On the symmetry of finite pseudorandom binary sequences

by

Balázs Sziklai

Submitted to

Central European University

Department of Mathematics and its Applications

In partial fulfillment of the requirements for the degree of Master of Science

Supervisor: Katalin Gyarmati

Budapest, Hungary

2010

Table of Contents

Chapter 1

Introduction

"Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin." János Neumann

Nowadays it is a popular statement that we live in an information society. Several definitions can be found on the internet about this concept. Perhaps the most straightforward is: "a post-industrial society in which information technology (IT) is transforming every aspect of cultural, political, and social life and which is based on the production and distribution of information" [2]. In this sense it is safe to say that the whole world heading toward this state. Since information is so important it is evident that the security of communication (i.e. exchange of information) is a crucial task. Cryptography (or cryptology; from Greek, *kryptos*, "hidden, secret"; *gráphō*, "I write"; and *logos*, "word, reason") emerged from this need during the second World War. Although in real life cryptographical problems appear mostly as a (computer) engineering task, the importance of security demands mathematical evidence of soundness. As a result today cryptography is becoming more and more an application of mathematics.

Vast majority of protocols applied in cryptography revolves around randomness. Random numbers are indispensable in encrypting messages securely. A well known example is the one-time pad introduced and patented by Vernon in 1917. In this basic encryption scheme the message is first converted to a binary string then added to a truly random string of the same size bitwise modulo 2 (XOR function). Shannon proved that the one-time pad has the property he called *perfect secrecy* [4]. The encrypted message (i.e. the ciphertext) provides no information about the original message to the cryptanalyst apart from its length. Even with infinite computational power it is impossible to retrieve the original message. Loosely speaking since the ciphertext appears random every decryption is equally probable. Naturally with the help of the random string (called 'the pad') the message can be recovered.

For example, suppose we sit in the military headquarters and we would like send the message "charge" to our troops. Assume two pads of paper containing identical random sequences of letters were somehow previously produced and securely issued to the valid parties. We choose the first appropriate unused page from the pad. The material on the selected sheet is the *key* for this message. Each letter from the pad will be combined in a predetermined way with one letter of the message. For example we can assign each letter a numerical value: "A" is 0, "B" is 1 and so on. The technique is to combine the key and the message using modular addition. The numerical values of the corresponding message and key letters are added together, modulo 26. Suppose the key material begins with the letters ULGHIX, then the coding would be done as follows:

	C	H	A	R	G	E
	2	7	0	17	6	4
	U	L	G	H	I	X
$\oplus$	20	11	6	7	8	23
	W	**S**	**G**	**Y**	**O**	**B**
	22	18	6	24	14	1

Anyone can successfully decrypt the message by subtracting the key from the cipher-text modulo 26. Since the key was generated randomly no one can guess what the original message was . Brute force attacks will find that the word ULGHIX produce the plaintext 'charge' but they will also find that PEVVAO produces 'hold on' and SAEYZX produces 'escape'. No adversary can possibly choose from these equally plausible messages.

	W	S	G	Y	O	B	W	S	G	Y	O	B	W	S	G	Y	O	B
	22	18	6	24	14	1	22	18	6	24	14	1	22	18	6	24	14	1
	U	L	G	H	I	X	S	A	E	Y	Z	X	P	E	V	V	A	O
$\ominus$	20	11	6	7	8	23	18	0	4	24	25	23	15	4	21	21	0	4
	C	**H**	**A**	**R**	**G**	**E**	**E**	**S**	**C**	**A**	**P**	**E**	**H**	**O**	**L**	**D**	**O**	**N**
	2	7	0	17	6	4	4	18	2	0	15	4	7	14	11	3	14	13

Despite its security, the one time pad has serious drawbacks in practice. To achieve perfect secrecy, a real random string should be generated and distributed among the valid parties. Furthermore the pad should be at least as long as the message, which is really impractical when the message itself is long. Instead of using real random numbers cryptographers turned toward algorithms which produce numbers close enough to random - so called pseudorandom. The idea is to stretch a short truly random seed into a long pseudorandom one.

Definition 1. *A pseudorandom number generator (PRNG) is a deterministic algorithm, which given a truly random string of length k, outputs a string of length $l \gg k$ which 'appears' to be random. The input of the PRNG is called the seed, while the output of the PRNG is called the pseudorandom number.*

We stress that the output of a PRNG is *not* random. To see this, consider a pseudorandom number generator G which converts an n long binary string into a $2n$ long one. The uniform distribution over $\{0,1\}^{2n}$ is characterized by the fact that each of the 2^{2n} possible strings is chosen with probability exactly 2^{-2n}. In contrast, consider the distribution generated by our PRNG. Since G receives an input of length n, the number of different possible strings in its range is at most 2^n. Thus the probability of a random string of length $2n$ is in the range of G is at most $2^n/2^{2n} = 2^{-n}$. That is, most strings of length $2n$ do not occur as an output of G.

This in particular means that it is trivial to distinguish between the uniform random distribution and a pseudorandom generator, given an *unlimited amount of time*. Thus a brute force attack can be effective against PRNGs. Note that we are implicitly applying Kerchoffs' principle. This is a widely used concept in cryptography which states that the encryption scheme itself should *not* be kept secure, so only the key should constitute the secret information shared by the communicating parties.

Consider that in the above example the pad was not generated randomly but by a pseudorandom generator G. It is possible that the only valid English six letter word in Gs range is 'charge'. The adversary is well aware of G's imperfection due to Kerckhoffs' principle, hence the encryption scheme is far from secure. To avoid this situation the seed must be chosen uniformly at random. Furthermore it should be sufficiently large so that a search over the total possible inputs is infeasible for an adversary.

Definition 2. *A pseudorandom number generator is said to pass all polynomial-time statistical tests if no polynomial-time algorithm can correctly distinguish between an output sequence of the generator and a truly random sequence of the same length with probability significantly greater that $\frac{1}{2}$.*

The above definition says nothing about the parameters of the polynomial in the "polynomial-time algorithm". There is no restrictions on the degree nor on the coefficient of the polynomial. Consequently it makes only sense in an asymptotic way. Every pseudorandom number generator incorporates a security parameter, which is an integer n. The running time for the adversary as well as the adversary's success probability are all viewed as functions of n. Increasing the security parameter will increase the running time for the adversary and decrease the probability of its success. Therefore for every polynomial time adversary there exist a security parameter for which the probability of success is only negligible. The asymptotic approach is rooted in complexity theory and widely used in cryptography.

The use of pseudorandom numbers are not limited to cryptography. They play an important role in many other fields of applied mathematics, in particular in the problems of statistics or numerical analysis. Without the intent of completeness, here are a few examples, where pseudorandom numbers are used:

- Simulation. When a computer is being used to simulate natural phenomena, random numbers are required to make things realistic.
- Sampling. It is often impractical to examine all possible cases, but a random sample will provide insight into what constitutes 'typical' behavior.
- Numerical analysis. Ingenious techniques for solving complicated numerical problems have been devised using random numbers.
- Computer programming. Random values make a good source of data for testing the effectiveness of computer algorithms.

Nevertheless the cryptographical application is by far the most important one. We send emails, transfer money, use wireless internet connection every day. The security of all these applications is granted by the results of modern cryptography.

As the above list demonstrates, pseudorandom numbers are used extensively in a large variety of applications. For different tasks different kind of PRNGs are implemented. For instance the *linear congruential method* is commonly used for simulation purposes and probabilistic algorithms. However it is predictable and hence entirely insecure for cryptographic purposes. In order to gain confidence that a generator is adequate for its task, it should be subjected to a spectrum of statistical tests designed to detect the specific characteristics expected of random sequences.

Many misinterpret the quote of Neumann which we cited at the beginning of this chapter. He was definitely not against the use of pseudorandom numbers. In truly he was cautioning on mistaking the pseudorandom number generators for being truly 'random'. He also suggested one of the first algorithms to produce pseudorandom numbers. We will discuss the squaring method in details later. Neumann also stated, that in his experience "... it was more trouble to test random sequences, than to manufacture them.". The fact that the US based National Institute of Standards and Technology (NIST) registers more than 200 statistical tests whose aim is to ensure the good quality of PRNGs supports this statement [17].

Another reason why statistical tests are important is that there is no evidence that there exist an algorithm which satisfies both definition 1 and 2. We will come back to this question later.

There is a nice summary about the most commonly used statistical tests in the famous book of Menezes, van Oorschot, and Vanstone [1]. The first tests were designed to be *a posteriori* tests, meaning they examined the output of a PRNG and not the random number generator itself. Whenever a sequence passed these tests it was considered pseudorandom. On the other hand the aim of *a priori* or "theoretical" testing is to say something on the distribution of our PRNG. In 1996 Mauduit and Sárközy introduced new measures of pseudorandomness for finite binary strings [13].

We write

$$E_N = \{e_1, e_2, \ldots, e_N\} \in \{-1, +1\}^N.$$

We define the *well-distribution measure* by

$$W(E_N) = \max_{a,b,t} |U(E_N, t, a, b)| = \max_{a,b,t} \left| \sum_{j=0}^{t} e_{a+jb} \right|.$$

where the maximum is taken over all a, b, t such that $a \in \mathbb{Z}$, $b, t \in \mathbb{N}$ and $1 \leq a + b \leq a + tb \leq N$. While the *correlation measure* of order k is

$$C_k(E_N) = \max_{M,D} |V(E_N, M, D)| = \max_{M,D} \left| \sum_{n=1}^{M} e_{n+d_1} \ldots e_{n+d_k} \right|,$$

where $D = (d_1, \ldots, d_k) \in \mathbb{N}^k$, $0 \leq d_1 \leq \cdots \leq d_k$ and the maximum is taken over all D and M such that $M + d_k \leq N$.

Beyond the above mentioned measures Mauduit and Sárközy also studied the Legendre symbol as a natural candidate for producing pseudorandom sequences. They introduced the following construction: for an arbitrary prime p write

$$e_n = \left(\frac{n}{p}\right), \quad E_{p-1} = \{e_1, \ldots, e_{p-1}\}. \tag{1.1}$$

It can be shown that both the well-distribution and correlation measure of E_{p-1} is small. This construction can be further extended. Goubin, Mauduit and Sárközy constructed large families of pseudorandom sequences by replacing $f(x)$ for $1 \leq x \leq p-1$ in place of n in (1.1) (see [15]). Still E_{p-1} has one bad feature which makes it unsuitable for some applications. Namely, if $p = 4k+1$ for some integer k then $\left(\frac{a}{p}\right) = \left(\frac{p-a}{p}\right)$ and for $p = 4k+3$, $\left(\frac{a}{p}\right) = -\left(\frac{p-a}{p}\right)$ making E_{p-1} completely symmetric. To avoid this situation Gyarmati [3] introduced the symmetry measure:

$$S(E_N) = \max_{a<b} |H(E_N, a, b)| = \max_{a<b} \left| \sum_{j=0}^{[\frac{b-a}{2}]-1} e_{a+j} e_{b-j} \right|.$$

She also proved that the first $\frac{p-1}{2}$ elements of E_{p-1} have small symmetry measure. The symmetry property of finite binary sequences can be further studied. Every sequence $\{e_1, e_2, \ldots, e_N\} \in \{-1, +1\}^N$ contains a large symmetrical subset since both $\{e_i : \; e_i = e_{N-i}\}$ and $\{e_i : \; e_i = -e_{N-i}\}$ are symmetrical and one of them is large. This implies that symmetrical patterns can occur quite frequently. Among the possible forms of these patterns only intervals have been studied yet. In general we do not want to consider any binary sequence which shows some kind of symmetrical pattern as pseudorandom.

My aim is to generalize the symmetry measure introduced by Gyarmati. I will examine the two most basic concepts - multiple symmetry centers and arithmetically symmetric tails. I will give upper and lower bounds for the generalized measures and also show some examples which demonstrate that these generalizations are indeed necessary. I will also try to give an insight to the basic mathematical tools and notations of this field. In order to do this, I will discuss the concept of pseudorandom number generators and their testing in details.

First, in Chapter 2 I will describe the notion of randomness from different perspectives. In the next chapter I will introduce some pseudorandom number generators which are historically important. In Chapter 4 I will show how statistical tests work in practice. Finally in the last two chapters I will present my results and the conclusions.

Chapter 2

Randomness and random sequences

"You should call it entropy, for two reasons. In the first place your uncertainty function has been used in statistical mechanics under that name, so it already has a name. In the second place, and more important, no one really knows what entropy really is, so in a debate you will always have the advantage." János Neumann

The first comprehensive work on pseudorandom numbers was written by Knuth in his remarkable book, The Art of Computer Programming [5]. *American Scientist* has included this work among "100 or so Books that shaped a Century of Science", referring to the 20th century, and within the computer science community it is regarded as the first and still the best comprehensive treatment of its subject. The book consist of seven volumes from which only four have been published to date. The first half of the second volume is dedicated completely to random numbers. On the first 30 page Knuth describes some basic pseudorandom number generators, on the next 100 pages he shows how to test them. Only at the end of the chapter does he ask: "What is a random sequence?" and spends another 30 pages to build up a meaningful definition. We shall proceed in a reverse order and establish the notion of randomness before moving forward to the applications.

2.1 Intuitive notion of randomness

We can think about randomness as lack of information. The result of a coin-toss is random because we are unable to measure all the relevant factors (spinning, air resistance, etc.) which determine the orbit of the coin. The above phenomenon (i.e. unpredictability) is referred as *entropy* or *Shannon entropy* in information theory. Entropy quantifies, in the sense of an expected value, the information contained in a message, usually in units such as bits. A fair coin has an entropy of one bit. In comparison, tossing a coin which has heads on both sides has entropy of 0, since we have full information on the outcome. Similarly a random string of letters like "vdaphnchorwxbdte" has high entropy, but human language has low one. For instance in English language the letter 'q' is almost always followed by the letter 'u'. Hence it would be possible to guess the next letter (or even word) given some part of the text.

From philosophical point of view we can ask the question, does randomness exist in the real world? Or is it our inability to compute the complex factors which produces the random phenomena? If the second assumption is true, we may experience less and less randomness as our computational power grows. Then again maybe 'random events' are those for which we need a computer proportional to the size of the universe to calculate their result. They are not random in the idealistic sense of the word, but rather uncomputable. The whole world could be predestined like a giant clockwork. However there are signs that support that real randomness actually exist. In quantum mechanics, the *Heisenberg uncertainty principle* states that certain pairs of physical properties, like position and momentum, cannot be both known to arbitrary precision. According to some interpretations this is not a statement about the limitations of a researcher's ability to measure particular quantities of a system, but rather a statement about the nature of the system itself. Consequently there are things which are uncomputable regardless of our computational power - at least at subatomic level.

Although such a debate is thought provoking, it brings us very far from our original goal. Therefore we end our philosophical detour here and continue with the notion of random sequences.

2.2 Normality and equidistribution

We will now follow Knuth's footsteps[5] and try to develop a definition for random sequences. Consider the following real sequence:

$$\langle U_n \rangle = U_0, U_1, U_2, \dots \tag{2.1}$$

Let u and v be real numbers, such that $0 \leq u < v \leq 1$. If U is a random variable that is uniformly distributed between 0 and 1, the probability that $u \leq U < v$ is equal to $v - u$. For example the probability that U falls in $[1/3, 1/2]$ is $1/6$. We would like to extend this property for a sequence of random numbers. Since we are talking about an infinite set we have to formulate it as an asymptotic condition. An obvious way is to count how many times U_n lies between u and v, and the average number of times should equal $v - u$. Let $k(n)$ be the number of values of j, $0 \leq j < n$, such that $u \leq U_j < v$; we want the ratio $k(n)/n$ to approach the value $v - u$ as n approaches infinity:

$$\lim_{n\to\infty} \frac{k(n)}{n} = v - u \tag{2.2}$$

Furthermore we say that $\widehat{Pr}(u \leq U_n < v) = \lambda$ if $\lim_{n\to\infty} k(n)/n = \lambda$. Now we can formulate our definition:

Definition 3. *The sequence $U_0, U_1, \ldots$ is said to be equidistributed if $Pr(u \leq U_n < v) = v - u$ for all choices of $0 \leq u < v \leq 1$.*

Naturally we expect from any infinite random sequence to be equidistributed. On the other hand it is far from being a sufficient condition of randomness. For example, let $U_0, U_1, \ldots$ and $V_0, V_1, \ldots$ be two equidistributed sequences. It is not hard to see that the sequence

$$W_0,\ W_1,\ W_2,\ W_3, \cdots = \frac{1}{2}U_0,\ \frac{1}{2} + \frac{1}{2}V_0,\ \frac{1}{2}U_1,\ \frac{1}{2} + \frac{1}{2}V_1, \ldots$$

is also equidistributed, since the subsequence $\frac{1}{2}U_0, \frac{1}{2}U_1$ is equidistributed between 0 and 1/2 while the alternate terms $\frac{1}{2} + \frac{1}{2}V_0, \frac{1}{2} + \frac{1}{2}V_1$ are equidistributed between 1/2 and 1. In the sequence of W's, a value less than 1/2 is always followed by a value greater than or equal to 1/2, and conversely; hence the sequence is not random by any reasonable definition. A stronger property than equidistribution is needed.

A natural generalization of the equidistribution property, which removes the objection stated in the preceding paragraph, is to consider adjacent pairs of numbers of our sequence. We can require the sequence to satisfy the condition

$$\widehat{Pr}(u_1 \leq U_n < v_1 \ \ \textit{and} \ \ u_2 \leq U_n + 1 < v_2) = (v_1 - u_1)(v_2 - u_2)$$

for all choices of u_1, u_2, v_1, v_2, such that $0 \leq u_1 < v_1 \leq 1$ and $0 \leq u_2 < v_2 \leq 1$. In general, for any positive integer k we can require our sequence to be *k-distributed* in the following sense:

Definition 4. *The sequence in* (2.1) *is said to be k-distributed if*

$$\widehat{Pr}(u_1 \leq U_n < v_1, \ldots, u_k \leq U_{n+k-1} < v_k) = (v_1 - u_1) \ldots (v_k - u_k)$$

for all choices of numbers u_j, v_j such that $0 \leq u_j < v_j \leq 1$, for $0 \leq j \leq k$.

In other words an equidistributed sequence is a *1-distributed* sequence. Note that if $k > 1$, a *k-distributed* sequence is always *(k - 1)-distributed*, since we may set $u_k = 0$ and $v_k = 1$ in (2.2). At this point where we can formulate a definition which will hopefully bring us just a step away from a satisfactory condition of randomness:

Definition 5. *A sequence is said to be ∞-distributed if it is k-distributed for every positive integer k.*

It is indeed a strong although not a sufficient condition for randomness. To see this first let us consider integer valued sequences. A sequence $\langle X_n \rangle = X_0, X_1, X_2, \ldots$ is called a 'b-ary' sequence if $X_j \in \{0, 1, \ldots, b-1\}$ for every $j \in \mathbb{N}$. Thus, a 2-ary (binary) sequence is a sequence of zeros and ones. We also say that a k-digit 'b-ary' number is a string of k integers $x_1 x_2 \ldots x_k$, where $0 \leq x_j \leq 1$ for $0 \leq j \leq k$.

Definition 6. *A b-ary sequence is said to be k-distributed if*

$$\widehat{Pr}(X_n X_{n+1} \ldots X_{n+k-1} = x_1 x_2 \ldots x_k) = \frac{1}{b^k}$$

for all b-ary numbers $x_1 x_2 \ldots x_k$.

In this sense it is natural to speak about ∞-distributed b-ary sequences. The representation of a positive real number in the radix-b number system may be regarded as a b-ary sequence; for example, π corresponds to the 10-ary sequence 3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, ... It has been conjectured that this sequence is ∞-distributed, but nobody has yet been able to prove that it is even 1-distributed (for more on this topic see [10]).

Let us stop here for a moment and analyze the situation where k equals a million. A binary sequence that is 1,000,000-distributed is going to have runs of a million zeros in a row! Similarly, a $[0, 1)$ sequence that is 1,000,000-distributed is going to have runs of a million consecutive values each of which is less than 1/2. It is true that this will happen only $(1/2)^{1000000}$ of the time, on the average, but the fact is that it does happen. Indeed, this phenomenon will occur in any truly random sequence, using our intuitive notion of 'truly random'. One can easily imagine that such a situation will have a drastic effect if this set of a million 'truly random' numbers is being used in a computer-simulation experiment; there would be good reason to complain about the random number generator. However, if we have a sequence of numbers that never has runs of a million consecutive U's less than 1/2, the sequence is not random, and it will not be a suitable source of numbers for other conceivable applications that use extremely long blocks of U's as input. In summary, a truly random sequence will exhibit local nonrandomness. Local nonrandomness is necessary in some applications, but it is disastrous in others. We are forced to conclude that no sequence of 'random' numbers can be adequate for every application.

Before we go further we would like to point out that the conception of ∞-distributed b-ary sequences are not new at all, although in other fields of mathematics it is referred as *normal* numbers. Let $x = .x_1 x_2 x_3 \ldots$ be an infinite decimal to base b and let X_n denote the block of digits $x_1 x_2 \ldots x_n$. For any particular value j among this b possibilities, let $N(j, X_n)$ denote the number of occurrences of j in the block X_n. For example, if

$X_6 = 101011$, then $N(0, X_6) = 2$ and $N(1, X_6) = 4$. To illustrate the further meaning of this notation we mention that

$$\sum_{j=0}^{b-1} N(j, X_n) = n$$

Definition 7. *The number x is simply normal to base b if*

$$\lim_{n\to\infty} \frac{1}{n} N(j, X_n) = \frac{1}{b}$$

for each of the b different values of j.

Thus x is *simply normal* to base b if each digit j occurs with frequency $1/b$. A number x is *normal* to base b if each block B_k of k digits occurs with frequency $1/b^k$. Note that normal numbers are nothing else but ∞-distributed b-ary sequences. We do not know whether such numbers as $\sqrt{2}$ or π are normal to any base. On the other hand it is not hard to see that the decimal expansion obtained by writing the natural numbers in order gives a normal number.

Theorem 1. *The number*

$$x = 0.\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15 \ldots \tag{2.3}$$

formed by writing the natural numbers in succession is normal to base 10.

We omit the proof which is rather technical. The above theorem also shows that normality alone is not a sufficient condition of randomness.

The advantage of considering normal numbers instead of ∞-distributed sequences is that they are well understood and studied extensively in the past. For instance we know that the Lebesgue measure of normal numbers in the interval $[0, 1]$ is 1. In other words:

Theorem 2. *Almost all real numbers are normal to every base.*

Borel was the first to show this. A very nice and elementary proof is presented in Niven's book [16]. The theorem tells us that numbers which are not normal have a Lebesgue measure 0. We will come back to this theorem in the next section when we introduce the Martin-Löf definition of randomness. Another nice property of normal numbers that they pass a lot of statistical tests, like frequency test and serial test. We will describe these tests in Chapter 4.

Finally we reached a point where we can present a meaningful definition for random sequences. Theorem 1 shows that we cannot put equality sign between normal numbers (or ∞-distributed sequences) and random numbers. There are uncountably many sequences

$U_0, U_1, \ldots$ of real numbers between zero and one. If a truly random number generator is sampled to give values $U_0, U_1, \ldots$, any of the possible sequences may be considered equally likely, and some of the sequences (indeed, uncountably many of them) are not even equidistributed. On the other hand, using any reasonable definition of probability on this space of all possible sequences leads us to conclude that a random sequence is ∞-distributed with probability one. Therefore we are led to formalize the following definition of randomness:

Definition 8. *A* $[0, 1)$ *sequence is said to be 'random' if each of its infinite subsequences is* ∞*-distributed.*

However, the definition turns out to be too strict; any equidistributed sequence $\langle U_n \rangle$ has a monotonic subsequence with $U_{s_0} < U_{s_1} < U_{s_2} < \cdots$. The secret is to restrict the subsequences so that they could be defined by somebody who does not look at U_n, before deciding whether or not it is to be in the subsequence. The following definition now suggests itself:

Definition 9. *A* $[0, 1)$ *sequence* $\langle U_n \rangle$ *is said to be 'random' if, for every* effective algorithm *that specifies an infinite sequence of distinct nonnegative integers* s_n *for* $n \geq 0$ *the subsequence* $U_{s_0}, U_{s_1}, U_{s_2}, \ldots$ *corresponding to this algorithm is* ∞*-distributed.*

Although Knuth refines the above definition further since at this point it is not clear what exactly an *efficient algorithm* represents. We stop here since Definition 9 is already suitable for our goals. In the next section we will show three other approaches to describe random sequences.

2.3 Further approaches

The reader is probably overwhelmed by the number of definitions presented in the last section. Nevertheless it was inevitable, since randomness itself is a very complex phenomenon. As there is no royal road to geometry, there is no shortcuts to randomness either. However the reader might find some comfort in the fact that the completely different approaches which will be introduced in this section are mathematically equivalent.

The first efforts to describe randomness in the language of mathematics was made by Richard von Mises [21]. He had attempted to formalize the notion of a test for randomness in order to define a random sequence as one that passed all tests for randomness; however, the precise notion of a randomness test was left vague. Different schools of mathematics approached the problem from different angle. As a result many definitions were born along with Knuth's. Probably the most referred ones are the Martin-Löf definition of randomness and the Kolmogorov complexity.

Martin-Löf's key insight was to use the theory of computation to formally define the notion of a test for randomness [11]. This contrasts with the idea of randomness in probability; in that theory, no particular element of a sample space can be said to be random.

Martin-Löf randomness has since been shown to admit many equivalent characterizations - in terms of compression, randomness tests, and gambling. These bear little outward resemblance to the original definition, but they satisfy our intuitive notion of properties that random sequences ought to have: random sequences should be incompressible, they should pass statistical tests for randomness, and it should be difficult to make money betting on them. The existence of these multiple definitions of Martin-Löf randomness, and the stability of these definitions under different models of computation, give evidence that Martin-Löf randomness is a fundamental property of mathematics and not an accident of Martin-Löf's particular model.

For a finite binary string w let C_w denote the cylinder generated by w. This is the set of all infinite sequences beginning with w, which is a basic open set in Cantor space. The product measure $\mu(C_w)$ of the cylinder generated by w is defined to be $2^{-|w|}$. An effectively open set is an open set U such that there is a recursively enumerable set[1] $S \subseteq \mathbb{N}$ with $U = \bigcup_{i \in S} U_i$. A constructive null cover is a recursively enumerable sequence U_i of effective open sets such that $U_{i+1} \subseteq U_i$ and $\mu(U_i) \leq 2^{-i}$ for each natural number i. Every effective null cover determines a G_δ set of measure 0, namely the intersection of the sets U_i (in other words $G_\delta = \bigcap_i U_i$).

Definition 10. *A sequence is defined to be Martin-Löf random if it is not contained in any G_δ set determined by a constructive null cover.*

The null cover characterization conveys the intuition that a random real number should not have any property that is 'uncommon'. Each measure 0 set can be thought of as an uncommon property. Of course there is always a 0 measure set for every sequence, since the measure of a one-point set is 0. Martin-Löf's idea was to limit the definition to measure 0 sets that are *effectively* describable. Again we bump into the same idea as in the case of Knuth's definition. We have to construct a 0 measure set *beforehand* and only then we check if the sequence lies in it or not. If we cannot construct such a 0 measure set we regard the sequence random.

Another widely accepted definition of random sequences relies on the experience that they are hard to compress. Leonid Levin and Claus-Peter Schnorr proved this equivalent

[1]We do not intend to dive into the terminology of computability theory. Therefore we wont give a rigorous definition for terms like efficient algorithm or recursively enumerable sets.

characterization in terms of Kolmogorov complexity: a sequence is random if there is a uniform bound on the compressibility of its initial segments [7]. Kolmogorov complexity can be thought of as a lower bound on the algorithmic compressibility of a finite sequence (of characters or binary digits). It assigns to each such sequence w a natural number $K(w)$ that, intuitively, measures the minimum length of a computer program (written in some fixed programming language) that takes no input and will output w when run. Given a natural number c and a sequence w, we say that w is *c-incompressible* if $K(w) \geq |w| - c$.

Definition 11. *An infinite sequence S is Martin-Löf random if and only if there is a constant c such that all of S's finite prefixes are c-incompressible.*

The third equivalent characterization was due to Schnorr [19]. He exploited the idea that that no effective procedure should be able to make money betting against a random sequence. He defined a betting strategy d as a martingale. The martingale reads a finite prefix of the infinite sequence and bets money on the next bit. It bets some fraction of its money that the next bit will be 0, and then remainder of its money that the next bit will be 1. It doubles the money it placed on the bit that actually occurred, and it loses the rest. Let $d(w)$ denote the amount of money the martingale has after seeing the string w. The martingale characterization says that no betting strategy implementable by any computer (even in the weak sense of constructive strategies, which are not necessarily computable) can make money betting on a random sequence.

A martingale is a function $d : \{0,1\}^* \to [0, \infty)$ such that, for all finite strings w,

$$d(w) = \frac{d(w^\frown 0) + d(w^\frown 1)}{2} \tag{2.4}$$

where $a^\frown b$ is the concatenation of the strings a and b. Equation (2.4) is called the "fairness condition"; a martingale is viewed as a betting strategy, and the above condition requires that the better plays against fair odds. Let S be an infinite sequence and lets denote S_n the n long prefix of S. A martingale d is said to succeed on a sequence S if $\lim_{n\to\infty} d(S_n) = \infty$. A matringale d is constructive if there exists a computable function $\hat{d} : \{0,1\}^* \times \mathbb{N} \to \mathbb{Q}$ such that, for all finite binary strings w

1. $\hat{d}(w,t) \leq \hat{d}(w,t+1) < d(w)$ for all positive integers t,
2. $\lim_{t\to\infty} \hat{d}(w,t) = d(w)$

Definition 12. *A sequence is Martin-Löf random if and only if no constructive martingale succeeds on it.*

Li and Vitanyi's book An Introduction to Kolmogorov Complexity and Its Applications is an excellent introduction to all of the above presented ideas [8].

2.4 Finite pseudorandom sequences

Up to this point we considered only infinite sequences. Now we face some difficulties transferring the above ideas to the finite case. One may argue that there is no way to judge whether a finite sequence is random or not. Any binary sequence of length n has the same probability to appear in a coin flip. Still, nearly everyone would agree that the sequence 011101001 is "more random" than 101010101, and even the latter sequence is "more random" than 000000000. Although it is true that truly random sequences will exhibit locally nonrandom behavior, we would expect such behavior only in a long finite sequence, not in a short one.

From a point of view of statistics our expectation is not unrealistic at all. Consider the uniform distribution over $\{0, 1\}^n$. Lets choose a sequence randomly from this distribution, and count the number of ones it has. There are $n + 1$ possibilities as the sequence can have $0, 1, 2, \ldots, n$ ones. Most of the cases it will show around $[n/2]$ ones. Indeed we would be very lucky to observe a sequence with less than $\lfloor n/3 \rfloor$ or more than $\lfloor 2n/3 \rfloor$ ones. In this sense it is meaningful to say that it is improbable that a random binary sequence exhibits too few or too many ones.

The above test is called the *monobit test* and we will discuss it in details in Chapter 4. Many similar test and measure exist which try to describe the characteristics random sequence ought to have. Our aim is to design our pseudorandom number generator in such way that it emits sequences which are 'probable' in the above statistical sense. In the next chapter we will show some examples for PRNGs.

Chapter 3

Pseudorandom number generators

"The generation of random numbers is too important to be left to chance."
Robert R. Coveyou

Shortly after computers were introduced, people began to search for efficient ways to obtain random numbers within computer programs. Several methods were implemented like using previously generated tables of random numbers or attaching a physical random number generator to the computer. Neither of these were really satisfactory. The first eventually ran out of random numbers while the second was subject to malfunctions that are difficult to detect.

3.1 Middle-square method

One of the first PRNG was introduced by Neumann in 1949. The so called "middle-square method" has actually proved to be a comparatively poor source of random numbers. The algorithm looks like the following

1. take a $2n$ long integer
2. cut the middle part (which is an n long integer)
3. square it
4. repeat from step 2.

The above procedure fail to satisfy the definitions we set up in the introduction, since we can distinguish it from a real random generator in polynomial time in most of the cases. Basically two kind of problems can occur. The sequence of numbers the algorithm produces often degenerates to zero. For example,

$$12100043 \ \triangleright \ 1000^2 = 1000000 \ \triangleright \ 0$$

We can still fix this by adding a fifth step to the algorithm: every time zero is hit, go to step 1. Nevertheless even if we disregard the above problem it still can happen that the procedure ends up in a short cycle.

$$12475543 \ \triangleright \ 4755^2 = 22610025 \ \triangleright \ 6100^2 = 37210000 \ \triangleright \ 2100^2 = 441000 \triangleright$$
$$\triangleright \, 4100^2 = 16810000 \ \triangleright \ 8100^2 = 65610000 \ \triangleright \ 6100^2 = 37210000 \ldots$$

Several people experimented with the middle-square method in the early 1950s. N. Metropolis conducted extensive tests on it, mostly in the binary number system [5]. He showed that when 20-bit numbers are being used, there are 13 different cycles into which the sequence might degenerate, the longest of which has a period of length 142. On the other hand, working with 38-bit numbers, Metropolis obtained a sequence of about $750,000$ numbers before degeneracy occurred, and the resulting $750,000 \times 38$ bits satisfactorily passed statistical tests for randomness. This shows that the middle-square method can give usable results, but it is rather dangerous to put much faith in it until after elaborate computations have been performed.

3.2 Linear congruential method

The most popular random number generators in use today are special cases of the scheme introduced by Lehmer in 1949. The algorithm itself is very simple. We choose four positive integers which compose the engine of the PRNG,

- m, the modulus $m > 0$,
- a, the multiplier $2 \leq a < m$,
- c, the increment $0 \leq c < m$,
- X_0 the starting value $0 \leq X_0 < m$.

The sequence of pseudorandom numbers $\langle X_n \rangle$ is then obtained in the following way:

$$X_{n+1} = aX_n + c \quad (mod\ m) \qquad (3.1)$$

This is called a linear congruential sequence. Looking at (3.1) its is obvious that $a = 0$ and $a = 1$ does not result in a pseudorandom sequence. Consequently we cannot expect the sequence to be 'random' for any choice of m, a, c and X_0. For instance if $m = 1000$ and $a = c = 100$ the algorithm gets into a short cycle for any starting value X_0. Naturally the sequence will get into a loop eventually. The longest period the sequence can have is m. This is not a problem since we can choose m to be arbitrarily large, the question is rather how to achieve a period like that. When m is the product of distinct primes, only $a = 1$ will produce the full period, but when m is divisible by a high power of some prime there is considerable latitude in the choice of a. The following theorem makes it easy to tell if the maximum period is achieved.

Theorem 3. (Hull&Dobell,1962) *The linear congruential sequence defined by m, a, c, and X_0 has period length m if and only if*

- *c is relative prime to m;*
- *$a - 1$ is a multiple of p for every prime p dividing m;*
- *$a - 1$ is a multiple of 4 if m is a multiple of 4.*

For the proof see [5].

The linear congruential method gives more or less uniformly distributed integers from 0 to m. However they are predictable hence not random. We can easily compute the hidden parameters from a few consecutive output. For the same reason it is trivial that the linear congruential method does not satisfy the conditions of Definition 2. Clearly it is worthless from a cryptographyical point of view. Still there are plenty of other applications which need pseudorandom numbers and the simplicity of the algorithm[1] makes it a good candidate as a PRNG.

3.3 Linear feedback shift registers

Linear feedback shift registers (LFSR) were developed for cryptographyical purposes during the second world war. They have been used as pseudorandom number generators, due to the ease of construction from simple electronic circuits, long periods, and very uniformly distributed output streams. An LFSR constitutes basically from two parts. A shift register and a linear feedback function. The shift registers task is to shift its contents (which is a binary string) into adjacent positions within the register or, in the case of the

[1]It is not just easy to understand but also simple to put in a program language. An often used parameter-set is $m = 2^{32}$, $a = 1664525$, $c = 1013904223$.

position on the end, out of the register. The feedback function is the XOR function of some previously fixed bit positions. This is always so since the only linear function of single bits is the exclusive-or. The bit positions that affect the next state are called the taps. For example our LFSR has the following parameters:

- 4 bit long register
- the 3rd and 4th bits are the taps
- the initial value is 0001

The content of the register in the consecutive time slots:

$$0001 \triangleright 1000 \triangleright 0100 \triangleright 0010 \triangleright 1001 \triangleright 1100 \triangleright 0110 \triangleright 1011 \triangleright$$
$$\triangleright 0101 \triangleright 1010 \triangleright 1101 \triangleright 1110 \triangleright 1111 \triangleright 0111 \triangleright 0011 \triangleright 0001$$

The output sequence is $s = 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, \ldots$ with period 15. It is clear from the example that the maximum period of an LFSR is $2^n - 1$, since zero occurs if only if it was the starting value. Maximum length period can be achieved by choosing the taps carefully. There is no quick way to determine if a tap sequence is maximal length. However, there are some ways to tell if one is not maximal length:

1. Maximal length tap sequences always have an even number of taps.
2. The tap values in a maximal length tap sequence are all relatively prime. A tap sequence like 12, 9, 6, 3 will not be maximal length because the tap values are all divisible by 3.

The arrangement of taps in an LFSR can be expressed in finite field arithmetic as a polynomial modulo 2. This is called the feedback or characteristic polynomial (or in some literature connection polynomial). For instance the the feedback polynomial of the above LFSR is $x^4 + x^3 + 1$. Interestingly very few - even 2 or 4 - taps can suffice in reaching maximal period length. For example all the 100-long binary strings (except the zero string) can be generated with an LFSR having a feedback function $x^{100} + x^{63} + 1$. The importance of this representation is due to the following fact:

Theorem 4. *Let G be an n long LFSR. If $h(x) \in \mathbb{Z}_2[x]$ is a primitive polynomial of degree n, then the tap system represented by $h(x)$ produces an output sequence of period $2^n - 1$.*

For more details see [1].

Discovering a maximal length tap sequence leads automatically to another. If a maximal length tap sequence is described by [n, A, B, C], another maximal length tap sequence will be described by [n, n-C, n-B, n-A]. For example the mirror of the feedback polynomial which we mentioned above is $x^{100} + x^{37} + 1$. What is more, the mirror tap sequence will cycle through the states in reverse order! This is one of the reasons why shift registers are not used for cryptographic purposes anymore. Given a stretch of known plaintext and corresponding ciphertext, an adversary can intercept and recover a stretch of LFSR output stream. From that stretch of the output stream the adversary can construct an LFSR of minimal size that simulates the intended receiver by using the Berlekamp-Massey algorithm. This LFSR can then be fed the intercepted stretch of output stream to recover the remaining plaintext (in both directions, backward and forward as well).

Shift registers are still in use in digital broadcasting and in some computer applications since their simple structure allows to construct them in the forms of electronic circuits.

3.4 Further theoretical discussion

So far none of the pseudorandom generators we presented passed the conditions we set up in the introduction. We also mentioned that it might be the case that there is no algorithm which satisfies both Definition 1 and 2. However this is quite unlikely. Our belief is based on the fact the pseudorandom number generators can be constructed under the rather weak assumption that *one-way functions* exist. In order to do this first let us formulate another property of pseudorandom generators.

Definition 13. *A pseudorandom bit generator is said to pass the next-bit test if there is no polynomial-time algorithm which, on input of the first l bits of an output sequence s, can predict the $(l+1)$st bit of s with probability significantly greater than* $1/2$.

Although Definition 2 appears to impose a more strict requirement on pseudorandom generators than Definition 13 does, the next result asserts that they are, in fact, equivalent.

Theorem 5. *A pseudorandom bit generator passes the next-bit test if and only if it passes all polynomial-time statistical tests.*

Obviously if a PRNG cannot be distinguished from a real random generator in polynomial time it passes the next-bit test as well. For the other direction we need a so called hybrid argument. This proof technique is often used in cryptography to show that two distributions are computationally indistinguishable. We omit the proof which is rather technical, but the reader may find many hybrid type arguments in [4] from which the above theorem is clearly follows.

A one-way function f has the property that it is easy to compute, but hard to invert. The first condition means nothing more than f should be computable in polynomial time. As for the second we shall keep that in mind that ultimately we want a definition which is suitable for building secure cryptographic schemes. Therefore we formalize the hardness of inverting f by requiring that it is infeasible for any polynomial time adversary to invert f - that is to find a pre-image of a given value $y = f(x)$ - except with negligible probability. Consider the following experiment defined for any algorithm $\mathcal{A}$ and security parameter n.

The inverting experiment, $INV_{\mathcal{A},f}(n)$

1. Choose an input $x \in_R \{0,1\}^n$ (randomly) and compute $y = f(x)$.
2. $\mathcal{A}$ is given the security parameter and y as input and outputs x'.
3. The output of the experiment is defined to be 1 if $f(x') = y$ and 0 otherwise.

Note that $\mathcal{A}$ does not need to find x itself. It suffices for $\mathcal{A}$ to find any value x' for which $f(x') = y = f(x)$. Also observe that we are using the asymptotic approach again: we want to allow $\mathcal{A}$ to run in polynomial time in the security parameter n, irrespective to the length of y. Now we are ready to give an exact definition.

Definition 14. *A function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is one-way if the following two conditions hold:*

1. *There exists a polynomial time algorithm M_f computing f; that is, $M_f(x) = f(x)$ for all x.*
2. *For every polynomial time algorithm $\mathcal{A}$, there exists a negligible function $\epsilon(n)$ such that*
$$Pr(INV_{\mathcal{A},f}(n) = 1) \leq \epsilon(n)$$

We note that a one-way function which is length preserving (i.e. $|f(x)| = |x|$) and one-to-one is called one-way permutation. Naturally any one-way function can be inverted given enough time. It is always possible to simply try all values $x \in \{0,1\}^n$ until a value x is found such that $f(x) = y$. Also the one-way property does not mean that everything about the preimage is hidden. It is possible that $f(x)$ 'leaks' a lot of information about x, yet f is still hard to invert. For example let f be one-way and let its input be an n long string, $x_n = x_1x_2 \dots x_n$ and define $g(x) = (x_1, f(x))$. Although g reveals the first bit of its input it is trivially one-way.

Before we can construct a PRNG we need another function which is called hardcore predicate. Our aim is to find some information about x which is hidden by $f(x)$. Therefore we formulate the following definition

Definition 15. *A function $hc : \{0,1\}^* \rightarrow \{0,1\}$ is a hardcore predicate of a function f if*

1. *hc can be computed in polynomial time*
2. *for every polynomial time algorithm $\mathcal{A}$ there exist a negligible function $\epsilon(n)$ such that*
$$\Pr_{x \in_R \{0,1\}^n}[\mathcal{A}(f(x)) = hc(x)] \leq \frac{1}{2} + \epsilon(n).$$

The second condition states that it is infeasible for any polynomial time algorithm to correctly determine $hc(x)$ with probability significantly better than 1/2. Of course it is always possible to compute $hc(x)$ correctly with probability exactly 1/2 by random guessing.

Hardcore predicates are not so easy to construct. It is still an open question whether hardcore predicate exist for any one-way function. However a slightly weaker statement is true:

Theorem 6. *For any one-way function f there exists (constructively) a one-way function g along with a hardcore predicate gl for g.*

The hardcore predicate is denoted gl after Goldreich and Levin who proved Theorem 6 (see in [4]). Functions g and gl are constructed as follows:

$$g(x,r) \overset{def}{=} (f(x), r); \qquad gl(x,r) \overset{def}{=} \bigoplus_{i=1}^{n} x_i r_i$$

where $x \in \{0,1\}^n$ is the input and $r \in_R \{0,1\}^n$ chosen randomly. In this way the function $gl(x,\cdot)$ outputs the exclusive-or of a *random subset* of the bits of x. This is due to the fact that r can be viewed as selecting a random subset of $\{1,2,\ldots,n\}$ (i.e., when $r_i = 1$ the bit x_i is included in the XOR otherwise not). Thus, Theorem 5 essentially states that if f is an arbitrary one-way function, then $f(x)$ hides the exclusive-or of a random subset of the bits of x.

Now we obtained every tool to construct a pseudorandom number generator.

Theorem 7. *Let f be a one-way permutation and let hc be a hardcore predicate of f. Then $G(s) \overset{def}{=} (f(s), hc(s))$ constitutes a pseudorandom generator with expansion factor $l(n) = n+1$.*

Indeed G satisfies the criterions of both definitions. Clearly it expands the seed by one bit. However it is not so obvious why all polynomial time algorithm fail to distinguish between the output of $G(s)$ and a random string of length $n+1$. We remark first, that the initial n bit of G are *truly* random when the seed was chosen uniformly at random due to the fact that f is a permutation. Also note that hc is a hardcore predicate means that $hc(s)$ 'looks random'. This idea is supported by the fact that by definition the hardcore predicate passes the next-bit test. Putting these observation together we see that the entire output of G is pseudorandom.

Finally we still owe an explanation why the existence of one-way functions are more plausible than the existence of PRNGs. Let alone an unconditional proof of the existence of one-way functions would imply a major breakthrough in complexity theory. However our assumption is far from being unfounded. There are some computational problems that received much attention and have yet to yield polynomial time algorithms. Some of the most famous of these problems include *integer factorization*, the *subset-sum problem* and the *discrete logarithm problem.* For instance a candidate for one-way function can be defined the following way

$$f(x_1, x_2, \ldots, x_n, J) = (x_1, x_2, \ldots, x_n, \sum_{j \in J} x_j)$$

where each x_j is an n-bit string interpreted as an integer, and J is an n-bit string interpreted as a subset of $\{1, 2, \ldots, n\}$. Given an output $(x_1, x_2, \ldots, x_n, y)$ of this function, the task of inverting it is exactly that of finding a subset $J' \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{j \in J'} x_j = y$. This is called the subset-sum problem and it is $\mathcal{NP}$-complete.

3.5 Modern PRNGs

Listing all the pseudorandom generators which are in use today would be a hopeless task. Instead we briefly introduce three of them to show an insight into the recent applications.

Mersenne Twister

This particular generator was developed by Matsumoto&Nishimura in 1997 and is based on a matrix linear recurrence over a finite binary field $\mathbb{F}_2$ [12]. It produces very high-quality pseudorandom numbers, having been designed specifically to rectify many of the flaws found in older algorithms. Its name derives from the fact that period length is chosen to be a Mersenne prime. There are at least two common variants of the algorithm, differing only in the size of the Mersenne primes used. The newer and more commonly used one is the Mersenne Twister MT19937, with 32-bit word length. There is also a variant with 64-bit word length, MT19937-64, which generates a different sequence. The algorithm in its native form is not suitable for cryptography. Observing a sufficient number of iterates (624 in the case of MT19937) allows one to predict all future iterates. Another issue is that it can take a long time to turn a non-random initial state into output that passes randomness tests. Therefore usually a linear congruential generator is used to seed the Mersenne Twister beforehand. There are many advantages of the Mersenne

Twister. First of all it has an incredibly long period of $2^{19937}-1$. This alone would not be exceptional but the Mersenne Twister passes numerous statistical tests, including every tests in the Diehard test-suit (see next chapter). Also it is k-distributed to 32-bit accuracy for every $1 \leq k \leq 623$. Thanks to these facts the Mersenne Twister becomes more and more popular and the algorithm is embedded in such programs as MATLAB, R or Maple.

Blum Blum Shub

Despite its funny name Blum Blum Shub is one of the most secure PRNGs. It has an unusually strong security proof which relates the quality of the generator to the computational difficulty of integer factorization [6]. The algorithm was developed in 1986 by Lenore Blum, Manuel Blum and Michael Shub and works in the following way

$$x_{n+1} = x_n^2 \quad (mod\ M)$$

where $M = pq$ is the product of two large primes p, q. At each step of the algorithm, some output is derived from x_{n+1}. Usually it is either the bit-parity or the least significant bit of term in question. The reason behind this is to prevent leaking too much information about the internal state of the PRNG. Any adversary who can recover two consecutive states of the generator (x_n, x_{n+1}) would easily determine the modulus and therefore crack the system. The two primes, p and q, should both be congruent to 3 (mod 4) (this guarantees that each quadratic residue has one square root which is also a quadratic residue) and $gcd(\varphi(p-1), \varphi(q-1))$ should be small in order to have a big cycle length. For example let $p = 19$, $q = 23$ and $x_{-1} = 4$, where x_{-1} is the initializing value. We can expect to get a relatively big cycle length for those numbers, because $gcd(\varphi(p-1), \varphi(q-1)) = 2$. Using the least significant bit formula we obtain the following output sequence:

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
value	16	256	423	196	397	289	54	294	347	234	...
output	0	0	1	0	1	1	0	0	1	0	...

An interesting characteristic of the Blum Blum Shub generator is the possibility to calculate any x_i value directly (via Euler's Theorem):

$$x_i = x_0^{2^i \ mod\ (p-1)(q-1)} \quad (mod\ M) \tag{3.2}$$

Unfortunately the algorithm is very slow therefore the generator is not appropriate for use in simulations.

Legendre symbol

Many mathematcians noticed that looking at a string of consecutive integers it seems erratic which of these integers is a quadratic residue for a given prime p and which is not. Motivated by this fact, Mauduit and Sárközy initiated a comprehensive study on the randomness of the Legendre symbol [13], [15]. A pseudorandom number generator can be constructed the following way:

1. Choose a large prime p randomly, this will be our secret key.
2. Choose a polynomial $f(x) \in \mathbb{F}_p[x]$ which is not in the form of $b(g(x))^2$ for any $b \in \mathbb{F}_p$ and $g(x) \in \mathbb{F}_p[x]$.
3. The output sequence is $E_p = \{e_1, e_2, \ldots, e_p\}$ is defined by

$$e_n = \begin{cases} \left(\frac{f(n)}{p}\right) & for\ (f(n), p) = 1, \\ +1 & for\ p|f(n). \end{cases}$$

For example let $p = 13$ and $f(x) = x^3 + 3$ then our pseudorandom bit sequence is

E_p	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_{11}	e_{12}
value	$\left(\frac{4}{13}\right)$	$\left(\frac{11}{13}\right)$	$\left(\frac{4}{13}\right)$	$\left(\frac{2}{13}\right)$	$\left(\frac{11}{13}\right)$	$\left(\frac{8}{13}\right)$	$\left(\frac{5}{13}\right)$	$\left(\frac{5}{13}\right)$	$\left(\frac{1}{13}\right)$	$\left(\frac{12}{13}\right)$	$\left(\frac{1}{13}\right)$	$\left(\frac{12}{13}\right)$
output	1	-1	1	-1	-1	-1	-1	-1	1	1	-1	1

Rivat and Sárközy conducted an exhaustive statistical testing of the Legendre symbol [18]. We will discuss some of the results in the next chapter.

Chapter 4

Statistical testing

"The sun comes up just about as often as it goes down, in the long run, but this doesn't make its motion random." Donald E. Knuth

Naturally it is impossible to give a mathematical proof that a generator is a random bit generator. The tests described here help detect certain kinds of weaknesses the generator may have. *A posteriori* testing means that we take a sample output sequence of the generator and subject it to various statistical tests. However even if a sequence behaves randomly with respect to n different tests we cannot be sure in general that it will not be a miserable failure when it comes to the $(n+1)$st test; yet each test gives us more and more confidence in the randomness of the sequence. There are several options available for those who are interested in analyzing their PRNG. The National Institute of Standards and Technology lists the following reliable test-suites [17]:

Source/affiliation	**Statistical test**
1. Donald Knuth Stanford University	The Art Of Computer Programming Vol. 2
2. George Marsaglia Florida State University	DIEHARD
3. Helen Gustafson, et. al. Queensland University of Technology	Crypt-XS
4. Alfred Menezes, et. al. CRC Press, Inc.	Handbook of Applied Cryptography
5. Andrew Rukhin, et. al. NIST ITL	NIST Statistical Test Suite

The DIEHARD suite of statistical tests developed by George Marsaglia consists of fifteen tests, namely the: birthday spacings, overlapping permutations, ranks of 31×31

and 32×32 matrices, ranks of 6×8 matrices, monkey tests on 20-bit Words, monkey tests OPSO, OQSO, DNA, count the 1's in a stream of bytes, count the 1's in specific bytes, parking lot, minimum distance, random spheres, squeeze, overlapping sums and runs.

The Crypt-XS suite of statistical tests was developed by researchers at the Information Security Research Center at Queensland University of Technology in Australia. Crypt-XS tests include the frequency, binary derivative, change point, runs, sequence complexity and linear complexity.

The NIST Statistical Test Suite is the result of collaborations between the Computer Security Division and the Statistical Engineering Division at NIST. Statistical tests in the package include the: frequency, block frequency, cumulative sums, runs, long runs, Marsaglia's rank, spectral (based on the Discrete Fourier Transform), nonoverlapping template matchings, overlapping template matchings, Maurer's universal statistical, approximate entropy (based on the work of Pincus, Singer and Kalman), random excursions (due to Baron and Rukhin), Lempel-Ziv complexity and linear complexity.

As the above list shows there are many kind of tests - each describing different characteristics a random sequence should comprise. There is no way we could introduce all of them. Instead we will describe the five most commonly used tests in the following sections. We will also say a few words about *a priori* testing. Before we start to list the parameters of these tests we will show an example how statistical test works in practice.

4.1 An insight to statistical testing

The following example was published by Marsaglia in his remarkable article the "Monkey tests for random number generators" [9]. He uses the famous 'monkey at the typewriter' metaphor to demonstrate how random number generators work. Imagine a monkey who has a typewriter with 26 uppercase letters $A, B, \ldots, Z$ that he strikes at random. More exactly he produces uniform reals in $[0, 1)$ by means of a procedure $UNI()$ then converts them by taking the integer part of $26 \cdot UNI()$. Now the CAT test: how many keys must the monkey strike until he spells CAT?

There are $26^3 = 17,576$ possible 3-letter words so the average number of keystrokes necessary to produce CAT should be around 17,576, and the time to reach CAT should be very close to exponentially distributed.

The monkey which uses the linear congruential method $X_{n+1} = 69069 \cdot X_n \mod 2^{32}$ and generates $UNI()$ by dividing X_n by 2^{32} gets CAT after 13,561 keystrokes, then after 18,263, then another 14,872 strokes produces the third CAT. Quite satisfactory.

Now consider the shift-register monkey with feedback function $x^{31} + x^3 + 1$ that produces 31-bit integers. The shift-register monkey never spells CAT, even after two million keystrokes. He can not get KAT, DOG, GOD, SEX, WOW or ZIG either. But he can get ZAG, and too often - every few thousand keystrokes. Indeed, it turns out that this monkey was only able to get 7,834 of the possible 17,576 3-letter words, (in a string of 1,000,000 keystrokes) and of course with his limited vocabulary, he gets those words too often.

Note that inability to get CAT does not mean that there are broken keys on the typewriter. This monkey still types each of the letters A to Z with the expected frequencies (and thus would pass a standard test for letter frequency). For example, 26,000 keystrokes produced 984 C's, 967 A's and 1021 T's, quite satisfactory. Yet continuing the run to 2,600,000 keystrokes failed to produce a single CAT!

As silly as it seems, this is a very effective and convincing way to show the unsuitability of certain random number generators. Of course for practical purposes every 3-letter word has to be tested, not just CAT.

4.2 Five basic tests

The following statistical tests are widely used for determining whether a binary sequence possess some specific characteristics that a truly random sequence would be likely to exhibit. We emphasize again that the outcome of each test is not definite, but rather probabilistic. If a sequence passes all five tests, there is no guarantee that it was indeed produced by a random bit generator. Let $E_N = \{e_1, e_2, \ldots, e_{N-1}\}$ be a binary sequence of length n.

(i) Frequency test (monobit test)

The purpose of this test is to determine whether the number of 0's and 1's in E_N are approximately the same, as would be expected for a random sequence. Let n_0, n_1 denote the number of 0's and 1's in E_N respectively. The statistic used is

$$X_1 = \frac{(n_0 - N/2)^2}{N/2} + \frac{(n_1 - N/2)^2}{N/2} = \frac{(n_0 - n_1)^2}{N/2}$$

which approximately follows a χ^2 distribution with degree of freedom one if $N \geq 10$.

(ii) Serial test (two-bit test)

Let n_{00}, n_{01}, n_{10} and n_{11} denote the number of occurrences of 00, 01, 10 and 11 in E_N respectively and let n_0 and n_1 be the same as above. The statistic used is

$$X_2 = \frac{4}{N-1}(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2) - \frac{2}{N}(n_0^2 + n_1^2) - 1$$

which approximately follows a χ^2 distribution with degree of freedom two if $N \geq 21$.

(iii) Poker test

Let m be a positive integer such that $\lfloor N/m \rfloor \geq 5 \cdot 2^m$ and let $k = \lfloor N/m \rfloor$. Divide the sequence E_N into k non-overlapping parts each of length m. There are 2^m different m words. Let denote n_i the number of occurrences of the i^{th} type of sequence of length m where $1 \leq i \leq 2^m$. The statistic used is

$$X_3 = \frac{2^m}{N}\left(\sum_{i=1}^{2^m} n_i^2\right) - k$$

which approximately follows a χ^2 distribution with degree of freedom $2^m - 1$. Note that the poker test is a generalization of the frequency test: setting $m = 1$ in the poker test yields the frequency test.

(iv) Runs test

A *run* of E_N is a subsequence of E_N consisting of consecutive 0's or consecutive 1's which is neither preceded nor succeeded by the same symbol. The purpose of the runs test is to determine whether the number of runs (of either zeros or ones) of various lengths in the sequence E_N is as expected for a random sequence. A run of zeros is called a gap and a run of ones is called a block. The expected number of gaps and blocks of length i in a random sequence of length N is $m_i = (N - i + 3)/2^{i+2}$. Let k be equal to the largest integer i for which $m_i \geq 5$. Let B_i, G_i be the number of blocks and gaps, respectively, of length i in E_N for each i, $1 \leq i \leq k$. The statistic used is

$$X_4 = \sum_{i=1}^{k} \frac{(B_i - m_i)^2}{m_i} + \sum_{i=1}^{k} \frac{(G_i - m_i)^2}{m_i}$$

which approximately follows a χ^2 distribution with degree of freedom $2k - 2$.

(v) Autocorrelation test

The purpose of this test is to check for correlations between the sequence E_N and (non-cyclic) shifted versions of it. Let d be a fixed integer, $1 \leq d \leq \lfloor N/2 \rfloor$. The number of bits in E_N not equal to their d-shifts is $A(d) = \sum_{i=1}^{N-d-1} e_i \oplus e_{i+d}$, where $\oplus$ denotes the XOR operator. The statistic used is

$$X_5 = 2\left(A(d) - \frac{N-d}{2}\right)/\sqrt{N-d}$$

which approximately follows an $N(0;1)$ distribution if $N-d \geq 10$. Since small values of $A(d)$ are as unexpected as large values of $A(d)$, a two-sided test should be used.

As the above examples show in most of the cases we try to pinpoint a characteristics of binary sequences that we can compare with the χ^2 or the normal distribution. There are some other type of tests. For example we can assign to each binary sequence a random walk on the real line (0 means we go one step to the left and 1 means we go one step to the right). We know that the expected outcome of such a walk should be around zero. Another test is to find the shortest linear feedback shift register which outputs our sequence. Random sequences are characterized by a longer feedback register. A short feedback register implies non-randomness and so on. As we indicated in the beginning of this chapter there are countless number of tests and we do not intend to introduce all of them. Rather we move on to the topic of theoretical testing.

4.3 A priori testing

The idea behind *a priori* or as Knuth calls it "theoretical" testing is to prove something on the distribution of the PRNG instead of taking a sample from the possible outputs and testing them. Cassaigne, Maduit, Rivat and Sárközy provided many results in this topic, especially connected to the Legendre-symbol [13], [14], [15]. To give an insight now we present two examples for such results. For convenience sake we remind the reader to the definition of well-distribution measure.

$$W(E_N) \overset{def}{=} \max_{a,b,t} \left| \sum_{j=0}^{t} e_{a+jb} \right|$$

where $E_N = \{e_1, e_2, \ldots, e_N\} \in \{-1,1\}^N$ and the maximum is taken over all a, b, t such that $a \in \mathbb{Z}$, $b, t \in \mathbb{N}$ and $1 \leq a+b \leq a+tb \leq N$.

Theorem 8. *For all binary sequences $E_N = \{e_1, e_2, \ldots, e_N\} \in \{-1,1\}^N$ we have*

$$X_1 \leq \frac{1}{N}(W(E_N))^2$$

where X_1 is the test statistics for the monobit test.

Proof of Theorem 8

Let n_- and n_+ denote the number of -1's and 1's in E_N respectively. We have

$$n_- = -\frac{1}{2}\sum_{i=1}^{N}(e_i - 1), \ \textit{and}\ n_+ = \frac{1}{2}\sum_{i=1}^{N}(e_i + 1)$$

and so

$$X_1 = \frac{1}{N}(n_- - n_+)^2 = \frac{1}{N}\left(\sum_{i=1}^{N} e_i\right)^2 \leq \frac{1}{N}(W(E_N))^2$$

□

Another result involves the Correlation measure. Let $X = \{x_1, \dots, x_k\} \in \{-1,1\}^k$, $M \in \mathbb{N}$ and $E \in \{-1,1\}^\infty$. We write

$$T(E, M, X) = |\{n : 0 \leq n < M,\ \{e_{n+1, e_{n+2}, \dots, e_{n+k}}\} = X\}|$$

Then E is said to be normal (or ∞-distributed) if

$$|T(E, M, X) - M/2^k| = o(M)$$

for all fixed k and X, as $M \to \infty$. With the help of $T(E, M, X)$ we can transfer the normality requirement to finite sequences.

Definition 16. *A finite sequence $E_N = \{e_1, e_2, \dots, e_N\} \in \{-1,1\}^N$ is said to be pseudorandom if for all $k \in \mathbb{N}$ with $k \leq (\log N)/\log 2$ and for all $X = \{x_1, \dots, x_k\} \in \{-1,1\}^k$ we have*

$$\left| T(E_N, N+1-k, X) - \frac{N+1-k}{2^k} \right| \leq \sqrt{N}.$$

Definition 16 may be too strict, since it considers a sequence either pseudorandom or not. A more flexible approach is to construct a measure from $T(E_N, M, X)$.

Definition 17. *We define the Normality measure of order k by:*

$$N_k(E_N) = \max_{X \in \{-1,1\}^k} \max_{0 < M \leq N+1-k} |T(E_N, M, X) - M/2^k|,$$

and the Normality measure by

$$N(E_N) = \max_{k \leq (\log N)/\log 2} N_k(E_N).$$

Sequences with reasonably small values of Normality measure can be accepted as pseudorandom. It depends on the application how we interpret "reasonably", but as a rule of thumb we can say that it should be less or proportional to $\sqrt{N}$. The following theorem guarantees that Normality measure can be controlled with the help of the Correlation measure. For the reader's convenience we repeat the definition of the Correlation measure of order k:

$$C_k(E_N) = \max_{M,D} |V(E_N, M, D)| = \max_{M,D} \left| \sum_{n=1}^{M} e_{n+d_1} \dots e_{n+d_k} \right|,$$

where $D = (d_1, \dots, d_k) \in \mathbb{N}^k$, $0 \le d_1 \le \dots \le d_k$ and the maximum is taken over all D and M such that $M + d_k \le N$. Now the theorem

Theorem 9. (Mauduit&Sárközy) *For all N, E_N and $k < N$ we have*

$$N_k(E_N) \le \max_{1 \le t \le k} C_t(E_N).$$

Proof of Theorem 9

For all $k, N \in \mathbb{N}$, $X = \{x_1, \dots, x_k\} \in \{-1, 1\}^k$ and $1 \le M \le N + 1 - k$ we have

$$\begin{aligned}
&|T(E, M, X) - M/2^k| = \\
&= \left| |\{n : 0 \le n < M,\ \{e_{n+1}, e_{n+2}, \dots, e_{n+k}\} = X\}| - \frac{M}{2^k} \right| = \\
&= \left| \sum_{n=0}^{M-1} \frac{x_1 \cdots x_k}{2^k} \prod_{j=1}^{k} (e_{n+j} + x_j) - \frac{M}{2^k} \right| = \\
&= \left| \frac{x_1 \cdots x_k}{2^k} \sum_{1 \le d_1 < \dots < d_t \le k} \left(\prod_{j \in \{1,\dots,k\} \setminus \{d_1,\dots,d_t\}} x_j \right) \sum_{n=0}^{M-1} e_{n+d_1} \cdots e_{n+d_t} \right| \le \\
&\le \frac{1}{2^k} \sum_{\emptyset \ne D \subset \{1,2,\dots,k\}} \left| \sum_{n=0}^{M-1} e_{n+d_1} \cdots e_{n+d_t} \right| \le \frac{1}{2^k} \sum_{t=1}^{k} \binom{k}{t} C_t(E_N) \le \\
&\le \max_{1 \le t \le k} |C_t(E_N)|
\end{aligned}$$

$\square$

We managed to trace back two important characteristics of random sequences to the Well-distribution and Correlation measures. Should any PRNG have good scores at these measures they immediately pass the normality and monobit tests as well. Rivat and Sárközy proved in a conference paper that all the five basic tests can be controlled with the help of the Well-distribution and Correlation measures [18]. They also proved that the Legendre-symbol - with appropriate setup - can be adjusted to pass these tests. This can be considered the main advantage of a priori testing. In the next chapter we will introduce new measures and obtain some theoretical results.

Chapter 5

Results

"Perhaps in ten years society may derive advantage from the curves which these visionary algebraists will have laboriously squared. I congratulate posterity beforehand. But to tell you the truth I see nothing but a scientific extravagance in all these calculations. That which is neither useful nor agreeable is worthless. And as for useful things, they have all been discovered; and to those which are agreeable, I hope that good taste will not admit algebra among them." Frederick the Great

We have seen how useful the Well-distribution and Correlation measure is in theoretical testing. On the other hand it is clear hand that these two measures cannot possibly describe all the characteristics that pseudorandom sequences possess. This observation led to the introduction of the symmetry measure. To justify the necessity of a new measure we have to prove that it is fairly independent from the old ones. The following two constructions show that this is indeed the case. For the readers convenience we repeat the definition of the symmetry measure

$$S(E_N) = \max_{a<b} \left| \sum_{j=0}^{[\frac{b-a}{2}]-1} e_{a+j} e_{b-j} \right|,$$

where the maximum is taken over all $a, b \in \mathbb{N}$ such that $1 \leq a < b \leq N$.

Example 1

Consider a sequence $E_N = \{e_1, e_2, \ldots, e_N\} \in \{-1, 1\}^N$ such that each of the Symmetry, Correlation and Well-distribution measure of it are possibly small (it is a fact that all these measures can be $O\left((N \log N)^{1/2}\right)$ simultaneously). We define $E^*_{2N} = \{e^*_1, e^*_2, \ldots, e^*_{2N}\} \in \{-1, 1\}^{2N}$ by

$$e^*_n = \begin{cases} e_n & for\ 1 \leq n \leq N \\ e_{n-N} & for\ N < n \leq 2N \end{cases}$$

Then it is easy to see that the Well-distribution measure measure of E^*_{2N} is less than a constant times the corresponding measure of E_N. Furthermore the Symmetry measure $S(E^*_{2N}) \leq S(E_N) + C_2(E_N)$, but

$$C_2(E^*_{2N}) \geq \left| \sum_{n=1}^{N} e - n^* e^*_{n+N} \right| = N$$

Example 2

Consider the same sequence E_N again. We define $E'_{2N} = \{e'_1, e'_2, \ldots, e'_{2N}\} \in \{-1, 1\}^{2N}$ by

$$e'_n = \begin{cases} e_n & for\ 1 \leq n \leq N \\ e_{2N-n} & for\ N < n \leq 2N \end{cases}$$

Then the correlation measure of order 2 is less than a constant times $S(E_N)+C_2(E_N)$, while $W(E'_{2N}) \leq 2W(E_N)$. But

$$S(E'_{2N}) \geq \left| \sum_{n=1}^{N} e - n' e'_{2N-n} \right| = N$$

$\square$

There are many possible symmetrical patterns that a sequence can exhibit. The symmetry measure only indicates if a sequence has a large symmetrical subsequence or not. It is a reasonable idea to broaden the range of patterns that the symmetry measure can detect. In the forthcoming sections we will show two possible generalizations. We will also give upper and lower bounds for the generalized measures.

5.1 Generalizations

Let $A = (a_1, a_2, \ldots, a_t) \in \mathbb{N}^t$, $B = (b_1, b_2, \ldots, b_t) \in \mathbb{N}^t$, $a_1 < a_2 < \cdots < a_t$ and $1 \leq A < B \leq N$ elementwise (i.e. $1 \leq a_i < b_i \leq N$ for $i = 1, 2, \ldots, t$). We write

$$SM(E_N, t) = \max_{A<B} \left\{ \left| \sum_{j=0}^{[\frac{b_1-a_1}{2}]-1} (e_{a_1+j} e_{b_1-j}) \right| + \cdots + \left| \sum_{j=0}^{[\frac{b_t-a_t}{2}]-1} (e_{a_t+j} e_{b_t-j}) \right| \right\}, \qquad (5.1)$$

where the maximum is taken over all possible A's and B's such that $A < B$. In this way $SM(E_N, t)$ indicates if the sequence has multiple symmetrical subsequences or not.

We define $SD(E_N, d)$ by

$$SD(E_N, d) = \max_{a,b} \left| \sum_{j=0}^{[\frac{b-a}{2d}]-1} e_{a+dj} e_{b-dj} \right|, \tag{5.2}$$

which counts the largest symmetric arithmetic progression of difference $d \in \mathbb{N}$ outgoing from a specific center (determined by a and b).

Note that $SM(E_N, 1) = S(E_N) = SD(E_N, 1)$ so indeed both measures are generalizations of the symmetry measure.

Theorem 10. *There is an integer N_0 such that for $N > N_0$ we have*

$$a) \qquad SM(E_N, t) > \frac{7\sqrt{t}}{20}\sqrt{N} - \frac{t}{2}, \tag{5.3}$$

$$b) \qquad SD(E_N, d) > \frac{7}{20d}\sqrt{N}. \tag{5.4}$$

In the next theorem we will estimate $SM(E_N, t)$ and $SD(E_N, d)$ for fixed t's and d's and for 'random' binary sequences $E_N \in \{1, -1\}^N$ (i.e. we are choosing each $E_N \in \{1, -1\}^N$ with probability $1/2^N$). In this way the upper bounds hold only for the majority of sequences.

Theorem 11. *For all $\epsilon > 0$ there exits $N_0(\epsilon)$ such that for $N > N_0(\epsilon)$ we have:*

$$a) \qquad P\left((SM(E_N, t) < \frac{17t}{4}(N \log N)^{1/2}\right) > 1 - \epsilon, \tag{5.5}$$

$$b) \qquad P\left((SD(E_N, d) < \frac{17}{4\sqrt{d}}(N \log N)^{1/2}\right) > 1 - \epsilon. \tag{5.6}$$

5.2 Proofs

Proof of Theorem 10

We will use the same exponential sum in order to prove both parts of Theorem 10, namely let $E_N = \{e_1, e_2, \ldots, e_N\}$ be the usual N long binary sequence and $f(z) = \sum_{n=1}^{N} e_n z^n$. Throughout the proof $e(\alpha)$ will denote $\exp(2\pi i \alpha)$. By the Cauchy-Schwarz inequality and the Parseval formula we obtain:

$$K \overset{\text{def}}{=} \int_0^1 |f(e(\alpha))|^4 d\alpha \geq \left(\int_0^1 |f(e(\alpha))|^2 d\alpha\right)^2 \geq N^2. \tag{5.7}$$

On the other hand:

$$K=\int_0^1 |f^2(e(\alpha))|^2 d\alpha=\int_0^1\left|\sum_{n=1}^{N}\sum_{m=1}^{N}e_ne_me((n+m)\alpha)\right|^2 d\alpha=$$
$$=\int_0^1\left|\sum_{k=2}^{2N}\left(\sum_{\max\{1,\ k-N\}\le n\le\min\{N,\ k-1\}}e_ne_{k-n}\right)e(k\alpha)\right|^2 d\alpha=$$
$$=\sum_{k=2}^{2N}\left|\sum_{\max\{1,\ k-N\}\le n\le\min\{N,\ k-1\}}e_ne_{k-n}\right|^2. \tag{5.8}$$

Now this sum can be estimated from above by both symmetry measures. First for $2\le k_1,k_2,\dots,\ k_t\le 2N$ such that $k_1,\dots,k_t$ are different, we have

$$\sum_{i=1}^{t}\left|\sum_{\max\{1,\ k_i-N\}\le n\le\min\{N,\ k_i-1\}}e_ne_{k_i-n}\right|^2\le$$
$$\le\left(\sum_{i=1}^{t}\left|\sum_{\max\{1,\ k_i-N\}\le n\le\min\{N,\ k_i-1\}}e_ne_{k_i-n}\right|\right)^2\le(2SM(E_N,t)+t)^2.$$

Which yields

$$N^2\le K=\sum_{k=2}^{2N}\left|\sum_{\max\{1,\ k-N\}\le n\le\min\{N,\ k-1\}}e_ne_{k-n}\right|^2\le$$
$$\le\left\lceil\frac{2N-1}{t}\right\rceil(2SM(E_N,t)+t)^2. \tag{5.9}$$

By (5.7) and (5.9) we can conclude

$$SM(E_N,t)\ge\frac{\sqrt{t}}{2\sqrt{2}}\sqrt{N}-\frac{t}{2}>\frac{7\sqrt{t}}{20}\sqrt{N}-\frac{t}{2}$$

which completes the the proof of Theorem 1 (5.3). Similarly we can estimate (5.8) by using the definition of $SD(E_N,d)$:

$$\left|\sum_{\max\{1,\ k_i-N\}\le n\le\min\{N,\ k_i-1\}}e_ne_{k_i-n}\right|\le(2dSD(E_N,d)+1)^2$$

$$N^2\le\sum_{k=2}^{2N}\left|\sum_{\max\{1,\ k_i-N\}\le n\le\min\{N,\ k_i-1\}}e_ne_{k_i-n}\right|\le(2N-1)(2dSD(E_N,d)+1)^2,$$

from which (5.4) follows. □

Proof of Theorem 11

The following lemma was used in the paper of Cassaigne, Mauduit and Sárközy [14], we shall need it as well. They omit the proof, thus we prove it here.

Lemma 1. *Let k be an integer such that $k \le t^{\alpha}$ where $0 \le \alpha < 2/3$. If $t \to \infty$ then*

$$\binom{t}{[t/2]+k} = \binom{t}{[t/2]} \exp\left(\frac{-2k^2}{t} + O\left(\frac{k^3}{t^2}\right)\right). \tag{5.10}$$

Proof of Lemma 1

We denote by A the fraction of binomials.

$$\begin{aligned} A = \frac{\binom{t}{[t/2]+k}}{\binom{t}{[t/2]}} &= \frac{([t/2]!)^2}{([t/2]-k)!([t/2]+k)!} = \\ &= \frac{([t/2]+1-k)([t/2]+2-k)\dots([t/2]+k-k)}{([t/2]+1)([t/2]+2)\dots([t/2]+k)} = \\ &= \left(1-\frac{k}{[t/2]+1}\right)\left(1-\frac{k}{[t/2]+2}\right)\dots\left(1-\frac{k}{[t/2]+k}\right). \end{aligned} \tag{5.11}$$

Its clear from (5.11) that

$$\left(1-\frac{2k}{t}\right)^k \le A \le \left(1-\frac{2k}{t+2k}\right)^k.$$

It is enough to prove that $A \cdot \exp\left(\frac{2k^2}{t}\right) = \exp\left(O\left(\frac{k^3}{t^2}\right)\right)$ as $t \to \infty$. Let $k = t^{\beta}$,

$$\begin{aligned} \lim_{t\to\infty} A \cdot e^{\frac{2k^2}{t}} = \lim_{t\to\infty} A \cdot e^{2t^{2\beta-1}} &\ge \lim_{t\to\infty} \left(1-\frac{2t^{\beta}}{t}\right)^{t^{\beta}} e^{2t^{2\beta-1}} = \\ &= \lim_{t\to\infty} \left(\left(1-\frac{2t^{2\beta-1}}{t^{\beta}}\right)^{\frac{t^{\beta}}{2t^{2\beta-1}}}\right)^{2t^{2\beta-1}} e^{2t^{2\beta-1}} = \\ &= e^{-2t^{2\beta-1}} e^{2t^{2\beta-1}} = 1, \end{aligned}$$

$$\lim_{t\to\infty} A\cdot e^{\frac{2k^2}{t}} = \lim_{t\to\infty} A\cdot e^{2t^{2\beta-1}} \le \lim_{t\to\infty}\left(1-\frac{2t^\beta}{t+2t^\beta}\right)^{t^\beta} e^{2t^{2\beta-1}} =$$
$$= \lim_{t\to\infty}\left(\left(1-\frac{2t^{2\beta}/(t+2t^\beta)}{t^\beta}\right)^{\frac{t^\beta}{2t^{2\beta}/(t+2t^\beta)}}\right)^{2t^{2\beta}/(t+2t^\beta)} e^{2t^{2\beta-1}} =$$
$$= e^{-2t^{2\beta-1}\frac{1}{1+o(t^\beta/t)}} e^{2t^{2\beta-1}} = e^{\frac{o(t^{3\beta}/t^2)}{1+o(t^\beta/t)}} \le e^{o(t^{3\beta}/t^2)} \le e^{o(t^{3\alpha}/t^2)}.$$

Indeed the above estimates together establish (5.10) and complete the proof of the lemma.

Write $L = 4.25(N\log N)^{1/2}$. Then we have

$$P(SD(E_N,d) > L/\sqrt{d}) = P\left(\max_{a<b}\left|\sum_{j=0}^{[(b-a)/2d]-1} e_{a+dj}e_{b-dj}\right| > \frac{L}{\sqrt{d}}\right) \le$$
$$\le \sum_{a<b} P\left(\left|\sum_{j=0}^{[(b-a)/2d]-1} e_{a+dj}e_{b-dj}\right| > \frac{L}{\sqrt{d}}\right) \le$$
$$\le \binom{N}{2}\max_{a<b} P\left(\left|\sum_{j=0}^{[(b-a)/2d]-1} e_{a+dj}e_{b-dj}\right| > \frac{L}{\sqrt{d}}\right),$$

where $a,b\in\mathbb{N}$ such that $1\le a<b\le N$. It suffices to show that for all such a's and b's we have:

$$P\left(\left|\sum_{j=0}^{[(b-a)/2d]-1} e_{a+dj}e_{b-dj}\right| > L/\sqrt{d}\right) < \frac{2\epsilon}{N^2}. \tag{5.12}$$

Let $l = [(b-a)/2d]$. If $l \le L/\sqrt{d}$ then the probability in (8) is zero so we may assume that $l > L/\sqrt{d}$. Write

$$M \overset{\text{def}}{=} 6(l\log l)^{1/2} \tag{5.13}$$

and

$$|\{j:\ 0\le j\le l-1,\ e_{a+dj}e_{b-dj} = -1\}| = h. \tag{5.14}$$

Then we have

$$\sum_{j=0}^{l-1} e_{a+dj}e_{b-dj} = |\{j:\ 0 \le j \le l-1,\ e_{a+dj}e_{b-dj} = 1\}| -$$

$$- |\{j:\ 0 \le j \le l-1,\ e_{a+dj}e_{b-dj} = -1\}| =$$

$$= (l-h) - h = l - 2h.$$

The number of $2l$-long sequences for which (5.14) holds is $\binom{l}{h}2^h 2^{l-h} = \binom{l}{h}2^l$. So the probability of (5.14) is $\frac{1}{2^l}\binom{l}{h}$. Since $M \le L/\sqrt{d}$ it is enough to consider

$$P\left(\left|\sum_{j=1}^{l-1} e_{a+dj}e_{b-dj}\right| > M\right) = \sum_{h:\ |l-2h|>M} \frac{1}{2^l}\binom{l}{h} = \frac{1}{2^l}\sum_{h:\ |h-l/2|>M/2}\binom{l}{h}. \tag{5.15}$$

If N is large enough so is l since $l > L/\sqrt{d}$. Using Lemma 1 we can estimate the above sum.

$$\sum_{h:\ |h-l/2|>M/2}\binom{l}{h} = \sum_{h:\ |h-l/2|>3(l\log l)^{1/2}}\binom{l}{h} < l\binom{l}{[l/2]+[3(l\log l)^{1/2}]} \le$$

$$\le l\binom{l}{[l/2]}\exp\left(-2(3(l\log l)^{1/2})^2\frac{1}{l} + o(1)\right) <$$

$$< l\binom{l}{[l/2]}\exp\left(-18\log l + o(1)\right) < \frac{2^l}{l^{16}}. \tag{5.16}$$

Finally by (5.13) and (5.16) we conclude

$$P\left(\left|\sum_{j=0}^{l-1} e_{a+dj}e_{b-dj}\right| > L/\sqrt{d}\right) \le P\left(\left|\sum_{j=0}^{l-1} e_{a+dj}e_{b-dj}\right| > M\right) <$$

$$< \frac{1}{2^l}\frac{2^l}{l^{16}} < \left(\frac{\sqrt{d}}{L}\right)^{16} = o\left(\frac{1}{N^8}\right) < \frac{2\epsilon}{N^2},$$

which proves (5.12) and completes the second part of Theorem 2 (5.6). The first part immediately follows from the fact that $SD(E_N,d)\cdot t\cdot d \ge SM(E_N,t)$ is true for every d. In particular when $d=1$

$$P(SM(E_N,t) > tL) \le P(tSD(E_N,1) > tL) < \epsilon.$$

$\square$

5.3 Constructions

To justify the newly introduced generalized symmetry measures, we now show a sequence which passes the usual statistical tests even though it has a deeply symmetric structure.

Theorem 12. *There exists a sequence for which $S(E_N)$ is small, but $SD(E_N, 2)$ is big, that is*

$$S(E_N) \leq 54N^{1/2}\log N$$
$$N/4 \leq SD(E_N, 2) \leq N/2$$

Proof of Theorem 12

For the proof we will use two lemmas.

Lemma 2. *If p is a prime number, $f(x) \in F_p[x]$ is a polynomial of degree k such that it is not in the form $f(x) \in b(g(x))^2$ with $b \in F_p$, $g(x) \in F_p[x]$, and X, Y are real numbers with $0 < Y \leq p$ then writing*

$$\chi_p^*(n) = \begin{cases} \left(\frac{n}{p}\right) & for\ (n,p) = 1 \\ 0 & for\ p|n \end{cases}$$

we have

$$\left| \sum_{X<n\leq X+Y} \chi_p^*(f(n)) \right| < 9kp^{1/2}\log p.$$

For the proof see [13]. Indeed there this result is deduced from Weil's theorem [20]. Also we need the following result of Gyarmati we mentioned previously.

Lemma 3. *The first $\frac{p-1}{2}$ elements of E_{p-1} have small symmetry measure. That is*

$$S(E_{\frac{p-1}{2}}) < 18p^{1/2}\log p,$$

where $E_{\frac{p-1}{2}} = \left\{ \left(\frac{1}{p}\right), \left(\frac{2}{p}\right), \ldots, \left(\frac{(p-1)/2}{p}\right) \right\}$

For the proof see [3].

Now we can construct a sequence which has the required property. We define $\widehat{E}_{p-1} = (e_1, e_2, \ldots, e_j, \ldots, e_{p-1})$ where

$$e_j = \begin{cases} \left(\frac{j}{p}\right), & \text{if } 1 \leq j \leq \frac{p-1}{2} \\ \left(\frac{p-j}{p}\right), & \text{if } \frac{p+1}{2} \leq j \leq p-1, \quad j \text{ is even} \\ -\left(\frac{p-j}{p}\right), & \text{if } \frac{p+1}{2} \leq j \leq p-1, \quad j \text{ is odd} \end{cases}$$

In this way the second part of the sequence is the reflection of the first part except that we changed the sign of every second term. We will show that $\widehat{E}_{p-1}$ has small $S(\widehat{E}_{p-1})$ although $SD(\widehat{E}_{p-1}, 2)$ is huge. Indeed it is easy to check the second statement. Using (5.2) we obtain:

$$\left| \sum_{j=0}^{[(p-1)/4]-1} e_{1+2j} e_{p-2j} \right| = \frac{p-1}{4} \leq SD(E_{p-1}, 2). \tag{5.17}$$

To prove the first statement we have to check that $H(\widehat{E}_{p-1}, a, b)$ is small for every a and b such that $1 \leq a < b \leq p$.

I. If $b \leq \frac{p-1}{2}$ we can directly apply Lemma 3 and obtain

$$S(\widehat{E}_{p-1}) = \max_{a<b\leq(p-1)/2} \left| H(\widehat{E}_{p-1}, a, b) \right| < 18p^{1/2} \log p. \tag{5.18}$$

II. If $a > \frac{p-1}{2}$ we have a very similar case, except here the sign of every second term was changed. We denote the subsequence of the first $(p-1)/2$ element of $\widehat{E}_{p-1}$ by $E_{\frac{p-1}{2}}$ and the subsequence of the remaining $(p-1)/2$ element by $\widehat{E}_{\frac{p-1}{2}}$. Observe that for every given a and b the magnitude of the sum

$$H(\widehat{E}_{\frac{p-1}{2}}, a, b) = \sum_{j=0}^{[(b-a)/2]-1} e_{a+j} e_{b-j}$$

is the same as $H(E_{\frac{p-1}{2}}, \frac{p-1}{2} + 1 - b, \frac{p-1}{2} + 1 - a)$ up to sign. Hence,

$$\left| H(\widehat{E}_{\frac{p-1}{2}}, a, b) \right| = \left| H(E_{\frac{p-1}{2}}, \frac{p-1}{2} + 1 - b, \frac{p-1}{2} + 1 - a) \right| < 18p^{1/2} \log p. \tag{5.19}$$

III.The remaining case is where $a \leq \frac{p-1}{2}$ and $b > \frac{p-1}{2}$.

a) For $a = p - b$ we have $S(\widehat{E}_{p-1}) = 0$ by construction.

b) If $a > p - b$ then

$$\begin{aligned} \left| \sum_{j=0}^{[\frac{b-a}{2}]-1} e_{a+j} e_{b-j} \right| &= \left| \sum_{j=0}^{\frac{p-1}{2}-a} e_{a+j} e_{b-j} + \sum_{j=\frac{p+1}{2}-a}^{[\frac{b-a}{2}]-1} e_{a+j} e_{b-j} \right| \leq \\ &\leq \left| \sum_{j=0}^{\frac{p-1}{2}-a} e_{a+j} e_{b-j} \right| + 18p^{\frac{1}{2}} \log p. \end{aligned} \tag{5.20}$$

In this way we split the sum into a first (outer) and a second (inner) part. The inner part of the sum can be estimated by (5.19), so we only need to deal with the outer part.

$$\left|\sum_{j=0}^{\frac{p-1}{2}-a} e_{a+j}e_{b-j}\right| = \left|\sum_{j=0,\text{ j even}}^{\frac{p-1}{2}-a}\left(\frac{a+j}{p}\right)\left(\frac{p-b+j}{p}\right) - \sum_{j=1,\text{ j odd}}^{\frac{p-1}{2}-a}\left(\frac{a+j}{p}\right)\left(\frac{p-b+j}{p}\right)\right| \le$$

$$\le \left|\sum_{k=0}^{[\frac{p-1}{4}-\frac{a}{2}]}\left(\frac{a+2k}{p}\right)\left(\frac{p-b+2k}{p}\right)\right| + \left|\sum_{k=0}^{[\frac{p-1}{4}-\frac{a}{2}]}\left(\frac{a+2k+1}{p}\right)\left(\frac{p-b+2k+1}{p}\right)\right| \le$$

$$\le \left|\sum_{k=0}^{[\frac{p-1}{4}-\frac{a}{2}]}\left(\frac{(2k)^2+(a-b)2k-ab}{p}\right)\right| + \left|\sum_{k=0}^{[\frac{p-1}{4}-\frac{a}{2}]}\left(\frac{(2k+1)^2+(a-b)(2k+1)-ab}{p}\right)\right|.$$

Let $f_1(x) = 4x^2+2(a-b)x-ab \in \mathbb{F}_p[x]$ and $f_2(x) = 4x^2+2(b-a+1)x+ab+a-b+1 \in \mathbb{F}_p[x]$. It is easy to check that both $f_1(x)$ and $f_2(x)$ can be written as $b_1(g_1(x))^2$ and $b_2(g_2(x))^2$ respectively if and only if $a+b \equiv 0 \pmod p$. Now this is impossible since we assumed that $a > p-b$. Applying Lemma 2 twice with 0 and $[\frac{p-1}{4}-\frac{a}{2}]$ in place of X and Y we get

$$\left|\sum_{j=0}^{\frac{p-1}{2}-a} e_{a+j}e_{b-j}\right| = \left|\sum_{k=0}^{[\frac{p-1}{4}-\frac{a}{2}]}\left(\frac{f_1(k)}{p}\right)\right| + \left|\sum_{k=0}^{[\frac{p-1}{4}-\frac{a}{2}]}\left(\frac{f_2(k)}{p}\right)\right| =$$

$$= \left|\sum_{X\le k\le X+Y}\chi_p^*(f_1(k))\right| + \left|\sum_{X\le k\le X+Y}\chi_p^*(f_2(k))\right| \le$$

$$\le 18p^{1/2}\log p + 18p^{1/2}\log p = 36p^{1/2}\log p. \tag{5.21}$$

By (5.20) and (5.21) we obtain that $S(\widehat{E}_{p-1}) \le 54p^{1/2}\log p$.

c) For $a < p-b$ we can repeat the previous argument by symmetry. Again we split the sum. The inner part can be estimated by (5.18) and for the outer part we obtain the same result as in (5.21).

Now by (5.18), (5.19) and (5.21) we gather

$$S(\widehat{E}_{p-1}) \le 54p^{1/2}\log p,$$

which together with (5.17) completes the proof of Theorem 3. □

Now we show an example for a sequence E_N for which $SD(E_N,d)$ is small for every possible d.

Theorem 13. *If p is an odd prime and we write*

$$E_{(p-1)/2} = \left(\left(\frac{1}{p}\right), \left(\frac{2}{p}\right), \dots, \left(\frac{(p-1)/2}{p}\right) \right)$$

then we have

$$SD(E_{(p-1)/2}, d) \le 18p^{1/2} \log p$$

for every $d \in \mathbb{N}$.

Proof of Theorem 13

By definition

$$SD(E_{\frac{p-1}{2}}, d) = \max_{a<b} \left| \sum_{j=0}^{[(b-a)/2d]-1} e_{a+dj} e_{b-dj} \right|.$$

We have to estimate the sum for all possible a's and b's.

$$\begin{aligned} \left| \sum_{j=0}^{[(b-a)/2d]-1} e_{a+dj} e_{b-dj} \right| &= \sum_{j=0}^{[(b-a)/2d]-1} \left(\frac{a+dj}{p}\right) \left(\frac{b-dj}{p}\right) = \\ &= \sum_{j=0}^{[(b-a)/2d]-1} \left(\frac{-(dj)^2 + (b-a)dj + ab}{p}\right). \end{aligned} \tag{5.22}$$

Let $f(x) = -d^2x^2 + (b-a)dx + ab \in \mathbb{F}_p[x]$. It is easy to see that $f(x)$ is the form of $b(g(x))^2$ if and only if $a + b \equiv 0 \pmod p$. In the present case this is impossible, since $1 \le a < b \le (p-1)/2$. Applying Lemma 2 with 0 and $(b-a)/2d$ in place of X and Y we get

$$\left| \sum_{X \le j \le X+Y} \chi_p^*(f(j)) \right| = \sum_{j=0}^{[(b-a)/2d]-1} \left(\frac{-(dj)^2 + (b-a)dj + ab}{p}\right) < 18p^{1/2} \log p. \tag{5.23}$$

From (5.22) and (5.23) we obtain $S(E_{(p-1)/2}) < 18p^{1/2} \log p$ which gives the desired result. $\square$

Chapter 6

Conclusions

"It's hard to make predictions - especially about the future." Robert Storm Petersen

We established definitions of randomness and random sequences in order to understand the notion of pseudorandomness. With the very weak assumption that one-way functions exists we proved the existence of pseudorandom number generators. We have shown that it is not easy for a generator to fulfill all the requirements the different applications need. Statistical tests are implemented to determine whether a generator is suitable for its task or not. There are two different kind of approach regarding statistical tests. A posteriori testing means that we take a sample output sequence of the PRNG and examine if it exhibits such characteristics that we would expect from a random sequence. Theoretical or a priori testing means we analyze the PRNG to learn something on the distribution of it's output.

The correlation and well-distribution measures proved to be very successful tools in a priori testing. However these measures are not sensitive to symmetric patterns. A symmetry measure was introduced to detect long symmetrical subsequences in binary sequences. Since there are other possible symmetrical patterns we proposed two ways to generalize the symmetry measure.

One way is to check whether the sequence shows symmetric patterns in the form of an arithmetic progression. The other is to consider multiple symmetry centers when we search for symmetric subsequences. We introduced $SD(E_N, d)$ and $SM(E_N, t)$ for these two tasks respectively. They are indeed a generalization of the symmetry measure, since they produce the same value as $S(E_N)$ for $d = 1 = t$.

We gave upper and lower bounds to these generalized measures. In addition to that we provided an example to prove that $SD(E_N, 2)$ and $S(E_N)$ are independent of each other. The same argument shows that if $(d_1, d_2) = 1$ then $SD(E_N, d_1)$ and $SD(E_N, d_2)$

are unrelated. It seems plausible that similar constructions may work for $SM(E_N, t)$, however proving it is much harder. A possible future research problem is to construct sequences for which $S(E_N)$ small, but $SM(E_N, t)$ is big.

It remains to be shown that these two measures are as useful in theoretical testing as the correlation a well-distribution measures. As we have seen there are countless of other tests. There is no need to introduce a new criteria unless we are either able to trace back many pseudorandom characteristics to this single measure or we can construct large families of pseudorandom sequences with the help of it. Further research is needed to verify that $SD(E_N, d)$ and $SM(E_N, t)$ fulfill these requirements.

References

[1] P. van Oorschot A. Menezes and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[2] BusinessDictionary.com.
$http://www.businessdictionary.com/definition/information-society.html.$

[3] Katalin Gyarmati. On a pseudorandom property of binary sequences. *The Ramanujan Journal*, Vol. 8:89–302, 2004.

[4] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* Chapman and Hall/CRC, 2007.

[5] Donald E. Knuth, editor. *Art of computer programming, Volume 2: Seminumerical Algorithms.* Addison-Wesley, Reading, Massachusetts, 1981.

[6] M. Blum L. Blum and Michael Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15:364–383, 1986.

[7] L. Levin. On the notion of a random sequence. *Soviet Mathematics Doklady*, 14:1413–1416, 1973.

[8] M. Li and P. Vitanyi. *Introduction to Kolmogorov Complexity and Its Applications.* Springer Science, 1993.

[9] G. Marsaglia. Monkey tests for random number generators. *Computers and Mathematics with Applications*, 9:1–10, 1993.

[10] G. Marsaglia. On the randomness of pi and other decimal expansions. *Interstat*, 5, 2005.

[11] P. Martin-Löf. The definition of random sequences. *Inform. and Control*, 6:602–619, 1966.

[12] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8 (1):3–30, 1998.

[13] C. Mauduit and A. Sárközy. On finite pseudorandom binary sequences i: measure of pseudorandomness, the legendre symbol. *Acta Arithmetica*, 82:365–377, 1997.

[14] J. Cassaigne C. Mauduit and A. Sárközy. On finite pseudorandom binary sequences vii: The measure of pseudorandomness. *Acta Arithmetica*, 103, 2002.

[15] L. Goubin C. Mauduit and A. Sárközy. Construction of large families of pseudorandom binary sequences. *Journal of Number Theory*, 106:56–69, 2004.

[16] Ivan Niven. *Irrational Numbers.* The Mathematical Association of America, New Jersey, 1956.

[17] National Institute of Standards and Technology. Statistical testing of random number generators. *http://csrc.nist.gov/groups/ST/toolkit/rng/documents/nissc-paper.pdf.*

[18] J. Rivat and A. Sárközy. On pseudorandom sequences and their applications. *Information Transfer and Combinatorics, Lecture Notes in Computer Science*, 4123:343–361, 2006.

[19] C. P. Schnorr. A unified approach to the definition of a random sequence. *Mathematical Systems Theory*, 5:246–258, 1971.

[20] A. Weil. Sur les courbes algébriques et les veriétés qui s'en déduisent. *Acta Sci. Ind.*, 1041, Hermann, Paris 1948.

[21] Wikipedia.com.
http : //en.wikipedia.org/wiki/algorithmically_random_sequence.