Black Box Analysis of Stream Ciphers

E Dawson*, H Gustafson, N Davies**

*School of Mathematics
and Information Security Research Centre
Queensland University of Technology
GPO Box 2434
Brisbane Qld 4001
Australia

**Department of Mathematics, Statistics and Operational Research
Nottingham Polytechnic
Nottingham NG1 4BU
United Kingdom

ABSTRACT

In this paper methods are described for analysing the randomness of large binary strings produced by pseudorandom number generators. In particular it will be assumed that such a binary string is being used as the keystream in a stream cipher. Due to the simple nature of a stream cipher it is very important that the keystream have the appearance of randomness. The aim in this paper is to develop fast and efficient methods for measuring the randomness of binary strings. The most significant new result is that many of the standard tests can be replaced by other faster and more efficient methods.

1. Introduction

The use of binary strings which have the appearance of randomness plays an important role in many cipher systems. In particular in a stream cipher the keystream should have the appearance of randomness. A stream cipher is the process of encryption where the plaintext is divided into characters and each character is encrypted separately by a time varying function. The simplest and most often used stream cipher for encrypting binary plaintext is where the bit at time interval t of a pseudorandom binary generator is combined using modulo two addition with the plaintext bit at time interval t to produce the corresponding ciphertext bit. The bits from the pseudorandom generator denote the keystream. In most cases the keystream is formed by a deterministic generator which produces a periodic sequence. The length of a cryptogram should be only a small fraction of the period in order to avoid the cryptanalytic attack described in [12]. This being the case a cryptographer or user of the cipher may be interested in testing binary strings for randomness from such a large periodic sequence where the length of the string is equal in length to that of an average cryptogram. If this string deviates significantly from randomness in some fashion a cryptanalyst may be able to use the decrease in entrophy caused by this deviation together with the redundancy of the plaintext to conduct a successful statistical attack on an intercepted cryptogram. There are various computer based techniques for conducting such an attack described in [4] and [12]. The techniques described in [4] are for attacking simple substitution, transposition and polyalphabetic ciphers used on written text . These attacks are conducted automatically on a computer based on the cryptanalyst knowing the frequency distribution of single characters, digraphs and trigraphs from the language set; and based on the assumption that the cryptanalyst has intercepted the cryptogram. In certain cases these same techniques can be applied to a stream cipher in the case where the keystream is not random. For example suppose that the plaintext consists of standard English text which is coded in 8-bit ASCII. Suppose that this plaintext is encrypted using a stream cipher where 3/4 of the entries in the keystream are zero. Assume that the cryptanalyst knows both the type of plaintext being used (under this assumption the attacker knows the frequency distribution of the plaintext in terms of single characters, digraphs and trigraphs) and the probability of a zero in the keystream. In addition assume that the cryptanalyst has available an intercepted cryptogram. At the start of the attack the probability of any 8-bits of ciphertext corresponding to a plaintext character would be $(3/4)^8$ which is approximately .10. Hence about one in ten 8-bit ciphertext characters are in the clear. However, most other 8-bit ciphertext characters have several bit positions correct. In general only a relatively few of the ASCII characters correspond to most of the plaintext. The cryptanalyst can use a single character frequency distribution to make "educated" guesses on several encrypted 8-bit characters. By combining such guesses on single characters with digraph and trigraph information on the language set it should be a relatively easy task for the cryptanalyst to recover a substantial amount of plaintext. This entire attack procedure can be conducted automatically on the computer.

Two types of measures are commonly used to examine the randomness of binary strings of length n. The first measure examines the hypothesis that the string was based on n Bernoulli trials with the probability of a one on each trial being one half. In a Bernoulli trial as defined in [3] the trials are independent in that the probability of a one (or zero) in a trial does not change the amount of information about the outcomes of other trials. There are several standard statistical tests described briefly in [2] for examining the above hypothesis namely the frequency, serial, poker, runs and autocorrelation tests.

Another method for examining the hypothesis that a binary string of length n was based on n Bernoulli trials is to use the binary derivative as defined in [1]. For a string s of length n the binary derivative of s, written as der(s), is the string of length n-1 formed by combining, modulo two, overlapping 2-tuples from the original string. In a similar manner the kth binary derivative, $\mathrm{der}_k(s)$, is the string of length n-k formed by taking the binary derivative of the k-1 th binary derivative. For example a string of length 16 and its first four derivatives are given by

$$\begin{aligned} s &= 1000100011010101 \\ \mathrm{der}(s) &= 100110010111111 \\ \mathrm{der}_2(s) &= 10101011100000 \\ \mathrm{der}_3(s) &= 1111110010000 \\ \mathrm{der}_4(s) &= 000001011000 \end{aligned}$$

In Section 2 it will be shown that several of the standard binary statistical tests are equivalent to faster and more efficient statistical tests based on the binary derivative. In particular it will be shown that statistical tests based on the binary derivative can replace the serial test, and partially replace the generalised serial test and the autocorrelation test.

In Section 3 a string will be examined for a change point as described in [11]. A change point in a string is a point where the Bernoulli structure changes. For use in a stream cipher it is desirable to have a string with a uniform distributed Bernoulli structure throughout the string. As will be shown in Section 3 a string can be examined for a change point in order to measure the uniform Bernoulli structure in the original string.

A second measure for examining the randomness of a binary string is to obtain the complexity of the deterministic number generator used to form the string. Two different measures of complexity will be discussed in Sections 4 namely linear complexity and sequence complexity. These complexity measures examine the randomness of the string in the fashion in which knowledge of a small subsection of the string can be used to predict the remainder of the string. If this is possible the string would not be considered to be random especially in relation to its use in a stream cipher. For each of these complexity measures a method shall be described for differentiating between random and highly patterned strings. It will be suggested that the sequence complexity measure can be used in place of the autocorrelation test.

The aim in developing the various techniques for measuring the randomness of binary strings will be to have uniform,fast and efficient methods based on statistical tests. In many communications systems such as Fascimile machines or digital speech the length of an average message to be encrypted may be several million bits long. Hence it is useful to have fast and efficient techniques for measuring the randomness of strings of such length. It will be shown how some of the standard tests can be replaced by faster and more efficient tests.

It should be noted that the statistical tests described in this chapter are meant for the designer or user of a new stream cipher for analysis of the randomness of the keystream. It will be assumed in all of these tests that the keystream is being tested from a black box point of view. Based on this assumption the user of the testing procedures described in this paper only needs to have available the output binary sequence of the pseudorandom generator. No knowledge of the actual algorithm used in the generator is required to implement these procuedures.

2. Binary Derivative and Standard Statistical Tests

In this section the equivalence of some of the standard statistical tests from [2] is demonstrated for examining the hypothesis that a strings of length n was based on n Bernoulli trials with a probability of a one on each trial being one half, and tests based on the binary derivative for examining the same hypothesis. The notation $p(i)$ will be used for the fraction of ones in the i-th derivative where $p(0)$ denotes this fraction in the original string and $\pi(i)$ the corresponding population proportions.

2.1 Serial Test

For a string s let $n_{00}, n_{11}, n_{01}, n_{10}$ denote the number of 00, 11, 01, 10 overlapping 2-tuples respectively. Let $\mu_{00}, \mu_{11}, \mu_{01}, \mu_{10}$ denote the corresponding population means.

Let H_0 be the hypothesis

$$H_0\text{: } \pi(0) = \pi(1) = \tfrac{1}{2}. \qquad (1)$$

The hypothesis that $\pi(0)$ is 0.5 is equivalent to

$$\mu_{00} + \mu_{01} = \frac{n}{2} \qquad \left(\text{or } \frac{n}{2} - 1\right); \tag{2}$$

$$\mu_{10} + \mu_{11} = \frac{n}{2} \qquad \left(\text{or } \frac{n}{2} - 1\right). \tag{3}$$

The hypothesis that $\pi(1) = 0.5$ is equivalent to

$$\mu_{00} + \mu_{11} = \frac{n}{2} \qquad \left(\text{or } \frac{n}{2} - 1\right); \tag{4}$$

$$\mu_{01} + \mu_{10} = \frac{n}{2} \qquad \left(\text{or } \frac{n}{2} - 1\right). \tag{5}$$

In addition a 01 must occur in a string between two 10 pairs. Hence

$$n_{10} = n_{01} \text{ or } n_{01} + 1 \text{ or } n_{01} - 1$$

and so

$$\mu_{10} = \mu_{01} \text{ or } \mu_{01} + 1 \text{ or } \mu_{01} - 1. \tag{6}$$

(2) - (6) in four unknowns are equivalent to the hypothesis that

$$\mu_{00} = \mu_{10} = \mu_{01} = \mu_{11} = \frac{(n-1)}{4} \tag{7}$$

The hypothesis in (7) is the hypothesis of the serial test. Hence the null hypotheses associated with $\pi(0)$ and $\pi(1)$ for the original string s and its first binary derivative are equivalent to the null hypothesis of the serial test.

A comparison was made between the execution times on a personal computer of applying the serial test together with the frequency test, and calculating p(0) and p(1). A string of length one million bits was used. The execution time for computing p(0) and p(1) for the string was 128 seconds. In comparison the execution time for computing the statistics for the frequency and serial tests for this string was 248 seconds. Hence the test for randomness on a string of million bits using the binary derivative is almost twice as fast as using a combination of the frequency and serial tests on the computer used.

2.2 Generalised Serial Test

One can show that the hypothesis based on the binary derivative that

$$\pi(0) = \pi(1) = ... = \pi(i) = 0.5 \tag{8}$$

is partially equivalent to the hypothesis of the generalised i + 1-serial test. For example if n_{ijk} denotes the number of overlapping 3-tuples in the string of the form ijk then it can be shown that the hypothesis that

$$\pi(0) = \pi(1) = \pi(2) = 0.5 \qquad (9)$$

is equivalent to

$$\mu_{000} = \mu_{011} = \mu_{110} = \mu_{101} \; ; \qquad (10)$$

$$\mu_{111} = \mu_{100} = \mu_{001} = \mu_{010} \qquad (11)$$

where μ_{ijk} correspond to the population mean number of triples of the form ijk. The expressions (10) and (11) are derived in a similar fashion used to find (7).

It should be noted that it is possible for a string to be classified as random using the hypothesis (9) when in fact the string would be classified as patterned under the hypothesis of the generalised 3-serial test that all μ_{ijk} are equal. For example let s denote the string of length 42 given by

$$s = 101101101101101101101101101101100000000000$$

This string would be classified as random using the hypothesis (9) since $p(0) = 0.5$, $p(1) = 0.512$ and $p(2) = 0.525$. This string clearly has an inbalance in the distribution of n_{ijk} since

$$n_{011} = n_{110} = n_{101} = 10$$
$$n_{000} = 9$$
$$n_{100} = 1$$
$$n_{111} = n_{001} = n_{010} = 0$$

The above string would be classified as patterned under the hypothesis of the generalised 3-serial test that the μ_{ijk} are equal. By using an appropriate scaling factor one can clearly define a large string which satisfies (10) and (11) yet is unbalanced in the distribution of the n_{ijk} in that the values in (11) are approximately zero.

2.3 The Autocorrelation Test

For a string s of length n let s^i be the substring of length $n - i$ formed by deleting the last i entries of s. The autocorrelation test at lag i calculates the number of matches between s^i and the substring y^i formed by deleting the first i places of s. The expected number of matches if s is based on Bernoulli trial with a probability of a one on each trail being $\frac{1}{2}$, is $\frac{n-i}{2}$. The binary derivative can be used in place of the autocorrelation test for lags of $1, 2, \ldots, 2^j$ for $2^j \leq \frac{n}{2}$ since as shown in [6]

$$\text{der}_{2^i}(a_1 a_2 \ldots a_n) = a_1 \oplus a_{2^i+1}, a_2 \oplus a_{2^i+2}, \ldots, a_{n-2^i} \oplus a_n. \qquad (12)$$

where $\oplus$ denotes modulo two addition. Hence the number of zeros in $\text{der}_{2^i}(a_1 a_2 \ldots a_n)$ is the number of matches between s^{2^i} and y^{2^i}.

3. Change Point Problem

If a binary string is to be used as the keystream in a stream cipher it should have an approximately uniform distributed Bernoulli structure throughout the string. A string with 3/4 ones in the first half of the string and 1/4 ones in the second half would pass the frequency test. However, such a string would be considered to be highly patterned due to its nonuniform distribution of ones and zeros. Clearly if such a string was used as the key stream in a stream cipher, a cryptanalyst who was aware of such a change could use this information to conduct a statistical attack on the intercepted ciphertext.

The place in a string where the largest change occurs in the distribution of ones and zeros is called a change point. The change point problem is the problem concerned with finding this change point and measuring its significance. In [11] there is described a method for solving the change point problem. A brief description of this method is given as follows:

For a string s of length n let

S denote the number of ones in the string;

S_t denote the number of ones in the first t entries;

$$U_t = n(S_t - \frac{tS}{n}).$$

The hypothesis to be examined is that there is no significant change in the distribution of ones and zeros in the string. The point in the string where there is the greatest change in Bernoulli structure occurs at the absolute value maximum of the U_t which we shall denote as K. In order to test the above hypothesis a significance probability is calculated using the value of K. The approximate significance probability associated with this change point denoted by P_{sig} is given by

$$P_{sig} = \exp[-2K^2(Sn^2 - n S^2)^{-1}]. \qquad (13)$$

It should be noted that the above approximation of the significance probability of the change point is conservative in that the exact significance probability is less than or equal to this value.

The change in Bernoulli structure of the string s of length 42 given in Section 2.2 can be highlighted by applying the above formulae. For this string the change point occurs at position 31. The significance probability of this change point from applying the formula in (13) to this string is .003. Hence this change point is highly significant.

As demonstrated by the above example the change point statistic can be a powerful technique for identifying highly patterned strings. In particular as shown in this example the change point statistic provides a method of identifying highly patterned strings which are classified as random when only a few binary derivatives are used to examine the distribution of ones and zeros in the string.

As shown in [11] the above test for a change point can be applied to examine the hypothesis that a string was based on n Bernoulli trials with the probability of a one on each trial being π independently of the value of π. In this case it would be assumed that before the change point test has been applied one has already applied the frequency test to test the hypothesis that $\pi=1/2$.

4. Complexity Measures

A deterministic binary number generator is not considered to be complex for a certain measure if it is possible to predict a large string produced by the generator from knowledge of a small subsection of the string. If such a noncomplex string is used in a stream cipher as the keystream a cryptanalyst might be able to find a small amount of known plaintext of the same length required to predict the keystream. The cryptanalyst may be easily able to find such plaintext by using the redundancy of the language set i.e. silence periods in digital speech. In the case where the attacker is conducting wiretapping the equivalent portion of the keystream can be found. Using this portion of the keystream and the complexity weakness in the cipher an attacker may be able to derive the entire keystream and hence decrypt an entire intercepted cryptogram. Clearly, the number generator should be classified as complex with respect to any measure for which such an attack can be formed. There are two different complexity measures which will be discussed in this section namely linear complexity and sequence complexity. For each of these measures methods to differentiate between complex and noncomplex strings will be described.

4.1 Linear Complexity

An important complexity measure which can be used to differentiate between random and patterned strings is linear complexity. The length of the shortest linear feedback shift register (LFSR) which produces a given finite or periodic sequence is the linear complexity of the sequence. If the linear complexity of a sequence is L then the equivalent LFSR from [9] used to form the sequence can be found by applying the Berlekamp-Massey algorithm provided 2L consecutive terms of the sequence are known. Given L consecutive terms of the sequence and the defining LFSR of length L the entire sequence can be reconstructed by substitution into the recurrence relation given by this LFSR. Hence, in order to avoid sequence reconstruction as described above the value of L should be large. An example of a periodic sequence with good random statistical properties in terms of the distribution of ones and zeros but a small linear complexity is an m-sequence as described in [2]. A binary string consisting of the first period of an m-sequence would be classified as random using the standard statistical tests to test the hypothesis that the string was based on Bernoulli trials with the probability of a one on each trial being 1/2. However an m-sequence has a very small linear complexity in relation to its period length. In this fashion an m-sequence should not be used as the keystream in a stream cipher even though it has the "best" possible distribution of ones and zeros for a string with such a length.

For a binary string s of length n, the symbol L(s) will be used to denote the linear complexity of s. In [13] there are several properties developed for L(s) which we will use to differentiate between random and patterned strings. Three statistical tests for L(s) will be developed to measure the local randomness of s in terms of linear complexity based on applying these properties. It should be noted that these tests are dependent on computing the value of L(s) using the Berlekamp- Massey algorithm.

As shown in [13] for large n the expected value of L(s), E(L(s)), and variance of L(s), V(L(s)), for random strings are approximately given by $E(L(s))=n/2$ and $V(L(s))=86/81$.

The first statistical test on linear complexity is derived from applying Chebyshev's inequality [7] to these values of E(L(s)) and V(L(s)). From Chebyshev's inequality for large n the probability that L(s) differs from n/2 by k or larger is less than or equal to $86/(81k^2)$. For example for k=10 at least 99% of all random strings have a linear complexity within the range of n/2-10 and n/2+10 for a sufficiently large value of n. In this fashion one can test the hypothesis that the linear complexity of a string of length n is n/2 by comparing the actual linear complexity of the string with a confidence interval derived from Chebyshev's inequality.

There are some problems associated with using the above test as the only method for testing the linear complexity of a string. This test by itself would classify as random strings which may be highly patterned or contain large substrings which are highly patterned. For example the string of length n consisting of n/2-1 zeros, followed by a one and then followed by repeating these n/2 terms has a linear complexity of n/2. This string would be classified as being random using the above test for linear complexity. Clearly, such a string is highly patterned and would fail all the standard statistical tests. However it is possible to construct a string of length n which would pass the standard statistical tests based on the Bernoulli measure described in this paper which the would have a linear complexity of approximately n/2 and yet which would contain a large highly patterned substring. For example suppose that the first half of the string was formed by taking a large substring from an m-sequence and the second half of the string was formed by a "good" pseudorandom number generator. In this case it would be possible to predict the first half of the string based on knowing only knowing a few initial terms since the linear complexity of this part of the string is constant after a few terms.

Methods for avoiding the above problem has been suggested by several authors including [5], [10], [13] and [14]. These methods use the concept of the linear complexity profile of a string. For a string s of length n let s(i) be the substring formed by taking the first i bits of s. Let L(s(i)) for i=1,...,n denote the linear complexities of the s(i). The values of L(s(i)) are defined to be the linear complexity profile of s. From the above test the linear complexity profile of a random string should follow approximately the i/2 line. In [9] it is shown that the values of L(s(i)) are subject to the following constraints

$$\text{if } L(s(i-1)) > i/2 \text{ then } L(s(i))=L(s(i-1)); \tag{14}$$

$$\text{if } L(s(i-1)) \leq i/2 \text{ then } L(s(i))=L(s(i-1)) \text{or } L(s)=i-L(s(i-1)). \tag{15}$$

Using the above inequalities two statistical tests will be described for examining the hypothesis that a string has the the expected linear complexity profile of a random string. Each of these tests will be based on using the concept of a jump. If there is a change in linear complexity from s(i-1) to s(i) then this is called a jump. If a change in the linear complexity occurs at s(i) then the size of the jump is given by i-2L(s(i-1)) from (15). Hence if k bits occur between jumps the height of a jump is k. In [5] a chi-squared goodness of fit test is designed based on the number of jumps of height k as follows:

For a large string s of length n let the total number of jumps in linear complexity be F where f(k) is the number of jumps of height k. It can be shown that for a random string based on Bernoulli trials where the probability of a one on each trial is one half that the probability, p(k), that a given jump has height k is $(1/2)^k$. The hypothesis of the test is that the expected number of jumps of height k,e(k), in s is given by

$$e(k)=p(k)F. \tag{16}$$

The hypothesis as given by (16) can be tested using the chi-squared goodness of fit test from [3] to compare the values of the e(k) and f(k).

An alternative test for the linear complexity profile can be formed using the standard normal statistic. In [10] it is shown that the expected value and variance of the height of a jump are each two. For the given string s let u be the average height of the jumps. The value of u can be standardised using the standard normal statistic given by

$$z = \sqrt{F}\,(u - 2)/\sqrt{2} \tag{17}$$

where F is the total number of jumps in the string as stated above. It should be noted the statistic given by (17) should only be evaluated in the case where the total number of jumps is larger than 30.

It is possible for a highly patterned string to have a good linear complexity profile. In [15] an infinite binary sequence $y = y_1y_2\ldots$ is said to have a perfect linear complexity profile if the string of length i consisting of the first i terms of the sequence for all i has a linear complexity of $[(i+1)/2]$ where [ ] denotes the integer part of the number. In [13] it is conjectured and proven in [15] that the infinite sequence y whose terms are given by

$$y_j = \begin{cases} 1 \text{ if } j = 2^t \text{ for } t = 0, 1, 2, \ldots \\ 0 \text{ otherwise} \end{cases} \tag{18}$$

has a perfect linear complexity profile. Clearly any finite string taken from the above sequence would be highly patterned especially since there is a very bad balance in the distribution of zeros and ones. A sufficiently large string taken from the starting of the sequence defined by (18) would pass the statistics test based on the normal distribution given by (17). On the other hand this string would fail the above chi-squared test on the linear complexity profile given by (16). This suggests that the chi-squared test is a better test for identifying highly patterned strings.

4.2 Sequence Complexity

Another important complexity measure which can be used to differentiate between random and patterned strings is sequence complexity. For a string **s** of length n the sequence complexity of s, written as c(s), denotes the number of new patterns in s as one moves from left to right along the string. For example the string s = 1001111011000011110 has a value of six for c(s) since s can be broken into 1/0/01/1110/1100/001110 where new patterns are separated by a "/".

For large strings s of length n it is shown in [8] that the threshold level of n/log(n) can be used to distinguish between patterned and random strings in terms of sequence complexity. It should be noted that log(n) is evaluated base two. A string with a value of c(s) below this threshold level would be considered to be patterned.

The sequence complexity measure can replace the autocorrelation test as described in Section 2.3. The autocorrelation test examines a string for any periodic type behavior within the string. If there is such a periodic behavior then there is a repetition of a significant portion of the string. Such a repetition implies that the same patterns are being repeated in the string. In this fashion a string which has a high correlation between two or more components of the string would have a low sequence complexity. For example let **y** be a string of length n/2. Let **s** be the string of length **n** formed by the concatenation of **y** with itself. The value of c(s) is either c(y) or c(y)+1 depending on the format of the last pattern in s. The periodic behavior in the string **s** would be demonstrated by the fact that c(s) is significantly less than n/log(n).

It is much faster to determine c(s) for a string of length n than to evaluate all the different autocorrelation values at lags i=1,...,n/2. In Table 1 there is a list of the times required to evaluate c(s) and the autocorrelation test at lags $i = 1, 2, \ldots, \frac{n}{2}$ on a personal computer. It can be seen that it is much faster to evaluate c(s) on the computer used. It should be noted that it is very fast to evaluate the autocorrelation test if one only wants to test a few specific lags.

Length of String	Sequence Complexity Time in Seconds	Autocorrelation Time in Seconds
1000	1	4
2000	4	16
4000	15	60
8000	55	241

Table 1
Comparison of Times to Compute Sequence Complexity and Autocorrelation

It is possible to have a string which is classified as random using one of the measures of linear complexity or sequence complexity when the string would be classified as patterned using the other complexity measure. For example let **s** be a large string selected from the infinite sequence as given by (18). As mentioned in Section 4.1 such a string would be classified as random using the statistics test as defined by (17). However, this string would clearly be considered to be patterned using the sequence complexity measure since there are only a small number of new patterns formed in the string as one moves from the left to the right along the string. As a second example let **z** be the string of length $2^L - 1$ consisting of the first period of an m-sequence. Such a string would be considered to be random using the sequence complexity measure since every L bits in the string defines a different L bit pattern. However, this string would be classified as patterned using the linear complexity measure since the linear complexity is very small in relation to the string length.

5. Conclusion

Methods for measuring the randomness of a binary string which is to be used as the keystream in a stream cipher have been described. These methods have been concerned with the analysis of two hypotheses. The first hypothesis is that the string is based on Bernoulli trials with the probability of a one on each trial being 1/2. In general this hypothesis is equivalent to the hypothesis that the string represents the first period of a sequence which satisfies Golomb's Postulates as described in [2]. In [14] it is shown that a keystream in a stream cipher which deviates slightly from one or more of these postulates causes only a slight decrease in entrophy. For example it is shown that if the probability of a one in the keystream is 1/2+e instead of 1/2 then the amount of information leaked to an attacker is quadratic in e under the assumption that the attacker is aware of this probability i.e. if e is of the order of .001 then the amount of information per bit leaked to the cryptanalyst is of the order .000001. The results for Golomb's second and third postulates are similar. Hence slight deviations from the above hypothesis by a string should be allowed when applying one or more of the tests in standard statistical tests from [2] and the tests described in Sections 2 and 3.

The second hypothesis examined in this chapter is that the string is complex in that it is not possible to predict the string based on knowledge of a smaller subsection of the string. There were two complexity measures for examining this hypothesis, namely linear complexity and sequence complexity. Various methods were described for examining whether a string is classified as being complex with respect to each measure.

The aim in this paper was to develop fast and efficient methods for examining binary strings for randomness. It was shown that one can replace the serial test with a faster statistical test based on the binary derivative. In addition it was shown that the generalised serial test can be partially replaced by a statistics test based on higher binary derivatives. However in this case there are certain highly patterned strings which will be classified as random when using a statistic test based on higher binary derviatives alone and will be classified as patterned using a generalised serial test. As was shown one can overcome this problem by determining the significance of the largest change point in the string. The change point statistic allows one to identify a change in distribution of ones and zeros within a string. It was shown that the autocorrelation test can be replaced by the sequence complexity measure. It was shown that it is much faster to evaluate sequence complexity for a string than to apply the autocorrelation test.

In conclusion it would be recommended that the following set of tests be included in a computer based package for analysis of the randomness of a binary string

(a) poker test with hand size depending on the format of plaintext;
(b) runs test;
(c) statistical test (8) based on the distribution of ones and zeros in the string and its first three binary derivatives;
(d) statistical tests based on linear complexity profile of string;
(e) test as defined by (13) to measure the significance of the largest change point in the string;
(f) sequence complexity measure.

It would not be recommended to use a generator for forming the keystream in a stream cipher in the case where several binary strings from this generator equal in length to an average cryptogram fail one or more of the above set of tests at a highly significant level. However, there may be a weakness in the design of the generator which a cryptanalyst can use, provided the actual format of the generator is known, which will not be identified by any of the above tests.

6. References

1. Barbe, A. "Binary Random Sequences: Derivative Sequences and Multi-Level α-Typical Randomness", **Proceedings of the 8th Benelux Symposium on Information Theory**, 1987, pp. 21-38.

2. Beker, H. and Piper, F. **Cipher Systems: The Protection of Communications**, 1982, John Wiley and Sons.

3. Bhattacharyya, G. and Johnson, R., **Statistical Concepts and Methods**, 1977, John Wiley and Sons.

4. Carroll, J. and Robbins, L., "Computer Cryptanalysis", **Technical Report No. 223**, 1988, Dept. of Computer Science, The University of Western Ontario, London, Ontario.

5. Carter, G., "A statistical test for randomness based on the linear complexity profile of a binary sequence", **Technical Report for Racal Comsec Ltd.**, 1987.

6. Dawson, E., Davies, N. and Gustafson, H., "Binary derivative and random binary strings", **Technical Report No. 4/90, School of Mathematics, Queensland University of Technology**, 1990.

7. Kreyszig, E., **Introductory Mathematical Statistics**, John Wiley and Sons, 1970.

8. Lempel, A. and Ziv, J., "On the complexity of finite sequences", **IEEE Trans. on Information Theory**, Vol. IT-22, Jan. 1976, pp. 75-81.

9. Massey, J.L., "Shift register sequences and BCH decoding", **IEEE Trans. on Information Theory**, Vol. IT-15, Jan. 1969, pp. 122-127.

10. Niederreiter, H., "The probabilistic theory of linear complexity", in **Proc. EUROCRYPT '88, Lecture Notes in Computer Science**, Bol. 330, 1988, pp. 191-209.

11. Pettitt, A.N., "A non-parametric approach to the change-point problem", **Appl. Statist.**, Vol. 28 No. 2, 1979, pp. 126-135.

12. Rubin, F., "Computer methods for decrypting random stream ciphers", **Cryptologia**, Vol. 2 No. 3, 1978, pp. 215-231.

13. Rueppel, R.A., **Analysis and Design of Stream Ciphers**, Springer-Verlag, 1986.

14. Wanders, H.E., "On the significance of Golomb's randomness postulates in cryptography", **Philip J. Res.**, Vol. 43, 1988, pp. 185-222.

15. Wang, M., "Cryptographic Aspects of Sequence Complexity Measures", PhD. Thesis, Swiss Federal Institute of Technology, 1988.