximations that can be built by these CPs are presented in Table 5.3.

Variables	Distance	Approximation (%)
7	64	75
4	96	62,5

Table 5.3: ACORN III

This means that an approximation function can be built that depends on $n - 4(= 4)$ variables and it has the same distance as the approximation function that depends on $n - 3$ variables that was computed for the normal order of the variables.

The approximations of this function also stop at the point of $S_{i,154}$. Looking to find better approximations than the previously mentioned ones and taking into account the function with the permitted variables, $S_{i,154}$ was permuted with $S_{i,66}$ and later in another function with $S_{i,61}$ but no better approximation function were found. Thus, the results in Table 5.3 are considered to be the best approximations that can be found by the LPA for function ks_i.

It is important to mention that, if the variables x_i are replaced with x_{9-i} for $1 \leq i \leq 8$, it is confirmed that there are no SCPs because the last variable only exists once in the ANF, in a monomial of degree 1.

5.2 GRAIN-v1

NFSR

The feedback polynomial of the NFSR is defined as:

$$g(x) = 1 + x^{17} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{65} + x^{71} + x^{80} + x^{17}x^{20} + x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} + x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} + x^{28}x^{35}x^{43}x^{47}x^{52}x^{59}$$

For the purpose of analysing $g(x)$, $(x^{17}, x^{20}, x^{28}, x^{35}, x^{43}, x^{47}, x^{52}, x^{59}, x^{65}, x^{71}, x^{80})$ are replaced by $(x^1, x^2, x^3, x^4, x^5, x^6, x^7, x^8, x^9, x^{10}, x^{11})$ respectively.

Thus, $g(x)$ can also be written as:

$$g(x) = 1 \oplus x^1 \oplus x^2 \oplus x^3 \oplus x^4 \oplus x^5 \oplus x^6 \oplus x^7 \oplus x^8 \oplus x^9 \oplus x^{10} \oplus x^{11} \oplus x^1x^2 \oplus x^5x^6 \oplus x^9x^{10} \oplus x^2x^3x^4 \oplus x^6x^7x^8 \oplus x^1x^4x^7x^{10} \oplus x^2x^3x^5x^6 \oplus x^1x^2x^8x^9 \oplus x^1x^2x^3x^4x^5 \oplus x^6x^7x^8x^9x^{10} \oplus x^3x^4x^5x^6x^7x^8$$

(Note that the constant term 1 does not affect the degree to which a function can be approximated by another function with fewer number of variables).

By observing the polynomial, one can see that the variable x^{11}, which is the last variable, only appears once in a monomial with degree 1. This means that it affects the maximum possible bits; $\mathcal{BA}_{x^{11}} = \frac{2^{11}}{2}$. As a result, it is certain like it was mentioned in Chapter 3, that there won't be any SCPs. Applying the LPA to the sequence of the truth table of $g(x)$, it is presented that indeed there are only two CPs: (0,1025), (1024,0).

Therefore, in order to find good approximations of this function, the variables must be permuted. It is interesting to observe that just by replacing x^i with x^{12-i} for $1 \leq i \leq 11$ there are some good approximations that are presented in Table 5.4.

Variables	Distance	Approximation (%)
10	624	69,5
9	652	68,2
8	832	59,4
7	868	57,6
6	948	53,7
2	1004	50,1

Table 5.4: Grain-v1 NFSR I

It can be observed that the $n-1$ approximation has a big difference from the best approximation that is possible for a function of $deg = 6$ which is:

$$(1 - \frac{1}{2^6}) \times 100\% = 98{,}4\%$$

One could see that the variable x^1 which was placed last doesn't exist in a monomial which has degree equal to $\deg(g)$. So, one could think that maybe it would be better to place a variable that exists in a monomial with degree that equals $\deg(g)$. This is not always true. As an example, the variables are permuted as is shown in Table 5.5.

x^{17}	x^{20}	x^{28}	x^{35}	x^{43}	x^{47}	x^{52}	x^{59}	x^{65}	x^{71}	x^{80}
x^4	x^3	x^6	x^7	x^{10}	x^9	x^8	x^{11}	x^2	x^5	x^1

Table 5.5: Grain-v1 NFSR II

The last variable was set to be x^{59} to examine if the approximation function that depends on $n-1$ variables has less distance than the approximation that depends on $n-1$ variables when x^{17} is set to be last. The first approximation is very important because the approximations that depends on less variables have greater distance of the ones with more variables. For example, if the approximation of the function that depends on $n-1$ variables is 80%, then the approximations of the functions that depends on $n-x$ variables, where $x = 2,3,\dots,n-2$, is $\leq 80\%$ for the same x_n.

The approximations of the permuted function (Table 5.5) are shown in Table 5.6.

Variables	Distance	Approximation (%)
10	752	63,3
9	832	59,4
8	856	58,2
7	864	57,8
6	964	52,9
5	980	52,1
3	996	51,4

Table 5.6: Grain-v1 NFSR III

It can be seen that the approximation that depends on $n-1$ variables is not better than the one computed in Table 5.4.

From Table 5.6, it can be observed that the approximation functions with the least distance are the ones with 8 and 7 variables. This could be seen as a weakness for variable x^{52} and that it could compute a good approximation that depends on $n-1$ variables if it is set last. So, it would seem interesting to place the 8^{th} variable last to observe the approximations that will be calculated.

x^{17}	x^{20}	x^{28}	x^{35}	x^{43}	x^{47}	x^{52}	x^{59}	x^{65}	x^{71}	x^{80}
x^4	x^3	x^6	x^7	x^{10}	x^9	x^{11}	x^8	x^2	x^5	x^1

Table 5.7: Grain-v1 NFSR IV

The approximation that depends on $n-1$ variables of the permuted function (Table 5.7) approximates $g(x)$ 63,3%, like the one of the previously permuted function (Table 5.6). So, the interaction with the variables that were after x^{52} was the reason that the distance was little.

Finding a better approximation would be easy if it was known how each variable interacts with the others and how many bits are affected by each variable. However, this is indeed difficult for $g(x)$ because it has many monomials of different degrees that varies from 1 to 6.

It would be interesting to observe which variable if set last has the best approximation that depends on $n-1$ variables. Table 5.8 presents the variables that are placed last and the corresponding distance and approximation of the functions that can be built and depend on $n-1$ variables.

Variable set last	Distance	Approximation (%)
x^{47}	592	71,1
x^{43}	592	71,1
x^{35}	592	71,1
x^{71}	544	73,4
x^{65}	544	73,4
x^{20}	576	71,9
x^{28}	752	63,3

Table 5.8: Grain-v1 NFSR V

Thus, the best approximation that can be computed with LPA for function $g(x)$ that depends on $n-1$ variables is 73,4%.

Filter function

The filter function is correlation immune of the 1^{st} order. The filter function is defined as:

$$h(x) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$$

where the variables x_0, x_1, x_2, x_3 and x_4 correspond to $s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}$ and b_{i+63} respectively. The output of the filter function produces the keystream of the cipher.

The SCPs of $h(x)$ are shown in Table 5.9.

Variables	Distance	Minimum Distance	Approximation (%)
4	8	2,733501	75
3		8	
2	12	12	62,5

Table 5.9: Grain-v1 Filter Function I

The 3rd column presents the lowest possible distance of an approximation function to $h(x)$ based on equation (1) of this thesis. It is observed that for the approximation function that depends on 2 variables the distance is the least possible. For the approximation function that depends on 4

variables it is observed that the distance can be lower. Thus, a better approximation with 4 variables may be calculated with LPA. For this, a different approach is used.

The truth table of function $h(x)$ is a sequence of 32 bits. Thus, it is easier to observe how the variables interact with each other. For this, variable x_4 is examined that was last in the function and it needed 8 bits to be removed for the approximation function with 1 less variable. $\mathcal{BA}_{x_4}$ depends on these monomials:

$$x_4 + x_3x_4 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$$

	VARIABLES					MONOMIALS				
	x_0	x_1	x_2	x_3	x_4	x_4	x_3x_4	$x_0x_2x_4$	$x_1x_2x_4$	$x_2x_3x_4$
17	0	0	0	0	1	×				
18	1	0	0	0	1	×				
19	0	1	0	0	1	×				
20	1	1	0	0	1	×				
21	0	0	1	0	1	×				
22	1	0	1	0	1	×		+		
23	0	1	1	0	1	×			+	
24	1	1	1	0	1	×		+	×	
25	0	0	0	1	1	×	+			
26	1	0	0	1	1	×	+			
27	0	1	0	1	1	×	+			
28	1	1	0	1	1	×	+			
29	0	0	1	1	1	×	+			×
30	1	0	1	1	1	×	+	×		+
31	0	1	1	1	1	×	+		×	+
32	1	1	1	1	1	×	+	×	+	×

Table 5.10: Grain-v1 Filter Function II

In Table 5.10 is presented only the second half of the truth table because on the first half variable $x_4 = 0$ and no bits are affected. By the symbol × is meant that the specific bit in the output column of the truth table is affected by the variable and by the symbol + is meant that the bit is restored in its initial state. The shaded background of the boxes that include these symbols, indicates the last state of the bit.

Thus, the variable x_4 affects:

$$\frac{32}{2} - \frac{32}{4} - (\frac{32}{16} - \frac{32}{16}) - (\frac{32}{16} - \frac{32}{16}) - (\frac{32}{16} - \frac{32}{16}) - (\frac{32}{16} - \frac{32}{16}) = 8$$

Of course, doing this for variables that exist in functions that depends on greater number of variables is not so easy and it is time-consuming.

For variable x_1, that only exists in three monomials in the function ($x_1 + x_0x_1x_2 + x_1x_2x_4$), it is easier to compute $\mathcal{BA}_{x_1}$, that equals:

$$\frac{32}{2} - \frac{32}{8} - \left(\frac{32}{16} - \frac{32}{16}\right) = 12$$

Monomial x_1 affects 16 bits of the truth table. Monomial $x_0x_1x_2$ restore 4 bits to their initial state. At the monomial $x_1x_2x_4$ things get more complicated. However, it is easy to see that the previous monomial ($x_0x_1x_2$) restored 2 bits in the 1^{st} half of the truth table and 2 bits in the 2^{nd} half. The monomial $x_1x_2x_4$ affects 2 of the restored bits and restores 2 bits.

According to the above, the bits that are changed by x_1 are 12, so if x_1 is placed last, then the approximation function that depends on $n - 1$ variables should have a distance of 12 from $h(x)$. After the appropriate permutation, the SCP that is computed by LPA is: (12,9).

It is concluded that the best approximation that depends on 4 variables has a distance of 8 from $h(x)$. This is of course the best approximation that can be computed by using the LPA.

5.3 GRAIN-128

NFSR

The NFSR is defined as:

$$g(x) = 1 + x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{60} + x^{61}x^{125} + x^{63}x^{67} + x^{69}x^{101} + x^{80}x^{88} + x^{110}x^{111} + x^{115}x^{117}$$

The LPA has a limitation to the length of sequence that can take as input, which is 2^{12} meaning that 12 variables can be used. The function $g(x)$ has 19 variables and this complicates the procedure because applying the LPA algorithm to a 2^{19}-bit sequence is of high computational and memory requirements for our testing system.

Observing the function, it can be seen that all variables exist only once in the ANF. It is also observed that 5 variables in the function appear in monomials of degree 1. These 5 variables don't contribute to the computation of approximations, so all 5 can be ignored during the procedure, because they are of no use in the process of finding approximation functions, like it is shown in Proposition 2.

At this point, it is also safe to estimate the approximations that will be computed before using the LPA. For this function, it is ensured that there exist approximations with fewer number of variables, whose approximation degrees are 75% and $62{,}5\%$. This is because all the variables (excluding the 5 previously mentioned) exist in monomials of degree 2. It is also expected to compute approximations $< 62{,}5\%$.

The approximations can be estimated but the LPA is needed to calculate the number of variables for each approximation function. That is not possible with the current version of LPA, because excluding the 5 variables, there are still 14 variables.

Of course, the LPA is not necessary for the approximating function that depends on $n - 1$ variables. That is because each variable only appears once in the ANF, including the last one. Since it only appears once in a monomial of degree 2, it means that the approximation is 75%.

To be able to run the LPA, the function needs to be shortened to 12 variables. Thus, the following polynomial represent a part of the function $g(x)$:

$$x_1x_2 + x_3x_4 + x_5x_6 + x_7x_8 + x_9x_{10} + x_{11}x_{12}$$

(It can be observed that this specific part of the function is actually a bent function if taken as it is)

The use of a part of the whole function in the LPA can provide the way the distance grows while the variables that are needed decrease. This way, if there is a pattern, it is possible to find the approximations of the whole function without running the LPA. Of course, for the process of building these approximation functions the LPA is needed.

After running the part of the function in the LPA the following SCPs are calculated: $(1024{,}769), (1536{,}193), (1792{,}49), (1920{,}13), (1984{,}4)$. These results are illustrated in Table 5.11.

Variables	Distance	Approximation (%)
10	1024	75
8	1536	62,5
6	1792	56,3
4	1920	53,1
2	1984	51,6

Table 5.11: Grain-128 NFSR I

It is also necessary to mention CP: (2016,0) → Approximation 50,8%.

From the SCPs calculated by a part of the function, it is possible to understand what SCPs will occur if the truth table of the whole function is used in the LPA. That's because there is a pattern. The approximations depend on $12-2x$ variables and the distance is $2^{12-1}-\frac{2^{12-1}}{2^x}$ for $x=(1,2,3,4,5)$.

This pattern applied to the function $g(x)$ give us the results that are depicted in Table 5.12.

Variables	Distance	Approximation (%)
17	$\frac{2^{19}}{2^2}$ ή $2^{18}-\frac{2^{18}}{2}$	75
15	$2^{18}-\frac{2^{18}}{4}$	62,5
13	$2^{18}-\frac{2^{18}}{8}$	56,3
11	$2^{18}-\frac{2^{18}}{16}$	53,1
9	$2^{18}-\frac{2^{18}}{32}$	51,6
7	$2^{18}-\frac{2^{18}}{64}$	50,8
5	$2^{18}-\frac{2^{18}}{128}$	50,4

Table 5.12: Grain-128 NFSR II

Essentially, these approximation functions are nothing more than the same function with less variables. Each approximation function uses two less variables of one monomial of the polynomial, like it was shown in Proposition 1.

Filter function

From the state of the cipher, 9 variables (2 from the NFSR and 7 from the LFSR) are used as input in the Boolean function $h(x)$ as follows:

$$h(x) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8$$

where $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$ correspond to the tap positions $b_{i+12}, s_{i+8}, s_{i+13}, s_{i+20}$, $b_{i+95}, s_{i+42}, s_{i+60}, s_{i+79}, s_{i+95}$ respectively.

The highest possible approximation that depends on $n-1$ variables for this function is:

$$(1 - \frac{1}{2^3}) \times 100\% = 87{,}5\%$$

After running the LPA with sequence of the truth table the results that are presented in Table 5.13 are calculated. As it is shown, there is a function that depends on $n-1$ variables and it approximates $h(x)$ 87,5%

Variables	Distance	Approximation (%)
8	64	87,5
6	160	68,75
4	192	62,5
2	224	56,25

Table 5.13: Grain-128 Filter Function I

Since the best approximation function that depends on $n-1$ variables is found, there is no reason to permute the last variable. However, it is possible to find better approximations that depends on less variables. For this, the order of the variables is changed as in Table 5.14.

b_{i+12}	, s_{i+8}	s_{i+13}	s_{i+20}	b_{i+95}	s_{i+42}	s_{i+60}	s_{i+79}	s_{i+95}
x_7	x_1	x_2	x_3	x_6	x_5	x_4	x_0	x_8

Table 5.14: Grain-128 Filter Function II

As it can be seen, b_{i+12} and b_{i+95} are x_7 and x_6 respectively. This permutation was done to test if the variables that are in the same monomial and interact with x_8 can lead to better approximations. Of course, both of them are used in another monomial and that's something that complicates the interaction and the outcome. Thus, the function after the permutation is as follows

$$h(x) = x_7x_1 + x_2x_3 + x_6x_5 + x_4x_0 + x_7x_6x_8$$

The results of the LPA are presented in Table 5.15.

Variables	Distance	Approximation (%)
8	64	87,5
7	128	75
6	160	68,75
4	192	62,5
2	224	56,25

Table 5.15: Grain-128 Filter Function III

It is seen that not better approximations are computed, but there is one more approximation that depends on 7 variables.

5.4 GRAIN-128a

For Grain-128a, only the NFSR is tested because the function $h(x)$ is the same as in Grain-128.

The NFSR is defined as:

$$\begin{aligned} g(x) = 1 + x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{60} + x^{61}x^{125} + x^{63}x^{67} + x^{69}x^{101} \\ + x^{80}x^{88} + x^{110}x^{111} + x^{115}x^{117} + x^{46}x^{50}x^{58} + x^{103}x^{104}x^{106} \\ + x^{33}x^{35}x^{36}x^{40} \end{aligned}$$

The last three monomials are the addition to $g(x)$ of Grain-128. This function consists of 29 variables, 10 more than its previous version and once again all the variables only exist once in the polynomial.

This means that the LPA is not necessary in this function, like in the NFSR of Grain-128, in the process of computing the approximations. Like the NFSR of Grain-128, the number of the variables and the distance of the approximations from $g(x)$ depends on the degree of the monomials.

For example, to have the highest possible 1^{st} SCP, the variables of the monomial which is of degree 4 (since $\deg(g) = 4$), are permitted and placed last. The first approximation will be equal to $(1 - \frac{1}{2^4}) \times 100\% = 93{,}8\%$. This approximation function depends on 4 less variable of the function $g(x)$ which are the four prementioned variables. The distance of the approximation to the function $g(x)$ is $\frac{2^{29}}{2^4}$ bits. Thus, the 1st approximation depends on 25 variables, has a distance of $\frac{2^{29}}{2^4}$ bits and approximates $g(x)$ 93,8%. Specifically, this approximation function is the same as $g(x)$ without the last monomial.

Using Proposition 1, the rest of the approximations can be calculated without LPA.

5.5 PALS

The polynomial of the output combiner (h and g functions) is as follows:

$$\begin{aligned} h_i = {} & x^1 + x^2 + x^5 + x^5x^3 + x^6x^4 + x^7x^0 + x^7x^1 + x^7x^5 + x^8x^0 + x^8x^2 + x^8x^7x^0 \\ & + x^8x^7x^1 + x^8x^7x^3x^2 + x^8x^7x^4x^2 + x^8x^7x^4x^3x^2 + x^8x^7x^5x^2 \\ & + x^8x^7x^5x^3x^2 + x^8x^7x^5x^4x^2 + x^8x^7x^5x^4x^3x^2 + x^8x^7x^6x^2 + x^8x^7x^6x^3x^2 \\ & + x^8x^7x^6x^4 + x^8x^7x^6x^4x^2 + x^8x^7x^6x^4x^3 + x^8x^7x^6x^4x^3x^2 + x^8x^7x^6x^5 \\ & + x^8x^7x^6x^5x^2 + x^8x^7x^6x^5x^3 + x^8x^7x^6x^5x^3x^2 + x^8x^7x^6x^5x^4 \\ & + x^8x^7x^6x^5x^4x^2 + x^8x^7x^6x^5x^4x^3 + x^8x^7x^6x^5x^4x^3x^2 \end{aligned}$$

$$g = x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + h_{i-1}$$

The way the output combiner of PALS is built (having the NFSR output as input in an LFSR), makes the approximation techniques on function h_i of limited cryptographic value for the cryptanalysis of the cipher. However, it is still considered important to observe how the NFSR resist to this kind of

approximation technique because if the NFSR is compromised, then the difficulty to compromise the whole output combiner is decreased.

The function h_i consists of 9 variables and as it can be observed, it depends on many monomials of different degrees and all variables appear many times. This leads to a complex interaction between the variables, that complicates the procedure of finding better approximations with permutation. Thus, the function h_i is tested as it is.

The highest possible approximation that depends on $n - 1$ variables for this function is:

$$(1 - \frac{1}{2^6}) \times 100\% = 98{,}4\%$$

However, it is not expected to have such a high approximation due to the interaction of the variables.

After running the sequence of the truth table of the function in the LPA, the following SCPs are computed: $(128{,}131)$, $(160{,}97)$, $(188{,}40)$, $(208{,}17)$, $(240{,}9)$. Table 5.16 illustrates the details of each approximation function.

Variables	Distance	Approximation (%)
8	128	75
7	160	68,8
6	188	63,3
5	208	59,4
4	240	53,1

Table 5.16: PALS I

The function that depends on 8 variables approximates h_i 75%. This is not even close to the 98,4%. The other approximations have a slightly bigger distance from h_i from the approximations that depends on one more variable, with the last one that depends on 4 variables approximating h_i just $53{,}1\%$.

If the best approximations need to be calculated for this function with LPA, the $\mathcal{BA}$ need to be calculated for each variable and the interaction of these bits.

Chapter 6
Conclusions

The authenticated stream ciphers share similarities in their basic structures. The general framework of authenticated stream ciphers, which is based on these similarities, is considered to be very important and useful in the cryptographic community; many authenticated stream ciphers lie in the class of lightweight ciphers and thus, they are candidates for adoption in Internet of Things (IoT) applications, in which devices connected to the internet that can utilize limited resources for the purpose of encryption and authentication are being used. Authenticated stream ciphers combine security, high speed of execution and low hardware complexity. This means that such ciphers can be built to be lightweight, like ACORN. As a result, there is much attention to this category of ciphers by cryptographers and there will be much more in the upcoming years.

In this thesis, three authenticated stream ciphers are analysed and all of them are considered to be secure. Even if there are some attacks on Grain-128a and ACORN that show good results, there is no attack that comes close to compromising the ciphers. Of course, it should be mentioned that most of the attacks that are used on the ciphers are designed for prior ciphers that don't fall into the category of authenticated stream ciphers and they had to be adjusted. Thus, the designers could take into account these attacks and use the info to make sure that the ciphers would withstand

them. Future studies should concentrate more on the cryptanalytic methods on specifically authenticated stream ciphers. Especially the fusion of authentication and encryption methods should be properly examined for weaknesses that can be used to compromise the ciphers. This, certainly doesn't mean that attacks that are not designed specifically for authenticated stream ciphers should not be considered important. Such an attack could be one using the approximation technique that was described in Chapter 3.

The approximation technique is based on the relationship of binary sequences and the truth table of Boolean functions. The technique takes advantage of the advanced study that has been done on binary sequence and applies it to the truth table of Boolean functions. Specifically, the Error Linear Complexity Spectrum (ELCS) is used to find approximation functions that depend on less variables. This means that this technique could be widely used because of the vast use of Boolean functions in cryptographic ciphers. Based on the previous, the approximation technique can be considered as an evaluation tool for new ciphers but also can be used to existing algorithms in future studies.

In addition to the study of the technique, this thesis also introduces a theoretical framework that can be used to have the optimal results in the calculation of the approximations. As it was described in Chapter 3, the technique highly depends on the order of the variables of the function. Based on this, it was proved in this thesis that the proper order of the input variables can be calculated for some specific cases. It is shown how the bits affected by a variable in the truth table can lead to the highest possible approximation function that depends on 1 less variable. The method of calculating the number of the bits affected is given in detail in Chapter 3. However, to calculate approximation functions that depends on more than 1 less variables, the interaction between the bits affected in the truth table of the variables that will be disregarded must be calculate. In case the approximation function depends on 1 less variable, the interaction of variables is not taken into account because only one variable is disregarded.

Apart from the theoretical framework that is used to properly permute the input variables, there are also some suggestions that are useful to determine the right order of the variables in the approximation technique. It is also stated in which cases this technique is not necessary to compute the approximations and which parts (if there are any) of Boolean functions don't contribute to the calculation of approximations. It is also shown that the approximation is linked to the degree of the function and with that degree an upper bound limit (that concerns only the approximations that can be found by LPA) of the approximations can be calculated even before using the technique. All the previous, give a major advantage in the process of evaluating a function using this technique as

a cryptographic criterion. Furthermore, the work that is done in this thesis paves the way for further studies on this subject. For example, a future study could focus on the way the disregarded variables interact with one another.

Except the theoretical analysis of the approximation technique, the latter was also applied to the three authenticated stream ciphers and also the two prior ciphers of Grain-128a that are described in Chapter 4. The results of the application of the technique are written down in Chapter 5 and the methods of the theoretical framework that are introduced in this thesis are used, so that the technique is more effective. Each one of the ciphers has a different reaction to the technique that is mentioned in brief in the following paragraphs.

Firstly, the approximation technique is applied to ACORN. Due to the function's low degree, the approximation's upper bound limit for a function that depends on 1 less variable is only 75%. However, a function of this approximation (75%) is possible and it is calculated. After this, a function that approximates the function ks_i 62,5% is calculated that depends on 4 less variables. The approximation may be low, but 4 variables are disregarded. It is reminded that ks_i uses 8 variables, so only half of the variables are used in the approximation function that is indeed a critical reduction. It is also remarked that these approximations are the best possible that can be calculated by the technique.

The next ciphers that are tested are the ones of the Grain family and for start Grain-v1. For the feedback function of the NFSR that has 11 input variables there are three different permutations of the function $f(x)$ that are used for the technique, so as a result there are approximation functions that vary in terms of the depending variables (2 to 10) and of the approximations (50,1 – 69,5%). After that, the best possible approximation function that depends on 1 less variable is calculated and it approximates $f(x)$ 73,4%. For the filter function $h(x)$ that has 5 input variables, the best possible approximations that depends on 2 and 4 variables are calculated which approximate $h(x)$ 62,5% and 75% respectively. The one that depends on 4 variables is calculated after computing the $\mathcal{BA}$ of all the variables. The one with the least value is used as the last variable to calculate the prementioned approximation function. As concerns the other approximation function that depends on 2 variables, no further examination can be done after the first approximation on the regular order of the variables, since (1) shows that the approximation function that is calculated has the minimum distance.

After that, Grain-128 is used, whose feedback polynomial of the NFSR is drastically changed compared to the one in Grain-v1. The feedback function has 19 input variables, the degree of the function is 2 and all the variables only appear once in the function. The best approximation function that depends on 17 variables is calculated and approximates $g(x)$ 75%. Moreover, all the other best possible approximations are written down. This is possible due to the form of function $g(x)$ - the explanation is in Proposition 1. A significant pattern is also revealed during the calculation of the approximations that can be used in other Boolean functions that have a similar form. The filter function is composed of 9 variables and the best approximation function that depends on 8 variables is calculated and approximates $h(x)$ 87,5%. Other approximations are also calculated that can lower the number of variables to 2 with a 56,25% approximation.

The last cipher of the Grain family is also the one that offers authentication and it is Grain-128a. The feedback function is the same as in Grain-128 with the addition of three monomials in the polynomial. Proposition 1 is also applicable on this function and the best possible approximations can be calculated, although the use of the technique is not necessary like it is explained in Chapter 5. On the other hand, the filter function remains the same as in Grain-128 and so do the approximations.

For the end, the approximation technique is applied to the output combiner of PALS. The combiner function (h_i) has 9 input variables and $\deg(h_i) = 7$. The approximations functions that are calculated in the regular order of variables in h_i depends on 4 to 8 variables and approximates h_i 53,1 to 75%. This surely doesn't even come close to the upper bound limit of the best approximation that depends on 8 variables, but it needs to be mentioned that no permutation is done to this function. Thus, much better approximations may be possible and could be calculated in a future study.

From the results in Chapter 5, it is proved that the approximation technique can indeed calculate approximation functions of cryptographic value. Immediately, it is safe to state that the approximation technique can be used as a cryptographic criterion to evaluate functions in ciphers. However, it is also necessary for future studies to focus on finding the way to calculate the best possible permutation of variables in a function, so that the full potential of the approximation attack can be achieved each time. The work that is done in this thesis can be a starting point. Except that, future studies can also work on cryptanalytic techniques that can be combined with the approximation technique with the purpose of compromising a cipher.

Bibliography

[1] K. Limniotis and N. Kolokotronis, 'The error linear complexity spectrum as a cryptographic criterion of Boolean Functions', *Submitted to IEEE Trans. Inform. Theory (under review)*.

[2] G. S. Vernam, 'Cipher Printing Telegraph Systems For Secret Wire and Radio Telegraphic Communications', *Transactions of the American Institute of Electrical Engineers*, vol. XLV, pp. 295–301, Jan. 1926.

[3] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*. Berlin: Springer, 2010.

[4] S. W. Golomb, *Shift Register Sequences*. Laguna Hills, CA, USA: Aegean Park Press, 1981.

[5] J. Massey, 'Shift-register synthesis and BCH decoding', *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, Jan. 1969.

[6] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1996.

[7] A. Braeken and J. Lano, 'On the (Im)Possibility of Practical and Secure Nonlinear Filters and Combiners', in *Selected Areas in Cryptography*, 2006, pp. 159–174.

[8] C. De Cannière, 'Trivium: A Stream Cipher Construction Inspired by Block Cipher Design Principles', in *Information Security*, 2006, pp. 171–186.

[9] M. Hell, T. Johansson, and W. Meier, 'Grain: a Stream Cipher for Constrained Environments', *Int. J. Wire. Mob. Comput.*, vol. 2, no. 1, pp. 86–93, May 2007.

[10] H. Fredricksen, 'A Survey of Full Length Nonlinear Shift Register Cycle Algorithms', *SIAM Rev.*, vol. 24, no. 2, pp. 195–221, Apr. 1982.

[11] A. Tsuneda, K. Kudo, D. Yoshioka, and T. Inoue, 'Maximal-Period Sequences Generated by Feedback-Limited Nonlinear Shift Registers', *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E90A, pp. 2079–2084, 2007.

[12] R. A. Rueppel, *New Approaches to Stream Ciphers*. Swiss Federal Institute of Technology Zurich, 1984.

[13] H. Beker and F. Piper, *Cipher systems. The protection of communications*. London, Northwood Books, 1982, 1982.

[14] E. D. Erdmann, 'Empirical Tests of Binary Keystreams', Master's thesis, University of London, 1992.

[15] M. J. B. Robshaw, 'Stream Ciphers', RSA Laboratories, Redwood City, CA, Technical 701, Jul. 1995.

[16] M. Bellare and C. Namprempre, 'Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm', in *Journal of Cryptology*, 2000, vol. 21, pp. 531–545.

[17] J. Katz and M. Yung, 'Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation', in *Fast Software Encryption*, 2001, pp. 284–299.

[18] V. D. Gligor and B. G. Lindsay, 'Object Migration and Authentication', *IEEE Transactions on Software Engineering*, vol. SE-5, no. 6, pp. 607–611, Nov. 1979.

[19] C. Campbell, 'Design and specification of cryptographic capabilities', *IEEE Communications Society Magazine*, vol. 16, no. 6, pp. 15–19, Nov. 1978.

[20] R. R. Jueneman, S. M. Matyas, and C. H. Meyer, 'Message Authentication with Manipulation Detection Code', in *1983 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 1983, pp. 33–33.

[21] M. Bellare and P. Rogaway, 'Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography', in *Advances in Cryptology — ASIACRYPT 2000*, 2000, pp. 317–330.

[22] M. Bellare, T. Kohno, and C. Namprempre, 'Authenticated Encryption in SSH: Provably Fixing the SSH Binary Packet Protocol', in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2002, pp. 1–11.

[23] H. Krawczyk, 'The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)', in *Advances in Cryptology — CRYPTO 2001*, 2001, pp. 310–331.

[24] C. S. Jutla, 'Encryption Modes with Almost Free Message Integrity', in *Advances in Cryptology — EUROCRYPT 2001*, 2001, pp. 529–544.

[25] P. Rogaway, M. Bellare, and J. Black, 'OCB: A Block-cipher Mode of Operation for Efficient Authenticated Encryption', *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 3, pp. 365–403, Aug. 2003.

[26] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, 'UMAC: Fast and Secure Message Authentication', in *Advances in Cryptology — CRYPTO' 99*, 1999, pp. 216–233.

[27] D. Whiting, R. Housley, and N. Ferguson, *Counter with cbc-mac (ccm)*. RFC Editor, 2003.

[28] D. A. Mcgrew and J. Viega, *The Galois/counter mode of operation (GCM)*. 2004.

[29] P. Rogaway, 'Authenticated-encryption with Associated-data', in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, New York, NY, USA, 2002, pp. 98–107.

[30] D. A. McGrew and J. Viega, 'The Security and Performance of the Galois/Counter Mode (GCM) of Operation', in *Progress in Cryptology - INDOCRYPT 2004*, 2005, pp. 343–355.

[31] P. Rogaway, 'Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC', in *Advances in Cryptology - ASIACRYPT 2004*, 2004, pp. 16–31.

[32] P. Sarkar, 'A Simple and Generic Construction of Authenticated Encryption with Associated Data', *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 4, pp. 33:1–33:16, Dec. 2010.

[33] P. Sarkar, 'Modes of operations for encryption and authentication using stream ciphers supporting an initialisation vector', *Cryptography and Communications*, vol. 6, no. 3, pp. 189–231, Sep. 2014.

[34] P. Hawkes and G. Rose, 'Primitive Specification for SOBER-128', Cryptology ePrint Archive, 2003/081, 2003.

[35] N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, S. Lucks, and T. Kohno, 'Helix: Fast Encryption and Authentication in a Single Cryptographic Primitive', in *Fast Software Encryption*, 2003, pp. 330–346.

[36] D. Whiting, B. Schneier, S. Lucks, and F. Muller, 'Phelix-fast encryption and authentication in a single cryptographic primitive', eSTREAM ECRYPT, Stream Cipher Project 2005/020, 2005.

[37] 3GPP Task Force, 'Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification'. ETSI/SAGE, 2011.

[38] M. Ågren, M. Hell, T. Johansson, and W. Meier, 'Grain-128a: a new version of Grain-128 with optional authentication', *International Journal of Wireless and Mobile Computing*, vol. 5, no. 1, pp. 48–59, 2011.

[39] G. Sekar, 'The Stream Cipher Core of the 3GPP Encryption Standard 128-EEA3: Timing Attacks and Countermeasures', in *Information Security and Cryptology*, 2012, pp. 269–288.

[40] D. Watanabe and S. Furuya, 'A MAC Forgery Attack on SOBER-128', in *Fast Software Encryption*, 2004, pp. 472–482.

[41] F. Muller, 'Differential Attacks against the Helix Stream Cipher', in *Fast Software Encryption*, 2004, pp. 94–108.

[42] H. Wu and B. Preneel, 'Differential-Linear Attacks Against the Stream Cipher Phelix', in *Fast Software Encryption*, 2007, pp. 87–100.

[43] D. J. Bernstein, 'The Poly1305-AES Message-Authentication Code', in *Fast Software Encryption*, 2005, pp. 32–49.

[44] H. Wu, 'ACORN: a lightweight authenticated cipher (v3)', *Candidate for the CAESAR Competition. See also https://competitions.cr.yp.to/round3/acornv3.pdf*, 2016.

[45] H. Wu and T. Huang, 'The authenticated cipher MORUS (v1)', *CAESAR submission*, 2014.

[46] R. A. Rueppel, *Analysis and Design of Stream Ciphers*. Berlin, Heidelberg: Springer-Verlag, 1986.

[47] R. Games and A. Chan, 'A fast algorithm for determining the complexity of a binary sequence with period2^n(Corresp.)', *IEEE Transactions on Information Theory*, vol. 29, no. 1, pp. 144–146, Jan. 1983.

[48] K. G. Paterson, 'Perfect maps', *IEEE Transactions on Information Theory*, vol. 40, no. 3, pp. 743–753, May 1994.

[49] T. Etzion, 'Constructions for perfect maps and pseudorandom arrays', *IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 1308–1316, Sep. 1988.

[50] R. A. Rueppel, 'Linear Complexity and Random Sequences', in *Advances in Cryptology — EUROCRYPT' 85*, 1986, pp. 167–188.

[51] M. Stamp and C. F. Martin, 'An algorithm for the k-error linear complexity of binary sequences with period 2/sup n/', *IEEE Transactions on Information Theory*, vol. 39, no. 4, pp. 1398–1401, Jul. 1993.

[52] H. Niederreiter, 'Some Computable Complexity Measures for Binary Sequences', in *Sequences and their Applications*, 1999, pp. 67–78.

[53] K. Kurosawa, F. Sato, T. Sakata, and W. Kishimoto, 'A relationship between linear complexity and k-error linear complexity', *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 694–698, Mar. 2000.

[54] T. Etzion, N. Kalouptsidis, N. Kolokotronis, K. Limniotis, and K. G. Paterson, 'Properties of the Error Linear Complexity Spectrum', *IEEE Transactions on Information Theory*, vol. 55, no. 10, pp. 4681–4686, Oct. 2009.

[55] A. G. B. Lauder and K. G. Paterson, 'Computing the error linear complexity spectrum of a binary sequence of period 2/sup n/', *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 273–280, Jan. 2003.

[56] J. L. Massey, D. J. Costello, and J. Justesen, 'Polynomial weights and code constructions', *IEEE Transactions on Information Theory*, vol. 19, no. 1, pp. 101–110, Jan. 1973.

[57] A. Salagean, 'On the computation of the linear complexity and the k-error linear complexity of binary sequences with period a power of two', *IEEE Transactions on Information Theory*, vol. 51, no. 3, pp. 1145–1150, Mar. 2005.

[58] M. Matsui, 'Linear Cryptanalysis Method for DES Cipher', in *Advances in Cryptology — EUROCRYPT '93*, 1994, pp. 386–397.

[59] C. Ding, G. Xiao, and W. Shan, 'The Stability Theory of Stream Ciphers', in *Lecture Notes in Computer Science*, 1991, vol. 561.

[60] T. W. Cusick and P. Stanica, 'Chapter 7 - Stream Cipher Design', in *Cryptographic Boolean Functions and Applications (Second Edition)*, T. W. Cusick and P. Stanica, Eds. Academic Press, 2017, pp. 143–185.

[61] A. Canteaut, 'Lecture notes on Cryptographic Boolean Functions', *Inria, Paris, France*, 2016.

[62] F. J. MacWilliams and N. J. A. Sloane, *The theory of error correcting codes*. North-Holland, Amsterdam: Elsevier, 1977.

[63] Siegenthaler, 'Decrypting a Class of Stream Ciphers Using Ciphertext Only', *IEEE Transactions on Computers*, vol. C–34, no. 1, pp. 81–85, Jan. 1985.

[64] J. Pieprzyk, 'Non-linearity of Exponent Permutations', in *Advances in Cryptology, Proceedings of EuroCrypt'89, LNCS*, 1989, vol. 434, pp. 80–92.

[65] A. F. Webster and S. E. Tavares, 'On the Design of S-Boxes', in *Advances in Cryptology — CRYPTO '85 Proceedings*, 1986, pp. 523–534.

[66] Babbage, 'On the relevance of the strict avalanche criterion', *Electronics Letters*, vol. 26, no. 7, pp. 461–462, Mar. 1990.

[67] T. W. Cusick and P. Stanica, 'Chapter 3 - Avalanche and Propagation Criteria', in *Cryptographic Boolean Functions and Applications (Second Edition)*, T. W. Cusick and P. Stanica, Eds. Academic Press, 2017, pp. 31–54.

[68] R. A. Rueppel, 'Correlation Immunity and the Summation Generator', in *Advances in Cryptology — CRYPTO '85 Proceedings*, 1986, pp. 260–272.

[69] T. Siegenthaler, 'Correlation-immunity of nonlinear combining functions for cryptographic applications (Corresp.)', *IEEE Transactions on Information Theory*, vol. 30, no. 5, pp. 776–780, Sep. 1984.

[70] A. Canteaut, 'On the correlations between a combining function and functions of fewer variables', in *Proceedings of the IEEE Information Theory Workshop*, 2002, pp. 78–81.

[71] A. Canteaut and M. Videau, 'Symmetric Boolean functions', *IEEE Transactions on Information Theory*, vol. 51, no. 8, pp. 2791–2811, Aug. 2005.

[72] H. Wang, J. Peng, Y. Li, and H. Kan, 'On 2k-Variable Symmetric Boolean Functions With Maximum Algebraic Immunity k', *IEEE Transactions on Information Theory*, vol. 58, no. 8, pp. 5612–5624, Aug. 2012.

[73] O. S. Rothaus, 'On "bent" functions', *Journal of Combinatorial Theory, Series A*, vol. 20, no. 3, pp. 300–305, May 1976.

[74] N. Tokareva, *Bent functions: results and applications to cryptography*. Academic Press, 2015.

[75] A. M. Youssef and G. Gong, 'Hyper-bent Functions', in *Advances in Cryptology — EUROCRYPT 2001*, 2001, pp. 406–419.

[76] M. I. Salam, K. K.-H. Wong, H. Bartlett, L. Simpson, E. Dawson, and J. Pieprzyk, 'Finding State Collisions in the Authenticated Encryption Stream Cipher ACORN', in *Proceedings of the Australasian Computer Science Week Multiconference*, New York, NY, USA, 2016, pp. 36:1–36:10.

[77] I. Dinur and A. Shamir, 'Cube Attacks on Tweakable Black Box Polynomials', in *Advances in Cryptology - EUROCRYPT 2009*, 2009, pp. 278–299.

[78] Y. Todo, T. Isobe, Y. Hao, and W. Meier, 'Cube Attacks on Non-Blackbox Polynomials Based on Division Property', *IEEE Transactions on Computers*, vol. 67, no. 12, pp. 1720–1736, Dec. 2018.

[79] M. I. Salam, H. Bartlett, E. Dawson, J. Pieprzyk, L. Simpson, and K. K.-H. Wong, 'Investigating Cube Attacks on the Authenticated Encryption Stream Cipher ACORN', in *Applications and Techniques in Information Security*, 2016, pp. 15–26.

[80] V. A. Ghafari and H. Hu, 'A new chosen IV statistical distinguishing framework to attack symmetric ciphers, and its application to ACORN-v3 and Grain-128a', *Journal of Ambient Intelligence and Humanized Computing*, Jun. 2018.

[81] E. Filiol, 'A New Statistical Testing for Symmetric Ciphers and Hash Functions', in *Information and Communications Security*, 2002, pp. 342–353.

[82] X. Zhang, X. Feng, and D. Lin, 'Fault Attack on ACORN v3', *The Computer Journal*, vol. 61, no. 8, pp. 1166–1179, May 2018.

[83] E. Biham and A. Shamir, 'Differential fault analysis of secret key cryptosystems', in *Advances in Cryptology — CRYPTO '97*, 1997, pp. 513–525.

[84] J. J. Hoch and A. Shamir, 'Fault Analysis of Stream Ciphers', in *Cryptographic Hardware and Embedded Systems - CHES 2004*, 2004, pp. 240–253.

[85] A. Siddhanti, S. Sarkar, S. Maitra, and A. Chattopadhyay, 'Differential Fault Attack on Grain v1, ACORN v3 and Lizard', in *Security, Privacy, and Applied Cryptography Engineering*, 2017, pp. 247–263.

[86] S. Maitra, A. Siddhanti, and S. Sarkar, 'A Differential Fault Attack on Plantlet', *IEEE Transactions on Computers*, vol. 66, no. 10, pp. 1804–1808, Oct. 2017.

[87] M. Hell, T. Johansson, A. Maximov, and W. Meier, 'A Stream Cipher Proposal: Grain-128', in *2006 IEEE International Symposium on Information Theory*, 2006, pp. 1614–1618.

[88] Y. Lee, K. Jeong, J. Sung, and S. Hong, 'Related-Key Chosen IV Attacks on Grain-v1 and Grain-128', in *Information Security and Privacy*, 2008, pp. 321–335.

[89] O. Küçük, 'Slide Resynchronization Attack on the Initialization of Grain 1.0', eSTREAM-ECRYPT Stream Cipher Project, 2006/044, 2006.

[90] S. Banik, S. Maitra, S. Sarkar, and T. Meltem Sönmez, 'A Chosen IV Related Key Attack on Grain-128a', in *Information Security and Privacy*, 2013, pp. 13–26.

[91] S. Knellwolf, W. Meier, and M. Naya-Plasencia, 'Conditional Differential Cryptanalysis of NLFSR-Based Cryptosystems', in *Advances in Cryptology - ASIACRYPT 2010*, 2010, pp. 130–145.

[92] I. Dinur and A. Shamir, 'Breaking Grain-128 with Dynamic Cube Attacks', in *Fast Software Encryption*, 2011, pp. 167–187.

[93] I. Dinur, T. Güneysu, C. Paar, A. Shamir, and R. Zimmermann, 'An Experimentally Verified Attack on Full Grain-128 Using Dedicated Reconfigurable Hardware', in *Advances in Cryptology – ASIACRYPT 2011*, 2011, pp. 327–343.

[94] S. Banik, S. Maitra, and S. Sarkar, 'A Differential Fault Attack on Grain-128a Using MACs', in *Security, Privacy, and Applied Cryptography Engineering*, 2012, pp. 111–125.

[95] M. Ashouri, *Design of a New Stream Cipher: PALS*. 2018.

Appendix A

Lauder and Paterson Algorithm

The LPA has a significant part in the approximation technique that is analysed and used in this thesis. Thus, the source code of the algorithm that was used is given.

A.1 Source Code

```
/* program to compute CELCS of costed binary sequences */
/* modified to output tree info */
/* restricted to integer costs (but easily modified) */
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#define N 4096/* N is the period of the input sequence */
           /* and the maximum size of any arrays we need */

void celcs(int *s,int *cost, int l, int tsf, int lim, int c);
int min(int a, int b);

  FILE *fp;
  FILE *fp_output;

main()
{
  int i,k;
```

```
  char c;
  int s[N];
  int cost[N];



  fp=fopen("input.txt","r");
  fp_output=fopen("output.txt","w");

  /* input the initial sequence of N bits, setting all costs to 1 */
  for (i=0; i< N; i++)
    {
      c=fgetc(fp);
      if (c=='1')
      s[i]=1;
      else
      s[i]=0;
      cost[i]=1;
    }

  /* now run the celcs algorithm  */
  celcs(s,cost,N,0,N,0);

  fclose(fp);
  fclose(fp_output);

}

void celcs(int *s,int *cost, int l, int tsf, int lim, int c)
{
  int i;
  int L[N]; // N/2
  int R[N];
  int B[N];  // N/2
  int Lcost[N];  // N/2
  int Bcost[N];  //N /2
  int T=0;

  if (l >1)
    {
      /* calculate B(S) and L(S) */

      for (i=0;i < (l/2); i++)
      {
        L[i]=s[i];
        R[i]=s[i+(l/2)];
        B[i]=L[i]^R[i];
      }

      /* calculate costs for B and L, and calculate T */

      for (i=0; i < (l/2); i++)
      {
        Bcost[i]=min(cost[i],cost[i+(l/2)]);
        T+=B[i]*Bcost[i];
      }

      for (i=0; i < (l/2); i++)
      {
        if (B[i]==1)
          {
```

```
            if (cost[i] <= cost[i+(l/2)])
            {
              L[i]=R[i];
              Lcost[i]=cost[i+(l/2)]-cost[i];
            }
            else
            {
              Lcost[i]=cost[i]-cost[i+(l/2)];
            }
          }
        else
          Lcost[i]=cost[i]+cost[i+(l/2)];
      }

      /* output the tree information - omit this if only need spectrum
*/
      fprintf(fp_output,"B(S):");
      for (i=0; i < l/2; i++)
        fprintf(fp_output,"%1d_%1d ",B[i],Bcost[i]);
      fprintf(fp_output,"\n");

      fprintf(fp_output,"L(S):");
      for (i=0; i < l/2; i++)
       fprintf(fp_output,"%1d_%1d ",L[i],Lcost[i]);
      fprintf(fp_output,"\n");

      fprintf(fp_output,"T: %d\n\n",T);


      /* the main decision point in the algorithm */
      if (T > 0)
      {
        fprintf(fp_output,"CELCS(B(S),%d,%d,%d)\n",tsf,min(lim,tsf+T-
1),c+(l/2));
        celcs(B,Bcost,l/2,tsf,tsf+T-1,c+(l/2));
      }
      if (tsf + T <= lim)
      {
        fprintf(fp_output,"CELCS(L(S),%d,%d,%d)\n",tsf+T,lim,c);
        celcs(L,Lcost,l/2,tsf+T,lim,c);
      }
    }
  else
    {
      /* the case l=1 */
      fprintf(fp_output,"s[0]=%d,cost[0]=%d\n",s[0],cost[0]);
      if (s[0]==0)
      fprintf(fp_output,"CP: (%d,%d)\n",tsf,c);
      if ((s[0]==1) && (cost[0] > 0))
      fprintf(fp_output,"CP: (%d,%d)\n",tsf,c+1);
      if ((s[0]==1) && (tsf+cost[0] <= lim))
      fprintf(fp_output,"CP: (%d,%d)\n",tsf+cost[0],c);
      fprintf(fp_output,"\n");
    }
  return;
}


int min(int a, int b)
{
if (a < b)
```

```
  return(a);
else
  return(b);
}
```