On the Complexity of Pseudo-Random Sequences - or: If You Can Describe a Sequence It Can't be Random

Thomas Beth Zong-Duo Dai[1]

Europäisches Institut für Systemsicherheit (EISS)
Universität Karlsruhe (T.H.)
D – 7500 Karlsruhe, F.R. of Germany

"*Any one who considers automatical methods of producing random digits is, of course in a state of sin*"

(*J.v.Neumann*)[15]

Abstract

We shall prove in this note that the Turing-Kolmogorov-Chaitin complexity and the Linear Complexity are the same for practically all 0-1-sequences of length n, already for moderately large n.

1 Introduction

The availability of reproducible random - looking sequences is a precondition for many applications of modern cryptology, cf.[4,1,13].

Such sequences have to be generated at the sender's and receiver's side by deterministic automata. For this reason the generated sequences are called *Pseudo-Random Sequences* rather than random sequences. A central topic of research is the problem of determining the complexity of a given sequence S of length n, $n \in \mathbf{N}$, over a finite alphabet[13,14]. The view

[1]On leave from Academia Sinica,Beijing, China

J.J. Quisquater and J. Vandewalle (Eds.): Advances in Cryptology - EUROCRYPT '89, LNCS 434, pp. 533-543, 1990.

generally generated in approaching this problem is that of a cryptanalyst who, after a plain-text attack [4,1] having obtained a long random-looking string, has to determine a generating automaton A of size α as small as possible, reproducing S from a so-called seed key k as short as possible. The complexity of the sequence S is then the size α plus the length of the seed key k, which is widely considered as a measure of security of the sequence. A common practice is to choose the generating automaton A from the class $\mathcal{L}$ of linear feedback shift registers (LFSR)[1,13,8]. The so-called Berlekamp-Massey-Algorithm [2,3,5,10] gives a fast procedure to construct a generating LFSR of minimal so-called linear complexity for a given sequence of length n.

Amongst those, who have been working in the area of sequence complexity it has been known that the linear complexity measure λ is only one way of bounding the security from above, which sometimes seems to be far out of the practical relevance, as some well chosen examples show, cf.[7]. In order to overcome such difficulties other complexity measures have been considered, to describe the complexity of sequences and sequence generators, cf.[14].

To overcome the difficulties inherent to approaching the complexity problem within a fixed class of automata an alternative way is to consider the wide class of Turing machines (TM), as a universal class of generators instead. As any Turing machine (TM) can be obtained from programming a universal Turing machine (UTM), the length $\mathcal{X}$ of a shortest program for a UTM making it produce a given sequence S is called the Turing-Kolmogorov-Chaitin-complexity of this sequence. In the remainder of this paper we will show that for the complexity measures the relation $\lambda \sim \mathcal{X}$ holds for all but a set of arbitrary low probability of (0-1)-sequences of length n with growing n. This result can be generalized to arbitrary alphabets.

2 Basic Notations and Facts

In the following technical sections we will make use of the standard concepts of algorithms and complexity, cf. [9], which will lead to the notion of automata.

Let n be a positive integer and $2^n = \{0,1\}^n$ denote the set of all binary strings of length n, whereas $2^* = \{0,1\}^* = \bigcup_{n=0}^{\infty} 2^n$ denotes the set of all sequences over the alphabet $\{0,1\}$.

A deterministic automaton M with binary input/output is given as usually as $M = (S, \Sigma, \Omega, \delta, \alpha, \lambda)$, cf. [9] where

S is the set of states with a special initial state α. The input alphabet I and the output alphabet O coincide with $\Sigma = \{0, 1\}$. $\delta : S \times I \to S$ is the next-state-function, which is naturally extended to a mapping $\delta^{(n)} : S \times I^n \to S$ by setting

$$\begin{aligned} \delta(s; \underline{\varepsilon}) &:= \delta^n(s; \varepsilon_{n-1}, ..., \varepsilon_0) \\ &= \delta^{n-1}(\delta(s; \varepsilon_0); \varepsilon_{n-1}, ..., \varepsilon_1) \\ &\qquad \text{for } s \in S \text{ and } \underline{\varepsilon} = (\varepsilon_{n-1}, \cdots, \varepsilon_0) \in I^n. \end{aligned}$$

$\lambda : \Omega \to O$ denotes the output function, which generates an output, whenever the state has reached a state in the set of final states $\Omega \subset S$.

A state $s \in S$ can be reached from a state $t \in S$, if there exists a sequence of inputs $\underline{\varepsilon} \in \sigma^*$ such that

$$t = \delta(s, \underline{\varepsilon}).$$

An output sequence $\underline{x} \in 2^n$ is said to be *generated by* M iff there exists a sequence $(\omega_1, \omega_2, \cdots, \omega_n) \in \Omega^n$ in which ω_{i+1} can be reached from ω_i for $i = 1, 2, \cdots, n-1$ and where

$$\lambda(\omega_j) = x_j, \qquad \text{for } j \in [1, 2, \cdots, n].$$

The set of all sequences $\underline{x} \in O^*$ generated by M will be denoted by $A(M)$, and $A_n(M) = A(M) \cap \Sigma^n$.

The machine is then called a *sequence generator* for all $n \in \mathbb{N}$.

Let $\mu(M)$ denote the complexity of describing the finite state machine M, i.e. $\mu(M) = \text{size}(\delta) + \log_2(|S|)$ where the size (δ) is the number of bits needed to completely specify the next-state-function δ. Let $\mathcal{M}$ be a class of sequence generators (of a certain type). Let

$$\mu_{\mathcal{M}} : O^n \longrightarrow \mathbb{N} \cup \{\infty\}$$

be given by

$$\mu_{\mathcal{M}}(\underline{x}) = \min\{\mu(M) | M \in \mathcal{M}, \underline{x} \in A_n(M)\}.$$

If $\mu_{\mathcal{M}}(\underline{x}) \in \mathbf{N}$ we say that the output sequence $\underline{x}$ with respect to the machine type $\mathcal{M}$ affords a description of length $\mu_{\mathcal{M}}(\underline{x})$.

3 The Well-Known Case : Linear Sequence Generators

Let $\mathcal{L}$ be the class of linear feedback shift registers (LFSR), i.e. the class of linear recursions over $GF(2)$. Each linear recursion L is uniquely described by the feedback polynomial $q(z) \in GF(2)[z]$ and an initial state $(x_0, \ldots, x_{l-1})$, where l = deg q(x). The sequence of length n generated by the linear automaton L is $(x_0, \ldots, x_{l-1}, x_l, \ldots, x_{n-1})$, where

$$x_{t+l} = \sum_{i=0}^{l-1} c_i \cdot x_{t+i}, 0 \leq i < n - l$$

and $q(x) = x^l + \sum_0^{l-1} c_1 x^i$.

The complexity $\mu(L)$ of describing L within the class $\mathcal{L}$ of LFSR is $\mu(L) = 2 \cdot l$.

For a given sequence $\underline{x} \in 2^n$ we denote by

$$\lambda(\underline{x}) := \mu_{\mathcal{L}}(\underline{x})$$

the *linear complexity* of the sequence $\underline{x}$, which in other words is twice the length of the shortest linear feedback shift register which could have "produced" $\underline{x}$.

Theorem 3.1 *For any given $\underline{x} \in 2^n$ the Berlekamp-Massey-algorithm will compute $\lambda(\underline{x}), p(z)$, and $q(z)$ in $O(n^2 \log n)$ bit operations.*

Proof. This is the well-known result from shift-register analysis, cf. [5,10,13].

□

Theorem 3.2 *Let $k \in [0:n]$. Let $N_n(k) := |\{\underline{s} \in 2^n | \lambda(\underline{s}) = 2k\}|$. Then*

$$N_n(k) = \begin{cases} 2^{\min(2n-2k, 2k-1)} & \text{if } n \geq k > 0 \\ 1 & \text{if } k = 0. \end{cases}$$

Proof. [13].

4 The Most General Case: Turing Machine As Sequence Generator

Let $\mathcal{T}$ be the class of Turing machines (TM).

A Turing Machine TM can be considered as a finite state machine with an additional infinite tape as work space of tape symbols $Z = \{0, 1, \#\}$, which can be read from or printed to via a tape head that itself is controlled by the finite state control through move-commands $+1$(left), -1(right) or 0(stop). The formal description extends that of a finite state machine

$$T = (S, I, Z; \delta, \alpha, \Omega).$$

The next-state function δ is extended to a next-move function

$$\delta : S \times I \times Z \rightarrow S \times (Z \times \{\mathrm{Z}_3\})$$

which from a given state $\sigma \in S$, an input symbol $i \in I$ and a read symbol $z \in Z$ computes

$$\delta(\sigma, i, z) = (\sigma', z', m)$$

i.e. it changes the state from σ into σ', write a new tape symbol and moves the tape head by $m \in \mathrm{Z}_3$ (left, right or not at all). The work tape is upon start of the TM empty, i.e. filled with blanks #.

The concept of a Turing Machine can now be used to construct the so-called Universal Turing Machine (UTM), which in essence is a general purpose Turing Machine capable of simulating any other TM as given above.

The Universal Turing Machine is a Turing Machine U whose finite state control is designed such that it generates output strings $\underline{x} \in \{0, 1\}^*$ of the form $\underline{y} = (\underline{x}, \underline{p}(T))$ where $\underline{p}(T)$ is an input equal to the binary program which makes U behave like the Turing machine T generating x.

Theorem 4.1 *Any Turing machine T can be simulated in this form by a universal Turing machine U.*

Proof. [9]

Definition 4.2 *Let $\underline{x} \in 2^n$ be a binary sequence. Let T be a given Turing machine generating x. Let $\underline{p}(T)$ be the program , which makes this UTM simulate T.*

Then the number

$$\mu(T) := |\underline{p}(T)|$$

i.e. the length $|\underline{p}|$ of the program $\underline{p}$ is called the size of T.

The number

$$\mathcal{X}(\underline{x}) := \mu_{\mathcal{T}}(\underline{x}) = \min\{\mu(T) | T \text{ generates } \underline{x}, T \in \mathcal{T}\}$$

is called the Turing- Kolmogorov-Chaitin complexity *of the sequence $\underline{x}$.*

Theorem 4.3 *The function $\mathcal{X}$ generally is not computable (by a Turing machine)*

Proof. Suppose $\mathcal{X}$ is computable. Construct a Turing Machine T_k which generates and inspects binary sequences in say lexicographical order until a sequence $\underline{x}$ of complexity $\mathcal{X}(\underline{x}) > K$ larger than the constant K is found and then accepts this sequence $\underline{x}$.

The size of this machine is given by

$$\mu(T) = O(\log K) ,$$

so that the fact $\mu(T) < K$ for large K creates a contradiction.

□

In spite of this result we derive the following result on the average behaviour of $\mathcal{X}$:

Theorem 4.4 *Let $k \in [1 : n]$. Then*

$$|\{\underline{s} \in 2^n | \mathcal{X} \leq k\}| \leq 2^{k+1} .$$

Proof. Amongst the 2^{k+1} Turing machines T with $\mu(T) \leq k$ there are at most 2^{k+1} sequence generators.

□

Corollary 4.5 *On the space 2^n of all 0-1-sequences with equidistributed probability the following holds for all ε, $0 < \varepsilon < 1$:*

$$\text{Prob}\{\underline{s} \in 2^n | \mathcal{X}(\underline{s}) > (1-\varepsilon)n\} > 1 - 2^{-\varepsilon n+1} .$$

This corollary shows that practically all sequences of moderately large length have a Turing-Kolmogorov-Chaitin complexity close to the length of the sequence.

In other words this is intuitively clear:

A "true" random sequence of length n has no shorter description than just the sequence itself.

In the next section we shall investigate the relation between the TKC-complexity measure $\mathcal{X}$ and the linear complexity λ.

5 The Main Result

Theorem 5.1 *For any real number ε, $0 < \varepsilon < 1$, arbitrary integer n, and an arbitrary string $\underline{s} \in 2^n$, we have*

1. $\text{Prob}\{(1-\varepsilon)\lambda(\underline{s}) \leq \mathcal{X}(\underline{s})\} \geq 1 - \frac{8}{3}2^{-\frac{\varepsilon}{2-\varepsilon}n}$,
2. $\text{Prob}\{(1-\varepsilon)\mathcal{X}(\underline{s}) \leq \lambda(\underline{s})\} \geq 1 - \frac{1}{3}2^{-\varepsilon n+(1-\varepsilon)(1+\log n)+1} - \frac{1}{3}2^{-n}$

Lemma 5.2 *For any positive real numbers t and u, such that*

$$n + t < 2n, \qquad u < n,$$

we have

1. $\#\{s \in 2^n | n + t < \lambda(s)\} \leq \frac{1}{3}2^{n-t+1} - \frac{1}{3}$,
2. $\#\{s \in 2^n | \mathcal{X}(s) < n - u\} \leq 2^{n-u+1}$,
3. $\#\{s \in 2^n | \lambda(s) \leq n + t, n - u \leq \mathcal{X}(s)\} \geq 2^n - \frac{1}{3}2^{n-t+1} - 2^{n-u+1}$,
4. $Prob\{\lambda(s) \leq n + t, n - u \leq \mathcal{X}(s)\} \geq 1 - \frac{1}{3}2^{-t+1} - 2^{-u+1}$.

Proof.

The assertion "2." and the proposition "1. and 2. $\Rightarrow$ 3. $\Rightarrow$ 4." are trivial. Thus we need only to prove 1. By Theorem 3.2 we have

$$\begin{aligned}
\#\{s \in 2^n | n+t < \lambda(s)\} &= \sum_{n+t<2i\leq 2n} 2^{2n-2i} \\
&= \sum_{\substack{\lambda=\lfloor \frac{n+t}{2} \rfloor \\ 2\lambda+2\leq 2i\leq 2n}} 2^{2n-2i} \\
&= 2^{2n-2\lambda-2} \sum_{0\geq j\geq n-\lambda-1} 2^{-2j} \\
&= 2^{2n-2\lambda-2} \cdot \frac{1-2^{-2(n-\lambda)}}{3/4} \\
&= \frac{1}{3} \cdot 2^{2n-2\lambda} - \frac{1}{3} \\
&\leq \frac{1}{3} \cdot 2^{n-t+1} - \frac{1}{3}.
\end{aligned}$$

□

Lemma 5.3 *For any positive real number t, we have*

1. $\#\{s \in 2^n | 0 \leq \lambda(s) < n-t\} \leq \frac{1}{3} \cdot 2^{n-t+1} + \frac{1}{3}$,
2. $Prob\{n-t \leq \lambda(s)\} \geq 1 - \frac{1}{3}2^{-t+1} - \frac{1}{3} \cdot 2^{-n}$.

Proof. 2. is the consequence of 1. We need only to prove 1.

$$\begin{aligned}
\#\{s \in 2^n | 0 \leq \lambda(s) < n-t\} &= 1 + \sum_{1\leq 2i<n-t} 2^{2i-1} \\
&= 1 + \sum_{\substack{\lambda=\lceil \frac{n-t}{2} \rceil \\ 2\leq 2i\leq 2\lambda-2}} 2^{2i-1} \\
&= 1 + 2 \sum_{0\leq i\leq \lambda-2} 2^{2j} \\
&= 1 + 2 \cdot \frac{2^{2(\lambda-1)}-1}{3} \\
&\leq \frac{1}{3} \cdot 2^{n-t+1} + \frac{1}{3}.
\end{aligned}$$

□

Proof (of the Theorem 5.1)

1. Take $t := \frac{\varepsilon}{2-\varepsilon}n$, then $(1-\varepsilon)(n+t) = n-t$. Now we have

$$\begin{aligned} & \text{Prob}\{(1-\varepsilon)\lambda(s) \leq \mathcal{X}(s)\} \\ \geq\ & \text{Prob}\{\lambda(s) \leq n+t, (1-\varepsilon)(n+t) \leq \mathcal{X}(s)\} \\ =\ & \text{Prob}\{\lambda(s) \leq n+t, n-t \leq \mathcal{X}(s)\} \\ \geq\ & 1 - \frac{1}{3}\cdot 2^{-t+1} - 2^{-t+1} \\ =\ & 1 - \frac{8}{3}\cdot 2^{\frac{-\varepsilon}{2-\varepsilon}n} \qquad \text{by Lemma5.2.} \end{aligned}$$

2. Take $t = \varepsilon n - (1-\varepsilon)\lceil \log n \rceil$, then $(1-\varepsilon)(n + \lceil \log n \rceil) = n - t$.

$$\begin{aligned} & \text{Prob}\{(1-\varepsilon)\mathcal{X}(s) \leq \lambda(s)\} \\ \geq\ & \text{Prob}\{\mathcal{X}(s) \leq n + \lceil \log n \rceil, (1-\varepsilon)(n+\lceil \log n \rceil) \leq \lambda(s)\} \\ =\ & \text{Prob}\{n - t \leq \lambda(s)\} \\ \geq\ & 1 - \frac{1}{3}2^{-\varepsilon n + (1-\varepsilon)(1+\log n)+1} - \frac{1}{3}2^{-n} \quad \text{(by lemma 5.3)} \end{aligned}$$

□

The effect of theorem 5.1 can be written in simplified form as

Corollary 5.4 *For all* ε $(0 < \varepsilon < 1)$

$$P_{\varepsilon,n} := \text{Prob}\{\underline{s} \in 2^n | (1-\varepsilon)\cdot 2\lambda(\underline{s}) \leq \mathcal{X}(\underline{s}) \leq (1+\varepsilon)\cdot 2\lambda(\underline{s})\} \to 1$$

when $n \to \infty$.

Numerical example 5.5
For $\varepsilon = 0.01$ we have for sequence lengths n.

n	2000	4000
$P_{\varepsilon,n}$	0.998	0.999997.

6 Infinite Sequences

Following Niederreiter [12] we denote by $(\Omega, \mathcal{F}, m, \tau)$ the dynamical system on $\Omega = \{0,1\}^\infty$ with the one–sided Bernoulli–shift τ on the product space with the unique Haar–measure m (identical to the product measure).

For $\underline{s} \in \Omega$ and $m, n \in \mathbb{N}$ with $m \leq n$ we denote $\underline{s}_m^n = (s_m, \ldots, s_n)$.

Especially let $\underline{s}^n := \underline{s}_0^n$.

Then we formulate the above results in terms of probability as follows.

Theorem 6.1 *For* $\underline{s} \in \Omega$

$$\lim_{n\to\infty} \frac{\mathcal{K}(\underline{s}^n)}{\lambda(\underline{s}^n)} = 2$$

m-almost everywhere.

Proof. Apply the Borel–Cantelli lemma (cf. [6]) to the independant cylinder sets

$$A_{k,\varepsilon} := \{\underline{s} \in \Omega | (1-\varepsilon) \cdot 2\lambda(\underline{s}_{2^{k-1}}^{2^k-1}) \leq \mathcal{K}(\underline{s}_{2^{k-1}}^{2^k-1}) \leq (1+\varepsilon) \cdot 2\lambda(\underline{s}_{2^{k-1}}^{2^k-1})\}$$

for $k \in \mathbf{N_0}$ and $\varepsilon > 0$.

From corollary 5.4. we conclude that

$$\sum_{k=1}^{\infty} m(A_{k,\varepsilon}) = \infty$$

for all $k \in \mathbf{N_0}$ and $\varepsilon > 0$.

Thus the assertion.

□

We are indebted to Prof. Harald Niederreiter for suggesting this formulation of our results.

7 Conclusions

The results of section 6 show that for almost all sequences, which have a short Turing Machine description, cannot be considered as random sequences.

The introduction of the Turing machine concept shows as a consequence that there is no way by which a large subset of truly random looking sequences in 2^n can be generated by a comparatively small finite state machine. In other words, the results show that the use of so called non-linear generators is not gaining much in terms of the complexity of sequences, if the complexity of the generators is evaluated properly.

The implications to the design of new, highly complex secure encipherment algorithms cf. [11] will have to be investigated further, especially in view of circuit or VLSI complexity.

References

[1] Beker, H., Piper, F.: *Cipher Systems*, Northwood, 1982.

[2] Berlekamp, E.R.: *Coding Theory*, McGraw–Hill, 1968.

[3] Beth, T., Gollmann, D.: *Vorlesungsmanuskript: Signale, Codes und Chiffren*, 1988

[4] Beth, Heß, Wirl: *Kryptographie*, Teubner 1983.

[5] Blahut, R.E.: *Theory and Practice of Error Control Codes*, Addison-Wesley, 1983.

[6] Breimàn, L.: *Probability*, Addison–Wesley, 1968.

[7] Fumy, W., Rieß, H.P.: *Kryptographie*, Oldenburg, 1988.

[8] Gollmann, D.: *Kaskadenschaltungen taktgesteuerter Schieberegister als Pseudozufallsgeneratoren*, Dissertation Universität Linz, 1983.

[9] Hopcroft, F.E.; Ullman, F.D.: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.

[10] Massey, J.L.: *Shift Register Synthesis and BCH Decoding*, IEEE Trans. on Information Theory, **15** (1969), pp 122–127.

[11] Micali, S.; Schnorr, C.P.: *Efficient, Perfect Random Number Generators*, June 1988.

[12] Niederreiter, H.: *The Probabilistic Theory of Linear Complexity*, Proc. Eurocrypt 88, LNCS 330, p. 191 - 209.

[13] Rueppel, R.: *Analysis and Design of Stream Ciphers*, Springer, 1986.

[14] Schnorr,C.P.: *On the Construction of Random Number Generators and Random Function Generators*, Eurocrypt 1988, Davos.

[15] Schnorr,C.P., Stimin,H.: *Endliche Automaten und Zufallsfolgen*, Acta Informatica **1**, 345-359, (1972).

[16] Chaitin,G.J.: *Information, Randomness and Incompleteness*, World Scientific Publishing, Singapore 1987.