Linear complexity of transformed sequences

H. Fell

Rapports de Recherche

N° 1168

Programme 1
Programmation, Calcul Symbolique
et Intelligence Artificielle

LINEAR COMPLEXITY OF TRANSFORMED SEQUENCES

Harriet J. FELL

Février 1990

RR.1168

Linear Complexity of Transformed Sequences

Complexité linéaire de suites transformées

Harriet J. Fell[1]

Juin 1989

[1] This work was completed with support from the College of Computer Science at Northeastern University, Boston, Massachusetts, USA and from The Institut National de Recherche en Informatique et en Automatique, Rocquencourt, France.

Abstract

This paper deals with the effect of bit change errors on the linear complexity of finite sequences. Though a change in a single bit can cause a large change in linear complexity, it is shown that on the average the change will be small even when many bits, e.g. 10%, are changed. General bijections and k-fold mappings on the set of sequences of length n are studied and tight bounds are found on the average difference in linear complexity between a sequence and its image. It is also shown that any mapping, on sequences of length n that take *most* sequences to images of *low* linear complexity must take *many* sequences to images that are *far away* from them in Hamming distance.

Cet article traite des effets des erreurs de changement de bit sur la complexité linéaire des suites finies. Malgré le fait qu'un changement d'un seul bit peut causer un grand changement dans la complexité linéaire, il est démontré qu'en moyenne le changement sera petit, même quand plusieurs bits, par exemple 10%, sont changés. Les bijections générales et les fonctions qui n'envoient pas plus de k éléments sur le même point image sont étudiées et des bornes précises sont trouvées pour la moyenne de la valeur absolue de la différences entre la complexité linéaire d'une suite et son image. Il est démontré aussi qu'une fonction quelconque sur les suites de longueur n telle que *la plupart* de ses valeurss ont une complexité linéaire *assez petite* doit envoyer *beaucoup* de suites sur des images qui sont *éloignées* d'elles en distance de Hamming.

1 Introduction

The linear complexity of a finite sequence can change drastically if a single bit is changed or deleted. For example, the sequence $0, 0, \ldots, 0, 1$ of length n has maximal linear complexity, n, while deleting the last bit or changing it to a 0 results in a sequence of linear complexity 0. A shift register that generates a given sequence can be found in time linear in the linear complexity of the sequence [1], so sequences of low linear complexity are not cryptographically secure. At the *Workshop on Stream Ciphers*, at Karlsruhe, Germany, January 9-12, 1989, W. Diffie suggested that for many finite sequences, it might be possible to find small linear feedback shift registers (LFSRs) that generate nearby sequences. That is, if we can tolerate some errors, we might find it easy to generate a sequence close enough to a given sequence for cryptanalytic purposes.

In this paper, we first look directly at the effect of bit change errors on the linear complexity of sequences. We then look, more generally at functions that take sequences of length n into sequences of length n and study the average difference in the linear complexity of a sequence and its image. Finally, we apply these results to analyze the average change in linear complexity when errors, caused by changed bits, are introduced into a sequence.

2 Notation

We restrict our attention to sequences over the field with two elements, $\mathbf{Z}_2$. Let $\mathrm{S_n} := \{0,1\}^n$ be the set of all strings of zeros and ones of length n. If $s \epsilon \mathrm{S_n}$ then $s = s_1, s_2, \ldots, s_n$ where each s_i is either 0 or 1.

An infinite sequence $s = \{s_i\}_{i=1\ldots\infty}$ is said to be generated by a linear feedback shift register of length k if there exist constants $a_0, \ldots, a_{k-1}$ such that

$$s_{i+k} = a_{k-1}s_{i+k-1} + \ldots + a_1 s_{i+1} + a_0 s_i \qquad \textit{for } i \geq 1$$

For $s \epsilon \mathrm{S_n}$ we define the *linear complexity*, $\Lambda(s)$, to be the length of the smallest linear feedback shift register that generates a sequence whose first n terms are $s_1, s_2, \ldots, s_n$. If $k \leq n$, we will use $\Lambda_k(s)$ to denote the linear complexity of the sequence, $s_1, s_2, \ldots, s_k$.

If $s, t \epsilon \mathrm{S_n}$, we define $\Delta(s,t) \equiv |\Lambda(s) - \Lambda(t)|$ and for $1 \leq k \leq n$, $\Delta_k(s,t) \equiv |\Lambda_k(s) - \Lambda_k(t)|$.

3 Early Differences

Intuition says that changes near the start of a sequence should not have too much effect on the linear complexity of the sequence. This is true, in a sense, even if bits are lost or added, and the result is summarized by the following theorem.

Theorem 3.1 *Let $s \epsilon \mathrm{S_n}$ and $t \epsilon \mathrm{S_m}$ be such that $s_{n-i} = t_{m-i}$ for $i = 1 \ldots k < n, m$. Then*

$$| \Lambda_n(s) - \Lambda_m(t) | \leq \max(n-k, m-k).$$

Proof:

Intuitively, an LFSR, of length ℓ that generates s can have its register enlarged to house the first $m-k$ discrepant bits of t. After these bits are pushed out, the register will continue to work as it did before, tapping only the high ℓ bits of the register to produce the sequence, t. Formally, assume that $\Lambda(s) = \ell$ and $a_0, \ldots, a_{\ell-1} \epsilon Z_2$ are such that

$$s_{i+\ell} = a_{\ell-1}s_{i+\ell-1} + \ldots + a_1 s_{i+1} + a_0 s_i \qquad for\ i \geq 1,$$

let $b_{m-k+i} = a_i$ for $i = 0, \ldots, \ell$ and $b_j = 0$ for $0 \leq j < m-k$. Then

$$t_{i+m-k+\ell} = b_{m-k+\ell-1}t_{i+m-k+\ell-1} + \ldots + b_1 t_{i+1} + b_0 t_i, \qquad for\ i \geq 0$$

so $\Lambda_m(t) \leq \Lambda_n(s) + m - k$.

□

If two LFSRs have generated the same bits for a while, although not initially, we might expect them to continue producing identical bits. The following lemma formalizes this statment and will be of use in the next section.

Lemma 3.1 *Let $s \epsilon \mathrm{S_n}$ and $t \epsilon \mathrm{S_{n+r}}$ be such that $s_i = t_{r+i}$ for $i = j, \ldots, n$. If $\Lambda_{n-1}(s) + \Lambda_{n+r-1}(t) \leq n - j$ then either*

$$\Lambda_n(s) = \Lambda_{n-1}(s) \quad \text{and} \quad \Lambda_{n+r}(t) = \Lambda_{n+r-1}(t)$$

or

$$\Lambda_n(s) = n - \Lambda_{n-1}(s) \quad \textbf{and} \quad \Lambda_{n+r}(t) = n + r - \Lambda_{n+r-1}(t).$$

Proof:

Suppose $\Lambda_{n-1}(s) = \ell$ and $\Lambda_{n+r-1}(t) = m$. Then there exist $a_0, \ldots, a_{\ell-1}$ and $b_0, \ldots b_{m-1}$ such that

$$s_{i+\ell} = \sum_{j=0}^{\ell-1} a_j s_{i+\ell-j} \qquad n-1-\ell \geq i \geq 1$$

$$t_{i+m} = \sum_{j=0}^{m-1} b_j t_{i+m-j} \qquad n+r-1-m \geq i \geq 1.$$

We examine the next bits produced by these linear recurrences (or LFSRs) and see that they must be the same. Let

$$\tilde{s}_n := \sum_{j=1}^{\ell-1} a_j s_{n-j} \text{ and } \tilde{t}_{n+r} := \sum_{j=1}^{m-1} b_j t_{n+r-j}.$$

Now, using the fact that s and t have had $n-j$ identical bits, we have

$$\tilde{s}_n = \sum_{j=1}^{\ell-1} a_j s_{n-j} = \sum_{j=1}^{\ell-1} a_j t_{n-j}$$

$$= \sum_{j=1}^{\ell-1} a_j \sum_{k=1}^{m-1} b_k t_{n-j-k} = \sum_{k=1}^{m-1} b_k \sum_{j=1}^{\ell-1} a_j t_{n-j-k}$$

$$= \sum_{k=1}^{m-1} b_k t_{n-k} = \tilde{t}_n.$$

Hence, by the Berlekamp-Massey Synthesis Algorithm, [1], addition of one more identical bit either leaves the linear complexity of both sequences the same or changes them both as indicated in the statement of the lemma.

□

4 Changing Bits

The drastic increase in linear complexity caused by changing the last bit of a sequence of n zeros brings up the question of the general effect of a single bit change on the linear complexity of a finite sequence.

Fix an integer k , $1 \leq k \leq n$ and for each $s \epsilon \mathrm{S_n}$ be such that $s_k = 0$, let $t \epsilon \mathrm{S_n}$ be defined by $t_k = 1$ and $t_i = s_i$, for $i = 1 \ldots n$, $i \neq k$. Define

$$\overset{k}{\Delta}(n) := \frac{1}{2^{n-1}} \sum_{s \epsilon \mathrm{S_n}, s_k = 0} \mid \Lambda(s) - \Lambda(t) \mid .$$

This is the average change in linear complexity caused by a change in the k^{th} bit.

Although a change in the n^{th} bit can cause a severe change in linear complexity, on the average, the effect is surprisingly small.

Theorem 4.1 *The average change in the linear complexity of a n-bit string caused by a change in the last bit is given by*

$$\overset{n}{\Delta}(n) = \begin{cases} \frac{8}{9} + \frac{3n-4}{9 \cdot 2^{n-1}} \longrightarrow \frac{8}{9} & n \ even \\ \\ \frac{10}{9} + \frac{3n-4}{9 \cdot 2^{n-1}} \longrightarrow \frac{10}{9} & n \ odd \end{cases} .$$

Proof:

The distribution of $\Lambda(s)$ [3] is given by

$$card\{s : \Lambda(s) = k\} = \begin{cases} 1 & k = 0 \\ 2 \cdot 4^{k-1} & k \leq n/2 \\ 4^{n-k} & k > n/2 \end{cases} . \qquad (1)$$

Suppose $s, t \epsilon \mathrm{S_n}$ are such that $s_i = t_i, i = 1, \ldots, n-1$ while $s_n = 0$ and $t_n = 1$. Then $\Delta(s,t) = \mid \Lambda(s) - \Lambda(t) \mid$ is given by

$$\Delta(s,t) \ = \begin{cases} 0 & \Lambda_{n-1}(s) = \Lambda_{n-1}(t) \geq \frac{n}{2} \\ n-k & \Lambda_{n-1}(s) = \Lambda_{n-1}(t) < \frac{n}{2} \end{cases} .$$

The average difference in linear complexity when only the n^{th} bit is changed is given by

$$\overset{n}{\Delta}(n) = \frac{1}{2^{n-1}}\left[n\cdot 1 + \sum_{k=1}^{M}(n-2k)(2\cdot 4^{k-1})\right]$$

where $M = \frac{n}{2} - 1$ when n is even and $M = \frac{n-1}{2}$ when n is odd. Observing (once and for all) that

$$\sum_{k=1}^{M} 4^{k-1}k = \frac{(3M-1)4^M + 1}{9} \tag{2}$$

we have,

$$\overset{\pi}{\Delta}(n) = \frac{1}{2^{n-1}}\left[n + 2n\left(\frac{4^M - 1}{3}\right) - 4\left(\frac{(3M-1)4^M + 1}{9}\right)\right].$$

Finally, substituting the appropriate values for M gives the desired results.

□

It would be interesting to know the average change in linear complexity, $\overset{k}{\Delta}(n)$ for each k between 0 and n. Though we have not derived closed formulas for each of these averages, empirical data suggests the following conjectures [see tables].

Conjecture 1 *A change in the first bit of an n-bit sequence causes an average change in linear complexity given by*

$$a:\ \overset{1}{\Delta}(2n) = \overset{1}{\Delta}(2n-1) \qquad n \geq 1$$

$$b:\ \overset{1}{\Delta}(2n+1) = \overset{1}{\Delta}(2n) - \tfrac{1}{2^{2n}} \quad n \geq 1$$

$$c:\ \overset{1}{\Delta}(2n) = \tfrac{2}{3} + \tfrac{1}{3\cdot 4^{n-1}} \longrightarrow \tfrac{2}{3}$$

Partial Proof:

a : Suppose that $s, t\epsilon S_n$ differ only in the first place. Then by Theorem 3.1, $\Delta_{2n-1}(s,t) \leq 1$. If $\Lambda_{2n-1}(s)$ and $\Lambda_{2n-1}(t)$ are both greater than or equal to $\frac{2n}{2} = n$ then $\Lambda_{2n}(s) = \Lambda_{2n-1}(s)$ and $\Lambda_{2n}(t) = \Lambda_{2n-1}(t)$ and so $\Delta_{2n}s, t = \Delta_{2n-1}(s,t)$. If, on the other hand, $\Lambda_{2n-1}(s)$ and $\Lambda_{2n-1}(t)$ are both less than n, then $\Lambda_{2n-1}(s) + \Lambda_{2n-1}(t) \leq 2n-2$ and by Lemma 3.1, $\Delta_{2n}(s,t) = \Delta_{2n-1}(s,t)$. There is only one other possibility, that is, $\Lambda_{2n-1}(s) = n-1$ and $\Lambda_{2n-1}(t) = n$ or vice versa. But then we must have, $\Lambda_{2n}(t) = \Lambda_{2n-1}(t) = n$ and

$$\Lambda_{2n}(s) = \Lambda_{2n-1}(s) = n-1 \text{ or } \Lambda_{2n}(s) = 2n - \Lambda_{2n-1}(s) = n+1$$

which, in either case, implies $\Delta_{2n}(s,t) = \Delta_{2n-1}(s,t)$.

c : From (a) and (b) we have

$$\overset{1}{\Delta}(2n) = \overset{1}{\Delta}(2n-1) = \overset{1}{\Delta}(2n-2) - \frac{1}{2^{2n-2}}.$$

By induction, we see

$$\overset{1}{\Delta}(2n) = \overset{1}{\Delta}(2) - \sum_{j=1}^{n-1} 4^{-j} = \frac{2}{3} + \frac{1}{3 \cdot 4^{n-1}}.$$

$\triangle$

Conjecture 2 *Let C be a positive constant. Then*

$$\lim_{n \to \infty} \Delta_{\frac{n}{2} \pm C}(n) = 1.$$

The limit is approached from above.

Though these remain conjectures, an upper bound (4/3 when n is even and 5/3 when n is odd) on the values of $\overset{k}{\Delta}(n)$, $k = 1 \ldots n$ is obtained in the following section.

5 Bijections

Changing the k^{th} bit (or m bits in fixed positions) induces a bijection on S_n. We obtain a fairly low upper bound on the change in linear complexity caused by such bit flips by looking at the change caused by an arbitrary bijection.

position of difference	average difference	in decimal
1	683/1024	0.6669921
2	941/1024	0.9189453
3	1027/1024	1.0029297
4	1067/1024	1.0419922
5	1107/1024	1.0810547
6	1111/1024	1.0849609
7	1067/1024	1.0419922
8	1037/1024	1.0126953
9	971/1024	0.9482421
10	885/1024	0.8642578
11	1141/1024	1.1142578

overall average 11037/11264 0.9798473

Table 1: Average Difference in Linear Complexity for strings that differ in only one bit: String Length = 11

position of difference	average difference	in decimal
1	10923/16384	0.666687
2	15019/16384	0.916687
3	8193/8192	1.0001221
4	8405/8192	1.026001
5	8475/8192	1.0345459
6	16955/16384	1.0348511
7	16993/16384	1.0371704
8	17115/16384	1.0446167
9	17097/16384	1.0435181
10	17019/16384	1.0387573
11	2115/2048	1.0327148
12	16703/16384	1.0194702
13	8137/8192	0.9932861
14	15249/16384	0.9307251
15	7283/8192	0.889038
16	7283/8192	0.889038

overall average 255545/262144 0.9748268

Table 2: Average Difference in Linear Complexity for strings that differ in only one bit: String Length = 16

Theorem 5.1 *Let $\varphi : \mathrm{S_n} \longrightarrow \mathrm{S_n}$ be a bijection. Then the average value, Δ_φ, of $\mid \Lambda(s) - \Lambda(\varphi(s)) \mid$ is bounded above by*

$$\begin{cases} \frac{4}{3} - \frac{1}{3(2^{n-2})} & n \text{ even} \\ \frac{5}{3} - \frac{1}{3(2^{n-2})} & n \text{ odd.} \end{cases}$$

For each n, there exists a bijection that attains this bound.

Proof:

The average of the absolute value of the differences in linear complexity between s and $\varphi(s)$ is given by

$$\Delta_\varphi \equiv \frac{1}{2^n} \sum_{s \epsilon \mathrm{S_n}} \mid \Lambda(s) - \Lambda(\varphi(s)) \mid$$

$$\leq \frac{1}{2^n} \sum_{s \epsilon \mathrm{S_n}} \left(\mid \Lambda(s) - \frac{n}{2} \mid + \mid \Lambda(\varphi(s)) - \frac{n}{2} \mid \right).$$

$$= \frac{1}{2^{n-1}} \sum_{s \epsilon \mathrm{S_n}} \mid \Lambda(s) - \frac{n}{2} \mid$$

since φ is a bijection. So we evaluate the sum

$$\mathrm{S} \equiv \sum_{s \epsilon \mathrm{S_n}} \mid \Lambda(s) - \frac{n}{2} \mid .$$

From the distribution of linear complexity, (1), we see that if n is even,

$$\mathrm{S} = \frac{n}{2} + \sum_{k=1}^{n/2} 2 \cdot 4^{k-1} (\frac{n}{2} - k) + \sum_{k=1+n/2}^{n} 4^{n-k} (k - \frac{n}{2})$$

and if n is odd,

$$\mathrm{S} = \frac{n}{2} + \sum_{k=1}^{(n-1)/2} 2 \cdot 4^{k-1} (\frac{n}{2} - k) + \sum_{k=(n+1)/2}^{n} 4^{n-k} (k - \frac{n}{2}).$$

Let us first consider n even. Replacing $n-k+1$ with k in the last sum and combining terms yields

$$S = \frac{n}{2} + \sum_{k=1}^{n/2} 4^{k-1}\left((1+\frac{3n}{2}) - 3k\right).$$

Evaluating the geometric sum and using equation (2), we obtain

$$S = \frac{2}{3}(2^n - 1).$$

and finally,

$$\Delta_\varphi \le \frac{S}{2^{n-1}} = \frac{4}{3} - \frac{1}{3 \cdot 2^{n-2}}.$$

A similar calculation yields the result for n odd.

It is easy to describe a bijection that obtains the upper bound of Theorem 5.1. Start with the strings of highest and lowest linear complexity, working inward, and always choosing for an image the string most distant in linear complexity and not yet used. By the above proof, this bijection, φ, attains the maximum value for Δ_φ.

□

6 Other String Functions

A bijection on S_n will have a beneficial effect (i.e. lowering of linear complexity) on some strings, a detrimental effect on others and no effect at all on the rest. An algorithm that, given a string, $s \epsilon S_n$, tries to produce an LFSR, of "low" linear complexity which generates a sequence whose first n terms differ from s in only a small percentage of its bits, will surely not be a bijection. Ideally, bits will only be altered when the change is beneficial. In general, such an algorithm will induce a function $\varphi : S_n \to S_n$ but φ will not be a bijection. Here, we first consider such a function φ with the restriction that φ transforms only a bounded number of strings to the same image string. It is probably reasonable to expect that a useful algorithm will not transform too many strings to the same image, for example a function that alters no more than k bits. Later, we will drop this condition and require only that linear complexity of the image strings be "low".

Theorem 6.1 *Let $1 \leq k \leq n$ and let φ be a function, $\varphi : \mathrm{S_n} \to \mathrm{S_n}$ such that card$\{\varphi^{-1}(s)\} \leq 2^k$ for all $s \epsilon \mathrm{S_n}$. Then an upper bound for Δ_φ is given by*

$$\Delta_\varphi \leq \frac{k}{2} - \frac{1+2^k}{3 \cdot 2^{n-1}} + \begin{cases} 4/3 & n \text{ even} \quad k \text{ even} \\ 3/2 & n \text{ odd} \quad k \text{ odd} \\ 3/2 & n \text{ even} \quad k \text{ odd} \\ 5/3 & n \text{ odd} \quad k \text{ even} \end{cases} .$$

For each n and k, there exists a function, satisfying the conditions above such that Δ_φ attains this upper bound.

Proof:

As in the analysis of bijections, we have

$$\Delta_\varphi \equiv \frac{1}{2^n} \sum_{s \epsilon \mathrm{S_n}} | \Lambda(s) - \Lambda(\varphi(s)) |$$

$$\leq \frac{1}{2^n} \sum_{s \epsilon \mathrm{S_n}} \left(| \Lambda(s) - \frac{n}{2} | + | \Lambda(\varphi(s)) - \frac{n}{2} | \right)$$

$$= \frac{1}{2^n} \sum_{s \epsilon \mathrm{S_n}} | \Lambda(s) - \frac{n}{2} | + \frac{1}{2^n} \sum_{s \epsilon \mathrm{S_n}} | \Lambda(\varphi(s)) - \frac{n}{2} | .$$

The analysis for bijections gives an upper bound on the first of these sums so we have,

$$\Delta_\varphi \leq \frac{1}{2^n} \sum_{s \epsilon \mathrm{S-n}} | \Lambda(\varphi(s)) - \frac{n}{2} | - \frac{1}{3 \cdot 2^{n-1}} + \begin{cases} 2/3 & n \text{ even} \\ 5/6 & n \text{ odd} \end{cases} . \qquad (3)$$

Now we nust find an upper bound for

$$\mathrm{S} = \sum_{s \epsilon \mathrm{S_n}} | \Lambda(\varphi(s)) - \frac{n}{2} | .$$

This will be maximal when the image values, $\varphi(s)$ are as far as possible from $\frac{n}{2}$. Since each s can have up to 2^k pre-images, we start with the elements of S_n farthest fron $\frac{n}{2}$, assuming each has 2^k pre-images until we have enough pre-images to cover the 2^n elements of S_n. We obtain

<u>$n-k$ odd:</u>

$$S \leq 2^k \left[\frac{n}{2} + \frac{n}{2} + \sum_{j=1}^{M}(\frac{n}{2} - j)(2\cdot 4^{j-1} + 4^j)\right]$$

with $M = \frac{n-k-1}{2}$ as for this value of M;

$$2^k \left[1 + 1 + \sum_{j=1}^{M}(2\cdot 4^j - 1 + 4^j)\right] = 2^n$$

<u>$n-k$ even:</u>

$$S \leq 2^k \left[\frac{n}{2} + \frac{n}{2} + \left(\sum_{j-1}^{M-1}(\frac{n}{2} - j)(2\cdot 4^{j-1} + 2^{2j})\right) + (\frac{n}{2} - M)\left(2\cdot 4^{M-1}\right)\right]$$

with $M = \frac{n-k}{2}$ as for this value of M,

$$2^k \left[1 + 1 + \left(\sum_{j-1}^{\frac{n-k}{2}-1}(2\cdot 4^{j-1} + 4^j)\right) + 2\cdot 4^{\frac{n-k}{2}-1}\right] = 2^n.$$

Regrouping terms, we now obtain

<u>$n-k$ odd:</u>

$$S \leq 2^k \left[n + \sum_{j=1}^{M}\frac{3}{2}4^j(\frac{n}{2} - j)\right]$$

$$= 2^k \left[n + \frac{3n}{4}\left(\frac{4^{M+1} - 4}{3}\right) - 2\left(\frac{(3M-1)4^M + 1}{3}\right)\right].$$

By substituting $M = \frac{n-k-1}{2}$ and reducing we obtain

$$S \leq 2^k \left[n + n(2^{n-k-1} - 1) - \frac{2}{3}\left(\left(\frac{3(n-k-1)}{2} - 1\right)2^{n-k-1} + 1\right)\right]$$

$$= 2^k \left[2^{n-k-1}(k + \frac{5}{3}) - \frac{2}{3}\right]$$

and dividing by 2^n yields

$$\frac{S}{2^n} \leq \frac{1}{2}(k + \frac{5}{3}) - \frac{2^k}{3 \cdot 2^{n-1}} = \frac{k}{2} + \frac{5}{6} - \frac{2^k}{3 \cdot 2^{n-1}}.$$

<u>$n - k$ even:</u>

$$S \leq 2^k \left[n + \left(\sum_{j=1}^{M-1} \frac{3}{2} 4^j (\frac{n}{2} - j)\right) + 2^{2M-1}(\frac{n}{2} - M)\right]$$

$$= 2^k \left[n + 2^{2M-1}(\frac{n}{2} - M) + \frac{3n}{4}\left(\frac{4^M - 4}{3}\right) - \left(\frac{2(3M-4)4^{M-1}+1}{3}\right)\right].$$

By substituting $M = \frac{n-k}{2}$ and reducing we obtain

$$S \leq 2^k \left[2^{n-k-2}(2k + \frac{8}{3}) - \frac{2}{3}\right]$$

and dividing by 2^n yields

$$\frac{S}{2^n} \leq \frac{k}{2} + \frac{2}{3} - \frac{2^k}{3 \cdot 2^{n-1}}.$$

Finally, substituting these upper bounds into the inequality 3 gives the desired results.

As in the case of a bijection, we can construct a function, $\varphi S_n \rightarrow S_n$ that satisfies the conditions of the theorem and attains the upper bound by starting with those elements of S_n which have highest and lowest linear complexity, working inward and always choosing for an image the element of S_n which is still available and maximizes the difference in linear complexity. The only difference is that each element of S_n can be used as an image s^k times.

□

We now look at things from a slightly different angle. We assume that we have an algorithm that takes n-bit strings to n-bit strings and assume that all the image strings are of "low" linear complexity. The following theorem tells us that there must be a string that has "many" bits changed by the algorithm.

Theorem 6.2 *Let $\varphi : \mathrm{S_n} \rightarrow \mathrm{S_n}$ be such that*

$$\Lambda(\varphi(s)) \leq p(\log_2 n)$$

for all $s \epsilon \mathrm{S_n}$ and some polynomial p. Then there is a string, $s \epsilon \mathrm{S_n}$ such that the Hamming distance,

$$H(s, \varphi(s)) \geq \mu n$$

where μ tends to $\frac{1}{2}$ as n tends to infinity.

Proof:

The number of strings within Hamming distance μn of a fixed string, $s \epsilon \mathrm{S_n}$ is given by

$$\sum_{k=0}^{\lfloor \mu n \rfloor} \binom{n}{k} \leq 2^{nH_2(\mu)}$$

where $H_2(x) = -x \log_2(x) - (1-x)\log_2(1-x)$ is the *entropy* function, [2].

If $\varphi : \mathrm{S_n} \rightarrow \mathrm{S_n}$ is such that

$$\Lambda(\varphi(s)) \leq p(\log_2 n) \ll \frac{n}{2}$$

for some polynomial, p, then the number of candidates for $\varphi(s)$ is given by

$$1 + \sum_{k=1}^{\lfloor p(\log_2 n) \rfloor} 2 \cdot 4^{k-1} = 1 + 2\left(\frac{4^{p(\log_2 n)} - 1}{3}\right) = \frac{1 + 2^{2\lfloor p(\log_2 n) \rfloor - 1}}{3}.$$

If $M = \max card \varphi^{-1}$ then, as there are 2^n strings of n bits, we have

$$\left(\frac{2^{2p(\log_2 n)-1} + 1}{3}\right) M \geq 2^n. \qquad (4)$$

From this we see that M must satisfy

$$M \geq \frac{3 \cdot 2^n}{2^{2p(\log_2 n)-1} + 1} \geq 2^{n + 1 - 2p(\log_2 n)}.$$

If all the pre-images of s are within Hamming distance μn of s then

$$2^{nH_2(\mu)} \geq M \geq 2^{n + 1 - 2p(\log_2 n)}.$$

Comparing the exponents, we see that

$$H_2(\mu) \geq 1 - \frac{2p(\log_2 n) - 1}{n}.$$

So $H_2(\overline{\mu}) \longrightarrow 1$ and hence, $\mu \longrightarrow \frac{1}{2}$ as n tends to infinity.

□

From the following two corollaries, we see that any function $\varphi : \mathrm{S_n} \rightarrow \mathrm{S_n}$ such that most of the images of φ have low linear complexity must take many strings to images which are distant in Hamming distance.

Corollary 6.1 *(to the proof of Theorem 6.2) Let* X *be a subset of* $\mathrm{S_n}$ *with card*(X) $\leq 2^{n-1}$. *Let* $\varphi : \mathrm{S_n} \rightarrow \mathrm{S_n}$ *be such that*

$$\Lambda(\varphi(s)) \leq p(\log_2(n))$$

for some polynomial p and for all $s \epsilon \mathrm{S_n} - \mathrm{X}$. *Then there is an element* $s \epsilon \mathrm{S_n} - \mathrm{X}$ *such that the Hamming distance*

$$H(s, \varphi(s)) > \mu n$$

where $\mu \rightarrow \frac{1}{2}$ *as n tends to infinity.*

Proof:

The proof follows as for Theorem 6.2 with two changes. First, we look at the number of candidates for $\varphi(s)$ with $s \epsilon \mathrm{S_n} - \mathrm{X}$. Second, the right-hand side of the inequality (4) becomes $2^n - 2^{n-1} = 2^{n-1}$. The rest follows.

□

Corollary 6.2 *Let* X *be a subset of* $\mathrm{S_n}$ *with* $card(\mathrm{X}) = c \ll 2^{n-1}$. *Let* $\varphi : \mathrm{S_n} \rightarrow \mathrm{S_n}$ *be such that*

$$\Lambda(\varphi(s)) \;\leq\; p(\log_2(n))$$

for some polynomial p *and for all* $s\epsilon\mathrm{S_n} - \mathrm{X}$. *Then there are at least* $2^{n-1} - c$ *elements* $s\epsilon\mathrm{S_n} - \mathrm{X}$ *such that the Hamming distance*

$$H(s, \varphi(s)) \;>\; \mu n$$

where $\mu \rightarrow \frac{1}{2}$ *as* n *tends to infinity.*

Proof:
Enlarge X to cardinality 2^{n-1} by adding the $2^{n-1} - c$ elements of $\mathrm{S_n} - \mathrm{X}$ that have the largest values for $H(s, \varphi(s))$. The result then follows from the last corollary.

□

7 Conclusion and Future Work

The results of this paper tell us two things, of cryptographic importance, about the difference in linear complexity of strings and their neighbors vis-a-vis Hamming distance.
1: For large n there are strings in $\mathrm{S_n}$ which are cryptographically secure in the sense that they are far, in Hamming distance, from any string of "low" linear complexity.
2: There are enough such secure strings that we cannot expect to find an algorithm which for "most" strings produces nearby strings (in Hamming distance) of "low" linear complexity.

There are two paths open for future investigation. One is to classify those sequences which are close (in Hamming distance) to strings of "low" linear complexity. The results of this paper put bounds on how many such strings there are but do not indicate what they look like.

The other line of future research is to study the effect of synchronization errors on the linear complexity of strings. If we say that two strings are $k-$*close* if one can be obtained from the others by a sequence of no more then k errors where we now include added and lost bits as well as changes of bits, then a string is k-close to many more strings than it is within k of in Hamming distance. Theorem 6.2 does not immediately generalize in a useful way. As with Hamming distance, we should classify those strings which are k-close to strings of "low" linear complexity.

References

[1] J. M. Massy. *Shift-Register Synthesis and BCH Decoding.* IEEE Trans. Information Theory 15, 122-127 (1969).

[2] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North-Holland Mathematical Library, Amsterdam, 1977.

[3] R. A. Reuppel. *Analysis and Design of Stream Ciphers.* Springer, Berlin,1986.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique