Analysis of Pseudo Random Sequences Generated by Cellular Automata

Willi Meier [1)] Othmar Staffelbach [2)]

[1)] HTL Brugg-Windisch
CH-5200 Windisch, Switzerland

[2)] GRETAG, Althardstrasse 70
CH-8105 Regensdorf, Switzerland

Abstract

The security of cellular automata for stream cipher applications is investigated. A cryptanalytic algorithm is developed for a known plaintext attack where the plaintext is assumed to be known up to the unicity distance. The algorithm is shown to be successful on small computers for key sizes up to N between 300 and 500 bits. For a cellular automaton to be secure against more powerful adversaries it is concluded that the key size N needs to be about 1000 bits.

The cryptanalytic algorithm takes advantage of an equivalent description of the cryptosystem in which the keys are not equiprobable. It is shown that key search can be reduced considerably if one is contented to succeed only with a certain success probability. This is established by an information theoretic analysis of arbitrary key sources with non-uniform probability distribution.

1 Introduction

In [7],[8] S. Wolfram introduced cellular automata for pseudo random sequence generation with possible application for stream ciphers. Another application is suggested in [1], which uses pseudo random sequences produced by cellular automata for constructing collision free hash functions.

A one-dimensional cellular automaton consists of a (possibly infinite) line of sites with values $s_i \in GF(2)$. These values are updated in parallel (synchronously) in discrete time steps according to a fixed rule of the form

$$s_i' = \Phi(s_{i-r}, s_{i-r+1}, \cdots, s_{i+r}). \tag{1}$$

Practical implementations of cellular automata must contain a finite number of sites N. These are arranged in a circular register, so as to have periodic boundary conditions.

D.W. Davies (Ed.): Advances in Cryptology - EUROCRYPT '91, LNCS 547, pp. 186-199, 1991.

Cellular automata have been investigated in studies of the origins of randomness and chaos in physical systems ([6], [7]). As pointed out in [2], cellular automata differ from cryptographic mechanisms such as feedback shift registers in that, even if they are invertible, it is not possible to calculate the predecessor of an arbitrary state by simply reversing the rule for finding the successor. It is suggested in [7] that the problem of deducing the initial configuration from partial output is equivalent to the general problem of solving Boolean equations, which is NP-complete.

The investigations in [7] concentrate on rules (1) with 3 arguments, i.e., $r = 1$. The two rules that seem best as random sequence generators are nonlinear, and are given by

$$s_i' = s_{i-1} \text{ XOR } (s_i \text{ OR } s_{i+1}) \tag{2}$$

or equivalently, $s_i' = s_{i-1} + s_i + s_{i+1} + s_i s_{i+1} \bmod 2$, and

$$s_i' = s_{i-1} \text{ XOR } (s_i \text{ OR } (\text{NOT } s_{i+1})) \tag{3}$$

or equivalently, $s_i' = 1 + s_{i-1} + s_{i+1} + s_i s_{i+1} \bmod 2$. It is indicated in [7] that, with respect to certain measures, rule (2) appears to be more favourable than rule (3). For stream cipher applications the values $s_i(t)$ attained by a particular site i through time serve as pseudo random bits in a key stream sequence. The initial state of the register is used as the seed, or the secret key.

In view of the period of the output sequences, practical implementations with key size $N = 127$ have been suggested in [7]. In this paper we address the question whether this key size is sufficient to withstand cryptanalysis. To this end a cryptanalytic algorithm is developed in Section 2 which appears to be successful on a PC for key sizes up to N between 300 and 500 bits (depending on the updating rule). Using large parallel computers with special hardware, it is conceivable that cryptanalysis is possible up to a key size of about 1000 bits.

In this cryptanalysis a known plaintext attack is assumed where the plaintext is known up to the unicity distance. It is shown that there is an equivalent cryptosystem, such that the problem of deducing the original key can be reduced to finding the key in the equivalent system. It turns out that the number of keys in the equivalent system is much smaller. Thus a search for the equivalent key is much more efficient than a search for the original key.

Moreover the equivalent system has another property which is favourable for cryptanalysis, namely that certain keys have much higher probability than others. It turns out that in this situation the cost of a cryptanalytic attack is considerably reduced if one is contented to succeed only with a certain success probability δ (e.g., $\delta = 0.5$). In this respect, the efficiency of our cryptanalytic algorithm has been determined by numerous experiments for different key sizes.

The results of Section 3 show that the security of the cellular automaton for rule (2) is in fact quite low for key sizes $N \leq 300$. As an example we mention that, for key size $N = 200$ and success probability $\delta = 0.5$, the average number of trials μ necessary to find the key was obtained to be about 23,000. Thus the effective key size can be estimated as $\log_2 \mu = 14.5$ bits. For larger N we can extrapolate from the experiments that, for the cellular automaton (with rule (2)) to be secure, one needs the key size of about 1000 bits, as mentioned above.

If our cryptanalytic algorithm is applied to cellular automata satisfying rule (3), it turns out that cryptanalysis is much easier than for rule (2). In fact, for the average number of trials μ, the effective key size $\log_2 \mu$ is roughly half of the corresponding value for rule (2), e.g., for $N = 400$ we have obtained $\log_2 \mu = 13.8$ bits. Therefore, our cryptanalysis confirms to prefer rule (2) over rule (3).

The equivalent cryptosystem used in our attack implements a key source with a strongly non-uniform, but not explicitly known probability distribution. In general, for a given cryptosystem, knowledge of additional information (e.g., part of the plaintext) may result in an equivalent description of the cryptosystem where the keys in the equivalent system are no longer equiprobable. Therefore, an analysis of an "exhaustive search" for key sources with non-uniform probability distribution is of independent interest. Since by assumption one does not know the most probable keys, which would be tried first in an optimum search, the best one can do is to generate the trial keys according to the original probability distribution of the key source until the correct key has been found.

Suppose that the cryptanalyst performs m different attacks on m keys chosen by the cryptographer, and that the required numbers of trials are listed in ascending order $T_1 \leq T_2 \leq \ldots \leq T_m$. Then the mean value

$$\mu(m, \delta) = \frac{1}{\lceil \delta m \rceil} \sum_{i=1}^{\lceil \delta m \rceil} T_i$$

is an empirical value for the average number of trials to a given success probability δ.

In Section 4 lower and upper bounds for the expected value of $\mu(m, \delta)$ are derived for any key source. The natural question arises how $\mu(m, \delta)$ is related to the entropy of the key source. For stationary ergodic key sources with finite memory (as considered in [4]), it is shown that the expected value of μ satisfies

$$\lim_{N \to \infty} \liminf_{m \to \infty} \frac{\log_2 E[\mu(m, \delta)]}{N} = \lim_{N \to \infty} \limsup_{m \to \infty} \frac{\log_2 E[\mu(m, \delta)]}{N} = H_S,$$

where H_S denotes the per bit entropy of the key source S (as introduced by Shannon [5]). Although the key source considered in our analysis of cellular automata does not completely fit into the model of an ergodic source, one suspects that $E[\mu(m, \delta)]$ is related to the entropy. Therefore the experimental results in Sections 2 and 3 (notably Table 1) may also be applied to estimate the entropy of sequences produced by cellular automata (see [7], [3]).

2 Cryptanalysis on Cellular Automata

By (2) the values of the sites determined by a site vector $S(t) = (s_{i-n}(t), \ldots, s_i(t), \ldots, s_{i+n}(t))$ form a triangle as shown in Figure 1. The sequence $\{s_i(t)\}$ in the middle column is called the *temporal* sequence (at site i). The security of a stream cipher based on cellular automata relies on the difficulty of determining the seed from the given temporal sequence. In an exhaustive search one would try all possible seeds until the correct temporal sequence is produced.

$$
\begin{array}{ccccccccccc}
s_{i-n}(t) & * & \dots & * & s_{i-1}(t) & s_i(t) & s_{i+1}(t) & * & \dots & * & s_{i+n}(t) \\
 & * & \dots & * & s_{i-1}(t+1) & s_i(t+1) & s_{i+1}(t+1) & * & \dots & * & \\
 & & & * & \vdots & \vdots & \vdots & * & & & \\
 & & & & * & * & * & & & & \\
 & & & & & s_i(t+n) & & & & &
\end{array}
$$

Figure 1: Triangle determined by $s_{i-n}(t), \dots, s_{i+n}(t)$.

In this section we develop a different approach which finds the seed using the partial linearity of (2) by writing it in the form

$$s_{i-1}(t) = s_i(t+1) \text{ XOR } (s_i(t) \text{ OR } s_{i+1}(t)). \tag{4}$$

Given the values of the cells in two adjacent columns, this allows the values of all cells in a triangle to the left of the temporal sequence to be reconstructed. This process will be called *completion backwards*. By completion backwards, $N-1$ digits of the temporal sequence $\{s_i(t)\}$ and $N-2$ digits of its right adjacent sequence $\{s_{i+1}(t)\}$ determine the seed. Similarly the seed may also be reconstructed from $N-2$ digits of the temporal sequence and $N-1$ digits of its left adjacent sequence $\{s_{i-1}(t)\}$.

Thus if $N-1$ digits of the temporal sequence are given, knowledge of the seed is equivalent to knowledge of one of its adjacent sequences. In view of this equivalence an adjacent sequence can be considered as a key which determines the remainder of the temporal sequence. In a known plaintext attack only (a portion of) the temporal sequence is known, but neither of its adjacent sequences. Our aim is to search for a correct (right or left) adjacent sequence and to determine the seed by completion backwards. In fact it turns out that this search can be done much faster than a direct exhaustive search over the original seed.

In constructing adjacent sequences, which are consistent with a given temporal sequence, there is an essential difference between left and right adjacent sequences. We explain this difference for a general cellular automaton of width $2n+1$ where the sites $s_{i-n}(t), \dots, s_{i+n}(t)$ are not restricted by periodic boundary conditions.

Suppose we are given a temporal sequence $s_i(t), \dots, s_i(t+n)$. Referring to Figure 1, the problem of finding the left adjacent sequence is equivalent to completing the triangle consistently to the left of the temporal sequence. By (2), knowledge of $s_{i-n}(t), \dots, s_{i-1}(t)$ together with the temporal sequence is sufficient to determine the triangle to the left. However the sites $s_{i-n}(t), \dots, s_{i-1}(t)$ cannot be chosen arbitrarily. For example, in the case $s_i(t) = 1$ we must have $s_{i-1}(t) = s_i(t+1) + 1$, and there are similar restrictions at any site in the process of extending to the left.

On the other hand, for triangles to the right any choice of the sites $s_{i+1}(t), \dots, s_{i+n}(t)$ has a completion consistent with the given temporal sequence. This follows from the fact that, by (4), for any right adjacent $s_{i+1}(t)$ there is a left adjacent $s_{i-1}(t)$ consistent with the next digit $s_i(t+1)$ of the temporal sequence. Furthermore, according to (2), any choice of $s_{i+2}(t), \dots, s_{i+n}(t)$ has a consistent extension to

$s_{i+1}(t+1), \ldots, s_{i+n-1}(t+1)$. Iteration of this construction leads to the desired completion of the triangle to the right. This process will be called *completion forwards* which, for any choice of the sites $s_{i+1}(t), \ldots, s_{i+n}(t)$ constructs a right adjacent sequence consistent with the given temporal sequence.

Now suppose that the cellular automaton satisfies periodic boundary conditions with period N. This means that the indices are calculated modulo N. For illustration consider Figure 1 where $n = N-1$ and $s_{i-1}(t) = s_{i+N-1}(t), \ldots, s_{i-N+1}(t) = s_{i+1}(t)$. Thus to determine the seed we can either compute $(s_i(t), \ldots, s_{i+N-1}(t))$ or $(s_{i-N+1}(t), \ldots, s_i(t))$. For $t' \geq t$, the right adjacent $s_{i+1}(t')$ may not be chosen arbitrarily to be consistent with the next digit $s_i(t'+1)$ of the temporal sequence, as the left adjacent $s_{i-1}(t')$ may be restricted by the boundary conditions. Nevertheless, by neglecting the boundary conditions, for any choice of the values $s_{i+1}(t), \ldots, s_{i+N-1}(t)$, completion forwards yields a consistent triangle to the right, and one obtains a possible candidate for the right adjacent sequence. In fact, if $s_{i+1}(t), \ldots, s_{i+N-1}(t)$ coincide with the original seed, the correct right adjacent sequence will be produced. Surprisingly, it turns out that there are many (incorrect) seeds leading to the correct right adjacent sequence which, by completion backwards, determines the correct seed. This phenomenon is the basis of our cryptanalytic algorithm which, for given digits $s_i(t), \ldots, s_i(t+N-1)$ of the temporal sequence, determines the seed.

Algorithm.

1. Generate a random seed $s_{i+1}(t), \ldots, s_{i+N-1}(t)$.

2. Complete forwards, i.e. determine the right adjacent sequence by executing

   FOR $k := 1$ TO $N-2$ DO FOR $j := 1$ TO $N-k-1$ DO
   $s_{i+j}(t+k) := s_{i+j-1}(t+k-1)$ XOR $(s_{i+j}(t+k-1)$ OR $s_{i+j+1}(t+k-1))$

3. Complete backwards, i.e. determine the seed $s_{i-N+1}(t), \ldots s_{i-1}(t)$ by executing

   FOR $j := 1$ TO $N-1$ DO FOR $k := N-1-j$ DOWNTO 0 DO
   $s_{i-j}(t+k) := s_{i-j+1}(t+k+1)$ XOR $(s_{i-j+1}(t+k)$ OR $s_{i-j+2}(t+k))$

4. Load the cellular automaton with the computed seed and produce the output sequence up to the length according to the unicity distance. Terminate if the produced sequence coincides with the given temporal sequence else go to step 1.

The algorithm is illustrated by an example with $N = 5$ as shown in Figure 2. Let the seed be $(s_i, \ldots, s_{i+4}) = (0,1,0,1,1)$. Then, by the boundary conditions, $(s_{i-4}, \ldots, s_{i-1}) = (1,0,1,1)$, and the resulting temporal sequence is obtained as $(0,0,1,0,0)$. According to step 1 of the algorithm the digits $s_{i+1}, \ldots, s_{i+4}$ are chosen randomly. If s_{i+1} happens to be 1, then by rule (2) the right adjacent sequence produced turns out to be independent of the choice of s_{i+2}, s_{i+3} and s_{i+4}. (Therefore these digits are marked by $\star$ in Figure 2.) Thus there is only one right adjacent sequence with $s_{i+1} = 1$. As one can see from Figure 2, this right adjacent sequence is in fact correct. Thus with probability 1/2 the correct seed is found in the first trial.

s_{i-4}	s_{i-3}	s_{i-2}	s_{i-1}	s_i	s_{i+1}	s_{i+2}	s_{i+3}	s_{i+4}
1	0	1	1	**0**	1	0	1	1
	0	1	0	**0**	1	0	1	
		1	1	**1**	1	0		
			0	**0**	0			
				0				

Generation of temporal sequence.

s_{i-4}	s_{i-3}	s_{i-2}	s_{i-1}	s_i	s_{i+1}	s_{i+2}	s_{i+3}	s_{i+4}
1	0	1	1	**0**	1	⋆	⋆	⋆
	0	1	0	**0**	1	⋆	⋆	
		1	1	**1**	1	⋆		
			0	**0**	0			
				0				

Determination of the seed by completion backwards.

Figure 2: Cryptanalysis—a small example.

The example shows that for a given temporal sequence there may be only few right adjacent sequences, and that some of these occur with high probability, e.g., for the temporal sequence (0,0,1,0,0) the sequence (1,1,1,0) is generated with probability 1/2 as right adjacent sequence. The phenomenon as observed in the example also arises in a general automaton with arbitrary N. This is based on the fact that some changes in right-hand initial sites have no effect on the given part of the temporal sequence or its right adjacent sequence. In fact it has been observed in [7] that the effect of a bit change propagates to the left with speed rougly 1/4. By the partial linearity of (2) the speed of propagation to the right is 1. This further justifies to search for right adjacent rather than for left adjacent sequences.

In general, a temporal sequence $s_i(t),\ldots,s_i(t+n-1)$ of length n together with the values of $n+1$ sites $s_{i+1}(t),\ldots,s_{i+n+1}(t)$ determines a right adjacent sequence $s_{i+1}(t),\ldots,s_{i+1}(t+n)$ of length $n+1$. It appears that the number of right adjacent sequences strongly depends on the given temporal sequence. In fact for some temporal sequences there remain only very few possibilities for the right adjacent sequence. Suppose for example that the temporal sequence consists of n consecutive digits 0, i.e., $s_i(t) = s_i(t+1) = \ldots = s_i(t+n-1) = 0$. Let k be the smallest non-negative integer with $s_{i+1}(t+k) = 1$. Then rule (2) implies that $s_{i+1}(t+k) = s_{i+1}(t+k+1) = \cdots = s_{i+1}(t+n) = 1$. Thus the right adjacent sequence $s_{i+1}(t),\ldots,s_{i+1}(t+n)$ can only consist of k digits 0 followed by $n-k+1$ digits 1. Hence in this case the number of possible right adjacent sequences grows only linearly in the length n of the temporal sequence. By slightly different arguments one can show that a similar result holds for temporal sequences consisting of n consecutive digits 1.

For arbitrary temporal sequences statistical experiments show that also in general

there are only few possibilities for the right adjacent sequence. In Table 1, for every N, $2 \leq N \leq 17$, samples of 100 temporal sequences of length $N-1$ have been generated. For any of these temporal sequences the number of all possible right adjacent sequences have been determined by an exhaustive search over all 2^N seeds. As a result, for each sample the mean value μ and standard deviation σ are listed in Table 1. Note for example that for $N = 17$, in the average, there are only about 42 right adjacent sequences compared to all $2^{17} = 131,072$ possible sequences of length 17.

N	μ	σ	N	μ	σ
2	3.00	0.00	3	4.00	0.00
4	5.00	0.00	5	6.29	0.46
6	7.73	0.81	7	9.59	1.21
8	11.47	1.67	9	13.90	2.38
10	15.78	2.85	11	18.36	3.67
12	21.64	4.45	13	24.95	5.57
14	28.18	7.03	15	32.86	7.72
16	37.49	9.71	17	41.86	10.63

Table 1: Statistics on the number of right adjacent sequences.

3 Experimental results

In this section we report on the results of extensive experiments concerning our cryptanalytic algorithm. In these experiments we simulate a known plaintext attack where the keys are chosen with uniform probability as seed of the cellular automaton of size N (i.e., the size of the keys is N). The problem of cryptanalysis consists in determining the seed (or the key) from the produced output sequence, which is assumed to be known up to the unicity distance.

It turns out that the efficiency of our cryptanalytic algorithm strongly depends on properties of the known output sequence, i.e., on the chosen key. In fact, the algorithm is much more efficient for those keys whose corresponding output sequences (or temporal sequences) turn out to have only few right adjacent sequences. As pointed out in Section 2, for a cellular automaton with known output sequence the search for the original key K_s (the seed) is equivalent to finding the correct right adjacent sequence. Therefore we consider the right adjacent sequence as an equivalent key, denoted by K_r. By completion forwards there is a mapping

$$F : \{K_s\} \longrightarrow \{K_r\}. \tag{5}$$

By assumption, the keys K_s are equiprobable, whereas the keys K_r turn out to have

a non-uniform probability distribution determined by the mapping F,

$$P(K_r) = \frac{1}{2^N} |\{K_s \mid F(K_s) = K_r\}| . \tag{6}$$

For the cellular automaton to be secure, the mapping F is supposed to be complex and involved. Thus it has to be assumed that the probability distribution (6) is not explicitly known, neither to the cryptographer nor to the cryptanalyst. In particular one does not know the most probable keys which in an optimum search would be tried first. Therefore the best one can do is to generate the keys K_r according to the probability distribution (6). This is done by choosing K_s randomly according to the uniform distribution and to generate the key K_r as the corresponding right adjacent sequence. Then the expected number of trials T for finding a particular key K_r can be computed as

$$E[T|K_r] = \frac{1}{P(K_r)} \tag{7}$$

which in fact is lowest for the most probable keys. If the cryptanalytic attack is required to be successful on every key, the expected number of trials is

$$E[T] = \sum_{K_r} E[T|K_r]\, P(K_r) = \sum_{K_r} \frac{1}{P(K_r)} P(K_r) = |\{K_r\}|. \tag{8}$$

In the average this is much smaller than the number of trials necessary for an exhaustive search over K_s, as $|\{K_r\}| \ll 2^N$ (e.g., see Table 1). Nevertheless for large N it may become infeasible to find the key in every case, as $|\{K_r\}|$ may be too large. However it turns out that certain keys K_r appear with high probability $P(K_r)$ (i.e., are accidentially generated by the cryptographer with high probability). By (7), the expected number of trials for finding such keys may be much smaller than the average expected number as obtained in (8). Thus we are in the situation that the algorithm may not be able to find the key in every case, but may succeed with a certain success probability δ, say $\delta = 0.5$. In order to obtain an estimate of the complexity of the attack to a given success probability δ, we set up our experiments as follows.

1. Simulate the cryptographer by choosing a random key K_s as seed of the cellular automaton.

2. Simulate a known plaintext situation by generating the output sequence (or the temporal sequence) up to the unicity distance.

3. Apply the cryptanalytic algorithm of Section 2 in order to determine the key K_r (or K_s respectively) and count the number T of trials that have been necessary.

These steps are repeated a certain number of times m, and the required numbers of trials are listed in ascending order, $T_1 \leq T_2 \leq \ldots \leq T_m$. To obtain an empirical value for the average number of trials for success probability δ we compute the mean value

$$\mu = \frac{1}{\lceil \delta m \rceil} \sum_{i=1}^{\lceil \delta m \rceil} T_i . \tag{9}$$

In Table 2 the results of $m = 100$ experiments are listed for each key size $N = 10, 20, \ldots, 300$, as indicated. The success probability has been chosen to be $\delta = 0.5$. The value $\log_2 \mu$ as given in the third and sixth column indicates the *effective key size* with respect to this success probability δ.

Key size	μ	$\log_2 \mu$	*Key size*	μ	$\log_2 \mu$
10	2.9	1.5	20	5.5	2.4
30	10.2	3.4	40	23.0	4.5
50	23.9	4.6	60	73.7	6.2
70	102.6	6.7	80	178.9	7.5
90	270.5	8.1	100	334.1	8.4
110	687.5	9.4	120	1,696.3	10.7
130	1,626.0	10.7	140	1,956.8	10.9
150	2,740.3	11.4	160	4,131.3	12.0
170	9,267.8	13.2	180	10,206.1	13.3
190	14,876.0	13.9	200	23,367.5	14.5
250	97,583.1	16.6	300	272,195.6	18.1

Table 2: Complexity of the attack with success probability $\delta = 0.5$ for rule (2). (For technical reasons the entry for $N = 250$ corresponds to $\delta = 0.44$ and for $N = 300$ to $\delta = 0.27$.)

The results of our experiments allow to estimate the security of the cellular automaton for stream cipher applications. In this respect Table 2 shows that the security is quite low for key sizes $N \leq 300$. For larger key size we have to extrapolate the average complexity of the attack from the values μ obtained for $N \leq 300$. For this purpose we observe that the number of right adjacent sequences grows at most exponentially in the length n of the sequence, i.e.,

$$f_r(m+n) \leq f_r(m) f_r(n) \tag{10}$$

where $f_r(n)$ denotes the number of all possible length n right adjacent sequences corresponding to a given temporal sequence. Then (10) implies that the complexity of the attack also grows at most exponentially in the length N of the key.

The derivation of (10) is explained in Figure 3. Let $\mathcal{A}$ be the set of all possible right adjacent sequences $(s_{i+1}(t), \ldots, s_{i+1}(t+m-1))$ of length m, let $\mathcal{B}$ be the set of all possible right adjacent sequences $(s_{i+1}(t+m), \ldots, s_{i+1}(t+m+n-1))$ of length n, and let $\mathcal{C}$ be the set of all possible right adjacent sequences $(s_{i+1}(t), \ldots, s_{i+1}(t+m+n-1))$ of length $m+n$. Then for any right adjacent sequence $\{s_{i+1}\}$ in $\mathcal{C}$, the sequence $\{s'_{i+1}\}$ formed by the first m digits of $\{s_{i+1}\}$ lies in $\mathcal{A}$, and the sequence $\{s''_{i+1}\}$ formed by the last n digits of $\{s_{i+1}\}$ lies in $\mathcal{B}$. Therefore we have $|\mathcal{C}| \leq |\mathcal{A}| \cdot |\mathcal{B}|$, which proves (10). Inequality usually holds in (10), since the site values $s_{i+1}(t+m), \ldots, s_{i+m}(t+m)$ can be chosen arbitrarily in generating the set $\mathcal{B}$ whereas, in generating the set $\mathcal{C}$, they are restricted to conditions caused by the cellular automaton rule (2). Thus there

may exist pairs $(\{s'_{i+1}\}, \{s''_{i+1}\})$ in $\mathcal{A} \times \mathcal{B}$ whose concatenation is not a right adjacent sequence of length $m + n$.

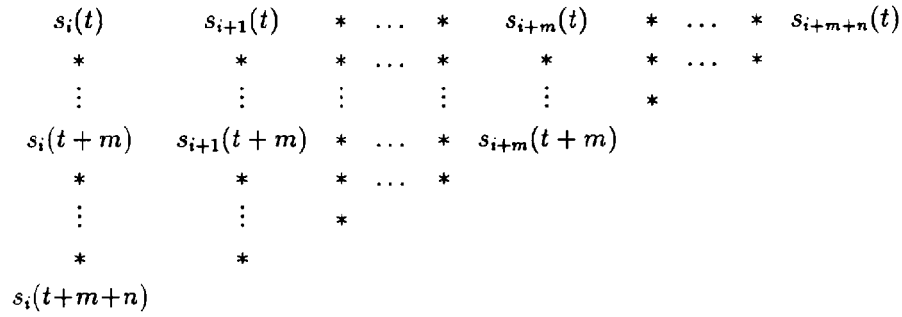

Figure 3: Generation of right adjacent sequences.

A subexponential growth of the average complexity of the attack is also suggested by Table 2, e.g., for $N = 200$ we have $\log_2 \mu = 14.5$ which is less than the double of $\log_2 \mu = 8.4$ for $N = 100$. Therefore, from the results for $N = 250$ or $N = 300$ we extrapolate values of N for which the attack requires approximately 2^{50} trials. These values lie in the range between $N = 750$ and $N = 900$. Therefore, using large parallel computers with special hardware, it is conceivable that cryptanalysis is possible up to a key size of about 1000 bits.

Our cryptanalytic algorithm also applies to cellular automata satisfying other rules, e.g., rule (3). It turns out that the algorithm for rule (3) is much more efficient than for rule (2). In Table 3 the results of $m = 100$ experiments are listed for each key size $N = 50, 100, \ldots, 400$ and for success probability $\delta = 0.5$. The results show that the effective key size $\log_2 \mu$ for rule (3) is roughly half of the effective key size for rule (2).

Key size	μ	$\log_2 \mu$	*Key size*	μ	$\log_2 \mu$
50	22.8	4.5	100	82.8	6.4
150	206.2	7.7	200	423.3	8.7
250	1,396.3	10.4	300	2,559.3	11.3
350	5,457.6	12.4	400	14,400.5	13.8

Table 3: Complexity of the attack with success probability $\delta = 0.5$ for rule (3).

4 Key Search for Key Sources with Non-uniform Probability Distribution

In Sections 2 and 3 the search for the seed K_s of a cellular automaton to a given output sequence was reduced to finding the correct right adjacent sequence. In this

equivalent description the right adjacent sequence was considered as a key K_r which turned out to have a non-uniform probability distribution. In general, for a given cryptosystem, knowledge of additional information (e.g., part of the plaintext) may result in an equivalent description of the cryptosystem where the keys in the equivalent system are no longer equiprobable. Therefore, an analysis of an "exhaustive search" for key sources with non-uniform probability distribution is of independent interest.

Suppose we are given a cryptographic system with key source S. Denote by $\mathcal{K}$ the set of all keys and let $\{P(k), k \in \mathcal{K}\}$ be the probability distribution of the keys. It is assumed that the individual probabilities of the keys are ***unknown*** (even to the cryptographer). If the keys are available in an ordered list $\mathcal{K} = \{k_1, k_2, \ldots, k_M\}$ the cryptanalyst could try all keys in this order until the correct key has been found. If this order is independent of the probability distribution, he cannot take any advantage of the fact that some keys are more probable than others. An *optimum search* would be to try the keys in the order of decreasing probabilities. Since by assumption this order is unknown, the best one can do is to generate the keys by the original key source S according to the probability distribution P. Then the expected number of trials T for finding a particular key k can be computed as

$$E[T|k] = \frac{1}{P(k)} \tag{11}$$

which implies that the most probable keys are easiest to find. However, over all keys the cryptographer may choose, the expected number of trials is

$$E[T] = \sum_{k \in \mathcal{K}} E[T|k]\, P(k) = \sum_{k \in \mathcal{K}} \frac{1}{P(k)} P(k) = |\mathcal{K}|. \tag{12}$$

This means that in the average, or in long terms, cryptanalysis cannot take advantage of the fact that the probability distribution of the keys is not uniform. However the situation turns out to be different if the cryptanalyst is contented with finding the key only with a certain success probability δ, e.g. $\delta = 0.5$.

As in our experiments in Section 3 suppose that the cryptanalyst performs m different attacks on m keys chosen by the cryptographer according to the probability distribution P of the key source S. The required number of trials are listed in ascending order $T_1 \leq T_2 \leq \ldots \leq T_m$. Then the mean value

$$\mu(m, \delta) = \frac{1}{\lceil \delta m \rceil} \sum_{i=1}^{\lceil \delta m \rceil} T_i \tag{13}$$

is an empirical value for the average number of trials to a given success probability δ. Our aim is to establish a relationship between the expected value of $\mu(m, \delta)$ in (13) and the entropy of the key source S.

Let $A \subset \mathcal{K}$ be a subset of keys. Then a key k produced by the source S accidentally lies in A with probability

$$P(A) = \sum_{k \in A} P(k). \tag{14}$$

The expected number of trials T conditioned on the event that the key, chosen by the cryptographer, lies in A is given by

$$E[T \mid k \in A] = \sum_{k \in A} E[T|k] \frac{P(k)}{P(A)} = \sum_{k \in A} \frac{1}{P(k)} \cdot \frac{P(k)}{P(A)} = \frac{|A|}{P(A)}. \tag{15}$$

Lemma 1 *For any δ, $0 < \delta < 1$, and any subset $A \subset \mathcal{K}$ with $P(A) > \delta$,*

$$\limsup_{m \to \infty} E[\mu(m, \delta)] \leq \frac{|A|}{P(A)}. \tag{16}$$

Proof. Let B denote the event that at least $\lceil \delta m \rceil$ keys, out of the m keys chosen by the cryptographer, lie in A. Then the expected value of $\mu(m, \delta)$, conditioned on the event B, satisfies the inequality

$$E[\mu(m, \delta) \mid B] \leq E[T \mid k \in A] = \frac{|A|}{P(A)}. \tag{17}$$

On the other hand the expected value of $\mu(m, \delta)$, conditioned on the complementary event of B (denoted by B^c), satisfies the inequality

$$E[\mu(m, \delta) \mid B^c] \leq \max\left(\frac{|A|}{P(A)}, \frac{M - |A|}{1 - P(A)}\right). \tag{18}$$

Since $P(A) > \delta$, by the law of large numbers, for any $\varepsilon > 0$ there is an integer m_0 such that for all $m \geq m_0$, the event B has probability $\lambda = P(B) > 1 - \varepsilon$. Thus

$$\begin{aligned} E[\mu(m, \delta)] &= E[\mu(m, \delta) \mid B]\, P(B) + E[\mu(m, \delta) \mid B^c]\, P(B^c) \\ &\quad \lambda\, E[\mu(m, \delta) \mid B] + (1 - \lambda)\, E[\mu(m, \delta) \mid B^c] \end{aligned} \tag{19}$$

The first term in (19) is bounded by (17), whereas the second term is of the form $(1 - \lambda)K$ where K is bounded by (18). Since the bound (18) is independent of m, by choosing ε sufficiently small (or equivalently, by choosing m sufficiently large), the second term in (19) can be made arbitrarily small. □

Lemma 2 *For any δ, $0 < \delta < 1$, and any subset $A \subset \mathcal{K}$ with $\varepsilon = 1 - P(A) < \delta$,*

$$\liminf_{m \to \infty} E[\mu(m, \delta)] \geq \frac{\alpha}{p_a}, \tag{20}$$

where p_a denotes the probability of the most probable key in A, and where $\alpha = (\delta - \varepsilon)^2/(4\delta)$.

Proof. Let $k \in \mathcal{K}$ and $p = P(k)$. For $0 < \beta < 1$, let $P(T < (\beta/p) \mid k)$ denote the probability of $T < \beta/p$, conditioned on the event that the cryptographer has chosen key k. Then by Bernoulli's inequality we have

$$P(T < (\beta/p) \mid k) = 1 - (1 - p)^{\lfloor \beta/p \rfloor} < 1 - (1 - \lfloor \beta/p \rfloor p) = \lfloor \beta/p \rfloor p \leq \beta.$$

Thus for any key in A,

$$P(T < (\beta/p_a) \mid k) \leq P(T < (\beta/p) \mid k) < \beta, \tag{21}$$

which can be applied to estimate

$$P(T < \beta/p_a) = \sum_{k \in A} P(T < (\beta/p_a) \mid k)\, P(k) + \sum_{k \in \mathcal{K} - A} P(T < (\beta/p_a) \mid k)\, P(k). \tag{22}$$

According to (21), the first term in (22) is bounded by $\beta(1-\varepsilon) < \beta$, whereas the second term is trivially bounded by $P(\mathcal{K} - A) = \varepsilon$. Hence

$$P(T < \beta/p_a) < \beta + \varepsilon. \tag{23}$$

Let $\delta_1 = P(T < \beta/p_a)$ and $\delta_2 = \beta + \varepsilon$. Since $\varepsilon < \delta$, it is possible to choose β such that $\delta_2 < \delta$. We will make the choice $\delta_2 = (\delta + \varepsilon)/2$. Let B denote the event that at most $\delta_2 m$ numbers T among $T_1, \ldots, T_m$ are below β/p_a. Then the expected value of $\mu(m, \delta)$ conditioned on B satisfies

$$E[\mu(m,\delta) \mid B] \geq \frac{(\delta - \delta_2)}{\delta} \cdot \frac{\beta}{p_a} = \frac{(\delta - \delta_2)(\delta_2 - \varepsilon)}{\delta p_a} = \frac{(\delta - \varepsilon)^2}{4\delta p_a}. \tag{24}$$

Since by (23), $\delta_1 = P(T < \beta/p_a) < \delta_2$, it follows, by the law of large numbers, that for any $\eta > 0$ there is an integer m_0 (depending on δ_1, δ_2 and η, only) such that for all $m > m_0$ the event B has probabilty $P(B) > 1 - \eta$. Hence

$$E[\mu(m,\delta] \geq E[\mu(m,\delta) \mid B]\, P(B) \geq \frac{(\delta - \varepsilon)^2}{4\delta p_a}(1 - \eta). \tag{25}$$

By choosing m sufficiently large, η can be made arbitrarily small. This implies that (20) is satisfied with $\alpha = (\delta - \varepsilon)^2/(4\delta)$. □

Now we consider the case where the keys produced by the source S are bit strings of length N, i.e., $M = 2^N$. We assume that S is a stationary ergodic source with a finite memory according to the model considered in [4]. In this model the state of the source is given by the k preceeding output bits $(u_{n-1}, \ldots, u_{n-k})$. Therefore any state has only two possible successors, namely $(u_n, \ldots, u_{n-k+1})$ with $u_n = 0$, or with $u_n = 1$. It follows that there is a one-to-one correspondence between the state sequence of the source and its output sequence. Therefore the entropy per symbol (or state) H_S, as introduced by Shannon in [5], can be considered as the entropy per ouput bit of the key source (see [4]). As a consequence, Shannon's Theorem 3 in [5] can be applied, which is stated here as follows.

Theorem 3 (Shannon) *For any $\varepsilon > 0$ and $\eta > 0$ there exists an integer N_0 such that for all $N \geq N_0$, there is a set $A \subset \mathcal{K}$ with probability $P(A) > 1 - \varepsilon$ such that the probability of any element in A lies within the bounds*

$$2^{-(H_S + \eta)N} \leq p \leq 2^{-(H_S - \eta)N}. \tag{26}$$

As a consequence the cardinality of A lies within the bounds

$$2^{(H_S - \eta)N} \leq |A| \leq 2^{(H_S + \eta)N}. \tag{27}$$

In Theorem 3 we may choose $\varepsilon < \min(\delta, 1-\delta)$ in order to apply Lemma 1 and Lemma 2, and thus we obtain

$$\alpha\, 2^{(H_S-\eta)N} \leq \liminf_{m\to\infty} E[\mu(m,\delta)] \leq \limsup_{m\to\infty} E[\mu(m,\delta)] \leq \frac{1}{1-\varepsilon} 2^{(H_S+\eta)N},$$

or by taking logarithms and dividing by N,

$$\begin{aligned} \frac{\log_2 \alpha}{N} + (H_S - \eta) &\leq \liminf_{m\to\infty} \frac{\log_2 E[\mu(m,\delta)]}{N} \\ &\leq \limsup_{m\to\infty} \frac{\log_2 E[\mu(m,\delta)]}{N} \leq (H_S+\eta) - \frac{\log_2(1-\varepsilon)}{N}. \end{aligned}$$

By choosing N sufficiently large, ε and η can be made arbitrarily small. At the same time α approaches the value $\delta/4$. Thus we have proved the following theorem. □

Theorem 4 *An ergodic stationary key source S with a finite memory satisfies*

$$\lim_{N\to\infty} \liminf_{m\to\infty} \frac{\log_2 E[\mu(m,\delta)]}{N} = \lim_{N\to\infty} \limsup_{m\to\infty} \frac{\log_2 E[\mu(m,\delta)]}{N} = H_S. \qquad (28)$$

References

[1] I. Damgård, *A Design Principle for Hash Functions*, Advances in Cryptology—Crypto'89, Proceedings, pp. 416–427, Springer-Verlag, 1990.

[2] W. Diffie, *The First Ten Years of Public-Key Cryptography*, Proceedings of the IEEE, pp. 560–577, 1988.

[3] P. Grassberger, *Toward a Quantitative Theory of Self-Generated Complexity*, International Journal of Theoretical Physics, Vol. 25, pp. 907–938, 1986.

[4] U. Maurer, *A Universal Statistical Test for Random Bit Generators*, Proceedings of Crypto'90, Springer-Verlag, to appear.

[5] C.E. Shannon, *A Mathematical Theory of Communication*, Bell Syst. Tech. Journal, Vol. 27, pp. 379–423, 623–656, 1948.

[6] S. Wolfram, *Origins of Randomness in Physical Systems*, Physical Review Letters, Vol. 55, pp. 449–452, 1985.

[7] S. Wolfram, *Random Sequence Generation by Cellular Automata*, Advances in Applied Mathematics 7, pp. 123–169, 1986.

[8] S. Wolfram, *Cryptography with Cellular Automata*, Advances in Cryptology—Crypto'85, Proceedings, pp. 429–432, Springer-Verlag, 1986.