EAI Endorsed Transactions
on Energy Web

Research Article **EAI.EU**

Using Hadamard transform for cryptanalysis of pseudo-random generators in stream ciphers

Guillermo Sosa-Gómez[1,*], Omar Rojas[1], Octavio Páez-Osuna[2]

[1]Universidad Panamericana. Escuela de Ciencias Económicas y Empresariales. Álvaro del Portillo 49, Zapopan, Jalisco, 45010, México
[2]Ronin Institute for Independent Scholarship, Montclair, NJ 07042, USA

Abstract

In this work we discuss results obtained from an application of the Hadamard transform to cryptanalysis, and in particular, we determine the probability to decipher different pseudo-random number generators used as components of stream ciphers. Also, we found a relationship between entropy and Hadamard's values.

Received on 02 March 2020; accepted on 09 April 2020; published on 15 April 2020

Keywords: pseudo-random generators, Hadamard transform, stream ciphers, entropy

doi:10.4108/eai.13-7-2018.163980

1. Introduction

Cryptography, or cryptology, deals with the practice and study of securing communications between parties. Thus, it is of the greatest importance nowadays, in particular since we live in a society with digital presence for almost all of our activities and assets, to be able to rely on the methods employed to secure such information. There are plenty of applications, such as in electronic commerce [1, 2] and blockchain [3–5], among others. In particular, stream ciphers, which are the object of our present study are useful for protecting data in real-time, like the encryption and decryption of a DVD [6]. Cryptanalysis is the study of methods to reveal the meaning of encrypted information, without access to the secret key. Typically, this translates into obtaining the key used to encrypt the information. In non-technical terms, this practice is known as breaking or forcing the cryptosystem. Despite the aim being always the same, the methods and techniques of cryptanalysis have changed drastically throughout the history of cryptography, adapting to an increasing cryptographic complexity. The methods of cryptanalysis have also changed as it is no longer possible to have unlimited success in breaking a cryptosystem, and there is a hierarchical classification of what constitutes an attack in practice, see [7]. In this work, we develop theoretical statistical attacks by searching for auto correlations in the output bits of stream ciphers. The term *cryptanalysis* is also used to refer to any attempt to circumvent the security of different types of algorithms and cryptographic protocols in general, and not just encryption [8]. Although the objective has always been the same, *i.e.* totally breaking the cryptosystem, the methods and techniques of cryptanalysis have changed drastically throughout the history of cryptography, adapting to a growing cryptographic complexity, which ranges from the pen and paper methods of the past, through machines like Enigma to the systems based on modern computers and other electronic devices. The purpose of any pseudo random generator (PRNG) is to obtain sequences that behave statistically as if they were random. Although some design criteria are known, how can you decide when a sequence is sufficiently random? How can unpredictability be measured? To overcome the complexities of modern day cryptography we must resort to advanced mathematics and algorithms. A very useful mathematical technique to reduce the complexity of a problem is that of changing its domain by means of a transformation. A transform represents the change from one domain to another, and with the right properties, it reduces the complexity of the given task. In the context of our problem, the cryptanalysis of stream ciphers, the Hadamard transform will prove to be useful. In the domain

*Corresponding author. Email: gsosag@up.edu.mx of random sequences there are those generated by intrinsically random physical processes, *i.e*, based on the assumption of the existence of random processes in nature. In many applications it is necessary to have the same sequence (apparently random) in two different experiments, so it is necessary to use ireproducible deterministic algorithms. Sequences produced by such algorithms are called pseudo-random. Pseudo-random sequences are used in various environments related to telecommunications. One of the most important steps in cryptanalysis is to establish correspondences between the elements of the output with probable clear text. The main task is to find ways of facilitating the computation of a probability that relates several terms of an output, and determine the likelihood to decipher the message or parts of it. In this work, we develop theoretical statistical attacks by searching for autocorrelations in the output bits of stream ciphers [9]. The term *cryptanalysis* is also used to refer to any attempt to circumvent the security of different types of algorithms and cryptographic protocols in general, and not just encryption [8]. Although the objective has always been the same, *i.e.* totally breaking the cryptosystem, the methods and techniques of cryptanalysis have changed drastically throughout the history of cryptography, adapting to a growing degree of cryptographic complexity, which ranges from the pen and paper methods of the past, through machines like Enigma to the systems based on modern computers and other electronic devices. For a general introduction in the subject, see [10, 11]. The purpose of any pseudorandom generator (PRNG) [12] is to obtain sequences that statistically behave as random. Although some design criteria are known, how can one decide whether a sequence is sufficiently random? How can unpredictability be measured? To overcome the complexities of modern day cryptography one must resort to advanced mathematics and algorithms. A very useful mathematical technique to reduce the complexity of a problem is that of changing its domain by means of a transformation. A transform represents the change from one domain to another, and with the right properties, it reduces the complexity of the given task. In the context of our problem, the cryptanalysis of stream ciphers, the Hadamard transform (also known as Walsh-Hadamard transform [13]) will prove to be useful. The structure of the paper is as follows. Section 2 presents the stream ciphers used in the current work Section 3 presents the results related to entropy of the stream ciphers under study. Section 4. Section 5 gives an example of how to apply our proposed method. Finally, Section 8 presents the conclusions of the word and some desirable future work.

2. Stream ciphers

A *stream cipher* algorithm is simply specifying a pseudorandom generator, which allows to encrypt messages of arbitrary length by combining the bit sequence produced with the message by exclusive-OR operation, symbol by symbol [14]. When designing stream ciphers, a number of properties must be taken into account besides generating *randomly looking* bit sequences [15]; the generated sequence must present the most unpredictable behavior possible, i.e., given a fraction of the sequence, it should not be possible to predict the rest, either before or after the given sub-sequence. More formally, a stream cipher can be viewed as a function $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$ which transforms a binary vector input $X = (x_1, \dots, x_n)$ of n bits into a binary vector output $Y = f(X) = (y_1, \dots, y_m)$ of m bits where $n, m \in \mathbb{N}$. Denoted as e_i with $1 \leq i \leq n$, the unit vectors $e_1 = (1, 0, \dots, 0, 0, 0), \dots, e_n = (0, 0, \dots, 0, 0, 1)$. To detect regularities in binary sequences, it is necessary to resort to probabilistic studies allowing for quantitative evaluation of the randomness of a sequence . Among the 15 statistical tests proposed by the NIST in its battery of tests to evaluate PRNG there is there is the so called *approximate entropy test* [16]. We will apply this technique as a measurement for our method. In what follows we briefly describe commonly used stream ciphers which we will also use as a way to illustrate our proposed method and obtained results.

2.1. Shrinking Generator

The so-called Shrinking Generator(SG) is a nonlinear key stream generator composed by two LFSRs [17] so that a control register SRS decimates the sequence produced by the other register SRA. S and A denote respectively their corresponding lengths and fulfill that $(S, A) = 1$ and $S < A$. $P_S(x), P_A(x) \in \mathbb{F}(2)[x]$ denote their corresponding primitive characteristic polynomials. The sequence $\{s_i\}$, produced by SRS, controls the bits of the sequence $\{a_i\}$ produced by SRA which are included in the output shrunken sequence Z according to the following rule: If $s_i = 1$ then $z_j = a_I$, and if $s_i = 0$ then a_i is discarded. As different pairs of SRA/SRS initial states can generate the same shrunken sequence, in the sequel we assume, without loss of generality, that the first term of the sequence $\{s_i\}$ equals 1, that is $s_0 = 1$. According to [17], the period of the shrunken sequence is: $T = (2^A - 1)2^{S-1}$,its linear complexity, notated LC, satisfies the following inequality: $A2^{S-2} < LC \leq A2^{S-1}$. It can be proven [18] that the shrunken sequence has also good distributional statistics. Therefore, due to all these good characteristics, this scheme has been traditionally used as a key stream sequence generator with application in secret-key cryptography. A potential problem is that the shrinking generator outputs bits at an irregular rate, and a timing attack might reveal something about the sampling sequence a unless some sort of buffering conceals this.(see [17])

2.2. A5/1

Stream ciphers are considered nowadays the fastest encryption procedures. Consequently, they are implemented in many practical applications e.g. the algorithms $A5$ in GSM communications (GSM). Cryptosystem $A5/1$ is a synchronous stream encryption that accepts a 64-bit session key $K_s = (k_0, \dots, k_{63}) \in \mathbb{F}(2)^{64}$ and an initialization 22-bit vector $IV = (v_0, \dots, v_{21}) \in \mathbb{F}(2)^{22}$ derived from the 22-bit frame number that is publicly known. It consists of three LFSRs of lengths 19, 22 and 23, called R1, R2 and R3, respectively. These LFSRs are irregularly synchronized. Each of them has a clocking bit C1, C2 and C3, respectively. Each time the LFSRs are registered, the three clock bits C1, C2 and C3 determine which of the LFSRs are synchronized. The output of the three LFSRs are XOR combined. The records are updated according to their primitive polynomials. Then the records produce sequences of maximum periods. There are several kinds of attacks are listed on A5/1 in section 1 in [19]. One of them is the method of Biryukov [20]. He found a known-key stream attack on A5/1 requiring about two seconds of the keystream and recovers Kc in a few minutes on a personal computer. The second one is the method of Barkan [19]. He proposed a cipher-text-only attack on A5/1 that can recover Kc by using only four frames of ciphertext. According to National Institute of Standards and Technology (NIST), all important cryptography tests (Frequency test, Serial test, Run, Long Run test, Poker test, Auto-Correlation test, Maurer's Universal Test) have applied on designed stream cipher algorithm. All of tests passed successfully. [21]

2.3. Blow-fish

Blow-fish [22] is an encryption of the DES type of 16 rounds, with blocks and keys from 64 bits up to 448 bits in length. The computing intensive key expansion phase creates eighteen 32-bit sub keys plus four S boxes of 256 32-bit inputs derived from the input key, for a total of 4168 bytes. Data encryption occurs through a 16-round Feistel network. Each round consists of a permutation dependent on the key and a substitution dependent on the key and the data. All operations are XORs and sums in 32-bit words. The only additional operations are four indexed memory reads per round. Blowfish's weak keys produce bad S-boxes, since Blowfish's S-boxes are key-dependent. There is a chosen plaintext attack against a reduced-round variant of Blowfish that is made easier by the use of weak keys.[23]

3. Entropy of stream ciphers

In information theory, Shannon's entropy or entropy (in honor of Claude E. Shannon)(see [24]) measures the uncertainty of a source of information, that is, entropy is a measure of randomness in information. Cryptographic algorithms are required to introduce high randomness on encrypted messages, with minimum or no dependence between the key and the ciphertext. With high randomness, the relationship between the key and the encrypted text becomes complex. This property is also called confusion. A high degree of confusion is desired to make it difficult to guess the plain text by an attacker. Entropy is thus a reflection of the performance in terms of effectiveness of a cryptographic algorithm. The maximum entropy of a message is reached when all symbols are equally probable. Entropy is computed by means of Shannon's formula

$$H(X) = -\sum_{i=0}^{N-1} p_i \, log_2(p_i)$$

where p_i is the probability of a given symbol and N is the length of message X. The amount of information associated with the simplest event, consisting of only two equiprobable possibilities, will be the unit of measure for entropy and is called bit. This is why the base 2 logarithm is used in the definition of entropy, so that the amount of information from the simplest event is equal to one. It can be said that the entropy of a random variable is the average number of bits needed to encode each of the states of the variable, assuming that each event is expressed using a message written in a binary alphabet.

Table 1. Entropy values for reference cryptosystems for $k = 4, 8, 12, 16$.

Generators	k=4	k=8	k=12	k=16
SG	3.95	7.52	8.27	11.87
A5/1	3.997	7.90	10.21	13.76
Blow-fish	3.999	7.989	10.10	14.35

The entropy of a truly random sequence of length k must be equal to k, in practice, however, source information rarely generates random messages and, in general information entropy value is less than ideal (see for example [25]). However, after messages are encrypted, their entropy should ideally be k. If the output of such encryption emits symbols with entropy less than k, there is a certain degree of predictability, which threatens its security. We conducted entropy tests for A5/1, Blow-fish and Shrinking Generator cryptosystems, results are listed in table ??. It can be observed that as k increases entropy drops from its ideal value as expected and, it does not imply these generators have no practical use.

4. Hadamard transform

The Hadamard Transform is perhaps the best known of non-sinusoidal orthogonal transformation. The Hadamard Transform has gained prominence in applications in the processing of digital signals [26], because it only uses sums and subtractions to compute. Therefore, its implementation in hardware is very efficient. The Hadamard transform proves also useful as a tool in the construction of q-bent functions [27] , and vectorial bent functions [28]. Let $\mathbb{V}_n$ be the vector space of dimension n over the binary Galois field $\mathbb{F}_2$. For two vectors a,$b \in \mathbb{F}_2^n$, we define the scalar product $a \cdot b = (a_1 b_1 \oplus \ldots \oplus a_n b_n)$ and the sum $a \oplus b = (a_1 \oplus b_1, \ldots, a_n \oplus b_n)$, where the product and the sum $\oplus$ (called XOR) are over $\mathbb{F}_2$. We recommend the articles of Bernasconi et al. [29] and Pommerening [30] for more on this topic.

Definition 4.1. The Hadamard transform of a function f on $\mathbb{V}_n$ (with the values of f taken to be real numbers) is the mapping $H(f) : \mathbb{V}_n \to \mathbb{R}$, defined by

$$H(f)(h) = \sum_{x \in \mathbb{V}_n} f(x)(-1)^{h \cdot x}. \tag{1}$$

It can be verified that a direct calculation of the complete Hadamard spectrum using previous definition implies a complexity of N^2 steps, with $N = 2^n$. However, there is a quick procedure to calculate the Hadamard transform that can be computed with only $Nlog(N)$ steps, using the concept of a butterfly diagram(see [31]).

4.1. Hadamard transform in cryptanalysis

Cryptanalysis is the study of methods to reveal the meaning of encrypted information, without access to the secret key. Typically, this translates into obtaining the key used to encrypt the information. In non-technical terms, this practice is known as breaking or forcing the cryptosystem; we will not discuss the details. Despite that the aim has always been the same, the methods and techniques of cryptanalysis have changed drastically throughout the history of cryptography, adapting to an increasing cryptographic complexity. The methods of cryptanalysis have also changed as it is no longer possible to have unlimited success in breaking a cryptosystem, and there is a hierarchical classification of what constitutes an attack in practice (see [7]). In the domain of random sequences there are those generated by intrinsically random physical processes, *i.e*, based on the assumption of the existence of random processes in nature. In many applications it is necessary to have the same sequence (apparently random) in two different experiments, so it is necessary to use ireproducible deterministic algorithms. Sequences produced by such algorithms are called pseudo-random. Pseudo-random sequences are used in various environments related to telecommunications. One of the most important steps in cryptanalysis is to establish correspondences between the elements of the output with probable clear text. The main task is to find ways of facilitating the computation of a probability that relates several terms of an output, and determine the likelihood to decipher the message or parts of it. Let $Z = \{z_1, z_2, \ldots, z_p\}$ be the output of a binary generator, and $z_w^j = (z_{j+wi-w})_{i=1}^k \in \mathbb{V}_k$ a rolling sequence with a window of size w of Z for all $j \in J = \{1, \ldots, p - wk + w\}$ and $\xi = (\xi_1, \ldots, \xi_k) \in \mathbb{V}_k$ for a fixed k. In our analysis, we want to find the probability $P\{z_w^j \cdot \xi = 0\}$ for $\xi \neq 0$. Our main goal is the computation of this probability, and thus we start by defining $n_0 =| \{j \in J : z_w^j \cdot \xi = 0\} |$, and $n_1 =| \{j \in J : z_w^j \cdot \xi = 1\} |$, such that $n_0 + n_1 = N := p - wk + w$. Then, by a classical result in probability, it follows that

$$P\{z_w^j \cdot \xi = 0\} = \frac{n_0}{N}, \tag{2}$$

for all $j \in J$. This probability depends of ξ. Now, before we proceed to apply certain transformations, we let

$$\begin{aligned}\tfrac{1}{N}\textstyle\sum_{i=1}^N (-1)^{z_w^j \cdot \xi} &= \tfrac{1}{N}\left(\textstyle\sum_{z_w^j \cdot \xi = 0} (-1)^{z_w^j \cdot \xi} + \sum_{z_w^j \cdot \xi = 1} (-1)^{z_w^j \cdot \xi}\right)\\ &= \tfrac{1}{N}(n_0 - n_1) = \Delta_\xi.\end{aligned} \tag{3}$$

Then,

$$\Delta_\xi = \frac{n_0 - n_1}{n_0 + n_1}. \tag{4}$$

The problem is, given Δ_ξ, to find the values of n_0 and n_1, which leads to a Diophantine equation that due to the conditions of the previous variables makes it necessary to resort to resources such as Kronecker's Theorem, which makes the solution of the problem very complicated. The idea then is to use a transform that as it is known represents the change from one domain to another and that due to its properties, it reduces the complexity of mathematical problems. This type of tool has been very useful and fundamental in solving problems in different fields, and of diverse nature. The Hadamard transform allows us to solve the problem posed. Thus, we transform the problem using the Hadamard Transform. It turns out that by finding a relationship between Δ_ξ, n_0 and n_1:

$$1 + \Delta_\xi = 1 + \frac{n_0 - n_1}{n_0 + n_1} = \frac{2n_0}{n_0 + n_1}. \tag{5}$$

Then, $\frac{1+\Delta_\xi}{2} = \frac{n_0}{N}$. The probability that we want to find is

$$P\{z_w^j \cdot \xi = 0\} = \frac{n_0}{N} = \frac{1 + \Delta_\xi}{2}, \tag{6}$$

reducing the problem to that of finding Δ_ξ.

4.2. Hadamard transform of a sequence

Suppose now that we obtain an output sequence Z, we want to count the repetitions of z_w^j, for this we build the set $\alpha = \{\alpha^l \in \mathbb{V}_k\}$, for $l = 1, \ldots, 2^k$ and we use the Alg. 1 . Then we compute the following

```
Data: $z_w^j$
Result: Counter $z_w^j$
for $l = 1$ to $2^k$ do
    $n_l = 0$;
    for $j = 1$ to $N$ do
        if $z_w^j = \alpha^l$ then
            $n_l = n_l + 1$;
            break;
        end
    end
end
```

Algorithm 1: Counter

$$\Delta_\xi = \frac{1}{N}\sum_{j=1}^{N}(-1)^{z_w^j \cdot \xi} = \frac{1}{N}\sum_{\alpha^l \in \alpha}(-1)^{\alpha^l \cdot \xi} \cdot n_l, \tag{7}$$

for all $\xi \in \mathbb{V}_k$. Each z_w^j is mapped to $y^j \in \mathbb{N}$ such that

$$y^j = \sum_{i=0}^{k-1} z_i^j \cdot 2^i, \tag{8}$$

for all $j \in J$, then we let $\Pi[y^j] = \Pi[y^j] + 1$, this is a counter for the number of y^j's such that $n_l = \Pi[l]$. Now let $\hat{n}_\xi = \sum_{\alpha^l \in \alpha}(-1)^{\alpha^l \cdot \xi} n_l$. Then to compute $\{\hat{n}_\xi, \xi \in \mathbb{V}_k\}$ and solve (3), we will use the Discrete Hadamard Transform. If we write $n_l = f(x)$ and to $(h, x) = (\alpha^l, \xi)$, then the transform can be written as

$$H(n_l)(\xi) = \sum_{\alpha^l \in \alpha} n_l(-1)^{\alpha^l \cdot \xi}, \tag{9}$$

arriving to

$$\Delta_\xi = \frac{\hat{n}_\xi}{N} = \frac{H(n_l)(\xi)}{N}, \tag{10}$$

which is a less cumbersome computation of Δ_ξ. Once Δ_ξ is obtained, we can determine if there are distinguished elements within $\mathbb{V}_k$ that can help us characterize certain parts of the output sequence Z or in this case by finding a certain relationship between the elements of the sequence.

5. Example

In this example, we illustrate our methods by analyzing the A5/1 described in the section 2.2. We use Key: 1223456789$ABCDEF$. We organize the output Z sequentially in blocks of length $k = 12$ and $w = 12$, and interpret these blocks as the sequences z_w^j. It has therefore length $p = 16380$. Each z_w^j is the binary expression of an integer in the interval $[0, 2^{12} - 1]$. We store their frequencies in vector Π of length 2^{12}; i.e., $\Pi_i = |\{r | i = \sum_{j=0}^{11} z_{12r+j} 2^j, r = 0, \ldots, 16380\}|$. We now resort to Hadamard matrices to transform our problem. Recall that we may construct Hadamard matrices to use Paley construction(see [32]) In our example, $N = p/k = 1365$ and define $\delta := \frac{H_{12} \cdot \Pi}{N}$ and $\delta_0 := 0$. In this experiment we find that $max(\delta) = 0.115$, corresponding to the 2195-th entry. Now $2195_{10} = 100010010011_2$, and with probability $\frac{1+\delta_{2195}}{2} = 0.5575$ which implies that, with high probability, 2195 is a distinguished element from the output. Indeed, we may verify this by computing

$$\frac{\sum_{i=0}^{N}(-1)^{z_{12i}+z_{12i+4}+z_{12i+8}+z_{12i+10}+z_{12i+11}}}{N} = 0.115 = \delta_{2195}, \tag{11}$$

which tells us that our computation is correct.

6. Hadamard values for A5/1, Blow-fish and SG

Using the example given in the previous section as a prototype, we tested our method with other well known generators such as $A5/1$, Blow-fish and SG. Table 2 summarizes the Hadamard values computed using 1000 rounds for each generator. Our method is able to identify distinguished elements on each generator for different values of k.

Table 2. Hadamard values for 1000 rounds for $A5/1$, Blow-fish and SG generators.

Generators	$k = 4$	$k = 8$	$k = 12$	$k = 16$
SG	0.515	0.616	0.618	0.748
A5/1	0.514	0.532	0.548	0.564
Blow − fish	0.514	0.531	0.549	0.566

7. Correlation analysis between entropy and Hadamard values

We proceeded to apply Pearson's p-test to determine correlation coefficients for entropy and Hadamard values. summarizes our findings. In all cases a strong correlation was obtained. It can be observe that the entropy values and the Hadamard values are strongly correlated. As the step size increases, the entropy value decreases and Hadamard values increase. No further exploration of this data will be conducted in the present work.

Table 3. Pearson's p-test results for Hadamard values and entropy correlation coefficients.

		SG	A5/1	BF
$k = 4$	r	-0.802	-0.597	-0.582
	p-value	$1.1\ 10^{-221}$	$5.4\ 10^{-98}$	$9.8\ 10^{-39}$
$k = 8$	r	-0.532	-0.292	-0.341
	p-value	$8.9\ 10^{-73}$	$4.2\ 10^{-21}$	$1.5\ 10^{-12}$
$k = 12$	r	-0.455	-0.069	-0.0727
	p-value	$2.5\ 10^{-51}$	0.028	0.027
$k = 16$	r	-0.411	-0.999	-0.685
	p-value	0.027	0.0012	0.041

8. Conclusions

We presented a method for the cryptanalysis of pseudorandom generators in a stream cipher. The method relies on the discrete Hadamard transform to map the problem from a statistical setting to a probabilistic domain, in which it is possible, with high probability, to single out elements of the output and pair them with their corresponding clear texts. We included a complete example as a prototype to illustrate the simplicity of our method and its accuracy. We summarized statistical results for the computation of Hadamard values for $A5/1$ and Blow-fish generators. We included results of a correlation analysis for entropy and Hadamard values indicating a strong correlation which we will explore in a future work.

References

[1] Ford, W. and Baum, M.S. (2000) *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption* (Upper Saddle River, NJ: Prentice Hall PTR).

[2] Torrubia, A., Francisco, J.M. and Marti, L. (2001) Cryptography regulations for e-commerce and digital rights management. *Computers & Security* **20**(8): 724–738.

[3] Kosba, A., Miller, A., Shi, E., Wen, Z. and Papamanthou, C. (2016) Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE symposium on security and privacy (SP)* (IEEE): 839–858.

[4] Nofer, M., Gomber, P., Hinz, O. and Schiereck, D. (2017) Blockchain. *Business & Information Systems Engineering* **59**(3): 183–187.

[5] Henry, R., Herzberg, A. and Kate, A. (2018) Blockchain access privacy: Challenges and directions. *IEEE Security & Privacy* **16**(4): 38–45.

[6] Venkatesulu, M. and Ravi, M. (2016) A stream cipher for real time applications. In *International Conference on Theoretical Computer Science and Discrete Mathematics* (Springer): 453–456.

[7] Katz, J. and Lindell, Y. (2014) *Introduction to Modern Cryptography* (Chapman and Hall/CRC).

[8] Schneier, B. (2000) A self-study course in block-cipher cryptanalysis. *Cryptologia* **24**(1): 18–33.

[9] Rueppel, R.A. (1986) Stream ciphers. In *Analysis and Design of Stream Ciphers* (Springer), 5–16.

[10] Sinkov, A. and Feil, T. (2009) *Elementary Cryptanalysis*, **22** (Washington: MAA).

[11] Dooley, J.F. (2018) *History of Cryptography and Cryptanalysis* (Cham: Springer).

[12] Håstad, J., Impagliazzo, R., Levin, L.A. and Luby, M. (1999) A pseudorandom generator from any one-way function. *SIAM Journal on Computing* **28**(4): 1364–1396.

[13] Beer, T. (1981) Walsh transforms. *American Journal of Physics* **49**(5): 466–472.

[14] Jiao, L., Hao, Y. and Feng, D. (2020) Stream cipher designs: a review. *Science China Information Sciences* **63**(3): 1–25.

[15] Ibáñez, M.S. and Díaz, R.G. (1999) *Generación y Análisis de Secuencias Pseudoaleatorias* (Edicions UPC).

[16] Cover, T.M. and Thomas, J.A. (2012) *Elements of Information Theory* (John Wiley & Sons).

[17] Coppersmith, D., Krawczyk, H. and Mansour, Y. (1993) The shrinking generator. In *Annual International Cryptology Conference* (Springer): 22–39.

[18] Shparlinski, I. (2001) On the linear complexity of the power generator. *Designs, Codes and Cryptography* **23**(1): 5–10.

[19] Barkan, E., Biham, E. and Keller, N. (2003) Instant ciphertext-only cryptanalysis of gsm encrypted communication. In *Annual international cryptology conference* (Springer): 600–616.

[20] Biryukov, A., Shamir, A. and Wagner, D. (2000) Real time cryptanalysis of a5/1 on a pc. In *International Workshop on Fast Software Encryption* (Springer): 1–18.

[21] Bakhtiari, M. and Maarof, M.A. (2011) An efficient stream cipher algorithm for data encryption. *International Journal of Computer Science Issues (IJCSI)* **8**(3): 247.

[22] Schneier, B. (1994) Description of a new variable-length key, 64-bit block cipher (blowfish). In Anderson, R. [ed.] *Fast Software Encryption* (Berlin, Heidelberg: Springer Berlin Heidelberg): 191–204.

[23] Chaudhari, M.P. and Patel, S.R. (2014) A survey on cryptography algorithms. *International Journal of Advance Research in Computer Science and Management Studies* **2**(3).

[24] Shannon, C.E. (1948) A mathematical theory of communication. *Bell system technical journal* **27**(3): 379–423.

[25] Tsuneda, A. (2019) Orthogonal chaotic binary sequences based on bernoulli map and walsh functions. *Entropy* **21**(10): 930.

[26] Cariow, A., Majorkowska-Mech, D., Papliński, J.P. and Cariowa, G. (2019) A new fast algorithm for discrete fractional hadamard transform. *IEEE Transactions on Circuits and Systems I: Regular Papers* **66**(7): 2584–2592.

[27] Chen, Z, G.T. and Klapper, A. (2019) On the q-bentness of boolean functions. *Designs, Codes and Cryptography* **87**(1): 163–171.

[28] Carle, C. and Mesnager, S. (2010) On the construction of bent vectorial functions. *International Journal of Information and Coding Theory* **1**(2): 133–148.

[29] Bernasconi, A., Codenotti, B. and Simon, J. (1996) On the fourier analysis of boolean functions. *preprint* : 1–24.

[30] Pommerening, K. (2005), Fourier analysis of boolean maps–a tutorial–, Johannes-Gutenberg-Universitaet working paper.

[31] Cooley, J.W. and Tukey, J.W. (1965) An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation* **19**(90): 297. doi:10.2307/2003354, URL https://www.jstor.org/stable/2003354.

[32] Van Lint, J.H., Wilson, R.M. and Wilson, R.M. (2001) *A Course in Combinatorics* (Cambridge University Press).