Stream Ciphers

RSA Laboratories Technical Report TR-701
Version 2.0—July 25, 1995

M.J.B. Robshaw
matt@rsa.com

RSA Laboratories
100 Marine Parkway
Redwood City, CA 94065-1031

003-903040-200-000-000

Contents

1 Introduction **1**

2 General background **2**

3 Classification **3**

4 Analysis **5**
4.1 Appearance 5
4.1.1 Period 5
4.1.2 Statistical measures 7
4.2 Measures of complexity 8
4.2.1 Linear complexity 8
4.2.2 Other measures of complexity 10
4.3 Some theoretical results 12

5 Congruential generators **13**

6 Shift register based schemes **14**
6.1 Linear feedback shift registers 14
6.2 Combination and filter generators 16
6.2.1 Correlation attacks 16
6.2.2 Two weak generators 17
6.2.3 Boolean functions 18
6.2.4 Three more attacks 19
6.3 Multiplexers 19
6.4 Clock control 20
6.4.1 Stop and go with variants 21
6.4.2 Cascades 22
6.5 Shrinking and self-shrinking generator 23
6.6 Summation generator 24

7 Alternative designs **25**
7.1 RC4 25
7.2 SEAL 25
7.3 Number-theoretic techniques 26
7.4 Other schemes 27
7.4.1 $1/p$ generator 27
7.4.2 Knapsack generator 27
7.4.3 PKZIP 28

7.5 Final examples 28
7.5.1 Randomized ciphers 28
7.5.2 Cellular automata 29

8 Conclusions **29**

1 Introduction

Cryptosystems are divided between those that are *secret-key* or *symmetric*, and those that are *public-key* or *asymmetric*. With the latter, the sender uses publicly known information to send a message to the receiver. The receiver then uses secret information to recover the message. In secret-key cryptography, the sender and receiver have previously agreed on some private information that they use for both encryption and decryption. This information must be kept secret from potential eavesdroppers.

There is a further division of symmetric cryptosystems into *block ciphers* and *stream ciphers*. The distinction between block and stream ciphers is perhaps best summarized by the following quotation due to Rueppel [126]:

> *Block ciphers* operate with a fixed transformation on large blocks of plaintext data; *stream ciphers* operate with a time-varying transformation on individual plaintext digits.

In this technical report we provide a review of current stream cipher techniques. Anyone looking through the cryptographic literature will be struck by a great difference in the treatment of block ciphers and stream ciphers. Practically all work in the cryptanalysis of block ciphers is focused on DES [94] and nearly all the proposed block ciphers are based in some way on the perceived design goals of DES. There is no algorithm occupying an equivalent position in the field of stream ciphers. There are a huge variety of alternative stream cipher designs and cryptanalysis tends to be couched in very general terms.

While much of this distinction might be attributed to the publication of DES as a Federal Standard and the subsequent high profile of this algorithm within both the business and cryptographic communities, there may well be an additional and more subtle factor to consider.

We will see in this report that many stream ciphers have been proposed which use very basic building blocks. The mathematical analysis of these components has been very advanced for some considerable time (see Section 6.1) and intensive design and cryptanalysis over the years has resulted in the formulation of a set of ground rules for the design of stream ciphers. It is well known that highly developed analytic techniques facilitate both design and cryptanalysis.

No such well-developed list could be given, until very recently, for block ciphers. The design criteria for DES were not published and cryptanalysis was, for a long time, frustratingly unsuccessful. There seemed to be

little general theory available. With the advent of differential cryptanalysis [7] and, more recently, linear cryptanalysis [80], both designers and cryptanalysts had new and clear-cut issues to consider and there has been considerable recent activity in both the design and analysis of block ciphers.

Curiously research into stream ciphers seems to be a predominantly European affair. By comparing the proceedings of the two major cryptography conferences we often see that Eurocrypt meetings dedicate several sessions to stream cipher issues whereas a single session is more often the norm at the US Crypto meetings. This imbalance in interest may well be a by-product of the pre-occupation with DES in the US, however an increased profile for stream ciphers can be expected as more developers look to stream ciphers to provide the encryption speeds they need.

This report aims to provide a snapshot of the different techniques available today and to report on the direction and status of both prior and contemporary research. We will avoid considerable depth on the different topics since we aim to 'cover the ground' and prefer to point the reader to sources of further details. For a more detailed source of information on stream ciphers, we strongly recommend the excellent article by Rueppel in *Contemporary Cryptology* [126] though there are also other survey articles in the literature [51, 138].

2 General background

Much of the popularity of stream ciphers can undoubtedly be attributed to the work of Shannon in the analysis of the *one-time pad* [1], originally known as the Vernam cipher [130].

The one-time pad uses a long string of *keystream* which consists of bits that are chosen completely at random. This keystream is combined with the plaintext on a 'bit by bit' basis. The keystream is the same length as the message and can be used only once (as the name one-time pad implies); clearly a vast amount of keystream might be required. We write the plaintext message m as a sequence of bits $m = m_0 m_1 \dots m_{n-1}$ and the binary keystream k which is the same length as the message as $k = k_0 k_1 \dots k_{n-1}$. The ciphertext $c = c_0 c_1 \dots c_{n-1}$ is defined by $c_i = m_i \oplus k_i$ for $0 \leq i \leq n-1$ where $\oplus$ denotes bitwise exclusive-or.

In his seminal paper [125] Shannon proved what many had previously believed, namely that the one-time pad is 'unbreakable'. In fact Shannon

[1] This name was adopted following its use during the Second World War with the help of a paper pad.

described the cryptosystem as being *perfect*; even an adversary with an infinite amount of computing power is unable to do better than guess the value of a message bit since the ciphertext is statistically independent of the plaintext.

Because of the practical problems involved with a system requiring such a vast amount of key information, the Moscow-Washington hotline used to be cited as perhaps the only place where the requirements for secrecy outweighed the problems of key management. Somewhat disappointingly Massey reports [126] that this is no longer the case and a conventional secret-key cipher requiring much less key is used instead.

A stream cipher attempts to capture the spirit of the one-time pad by using a short key to generate the keystream which *appears* to be random. Such a keystream sequence is often described as *pseudo-random* and deciding what constitutes a pseudo-random sequence forms much of the work in the field of stream ciphers. We will say that the keystream is generated by the *keystream generator*; other terms in the literature include *pseudo-random sequence generator* and *running key generator*.

Stream ciphers can be very fast to operate; they are generally much faster than block ciphers. Since the keystream can often be generated independently of the plaintext or ciphertext such generators often have the advantage that the keystream can be generated prior to encryption or decryption, with only an easy combining step left when the message or ciphertext is to be processed.

3 Classification

We will very informally describe the *state* of a cryptosystem as the values of a set of variables that together provide a unique description of the status of the device.

When we design a stream cipher there are essentially two concerns. The first is how to describe the next state of the cryptosystem in terms of the current state, and the second is how to express the ciphertext in terms of the plaintext and the state.

The second issue is perhaps the easiest to solve since almost invariably the ciphertext is expressed as the bit-wise exclusive-or of the plaintext and a function of the state of the cryptosystem. (Combining functions other than exclusive-or might also be considered.) The sequence generated by the function of the state of the cryptosystem is conventionally known as the keystream.

The choice of the expression of the next state of the cryptosystem, which constituted our first design decision, provides us with a classification of stream ciphers into two types.

If the next state of the cryptosystem is defined independently of both plaintext and ciphertext then the stream cipher is termed *synchronous*.

In such a scheme each plaintext bit is encrypted independently of the others and the corruption of a bit of the ciphertext during transmission will not affect the decryption of other ciphertext bits. The cipher is described as having no *error-propagation* and though this appears to be a desirable property, it has several implications. First, it limits the opportunity to detect an error when decryption is performed, but more importantly an attacker is able to make controlled changes to parts of the ciphertext knowing full well what changes are being induced on the corresponding plaintext.

Of more practical significance, both the encrypting and decrypting units must remain in step since decryption cannot proceed successfully unless the keystreams used to encrypt and decrypt are synchronized. Synchronization is usually achieved by including 'marker positions' in the transmission; the net effect being that a bit of ciphertext missed during transmission results in incorrect decryption until one of the marker positions is received.

In contrast *self-synchronizing* or *asynchronous* stream ciphers have the facility to resume correct decryption if the keystream generated by the decrypting unit falls out of synchronization with the encrypting keystream. For these stream ciphers the function that defines the next state of the cryptosystem takes as input some of the previously generated ciphertext. The most common example of this is provided by some block cipher in what is termed cipher-feedback (CFB) mode [95].

Suppose the encryption of a bit depends on c previous ciphertext bits. The system demonstrates limited error propagation; if one bit is received incorrectly then decryption of the following c bits may be incorrect. Additionally however, the system is able to resynchronize itself and produce a correct decryption after c bits have been received correctly. This makes such ciphers suitable for applications where synchronization is difficult to maintain.

Self-synchronizing stream ciphers have some limited error propagation which may or may not be viewed as an advantage. Certainly any changes made by an attacker to the ciphertext will have additional consequences on other parts of the decrypted plaintext. However Rueppel suggests [114] that there are two drawbacks to self-synchronizing stream ciphers.

First, an opponent knows some of the variables being used as input to the generator since this input is taken from the ciphertext. Second, these

generators have a limited analyzability because the keystream depends on the message. Nevertheless, the design of self-synchronizing stream ciphers has been addressed to a limited extent in the literature [101, 26] and Maurer [81] has provided some framework for a general assessment of the security offered by these stream ciphers. The rest of this report is concerned with synchronous stream ciphers or keystream generators.

4 Analysis

There are many different considerations we must keep in mind when we consider the suitability of a keystream generated by some stream cipher. The criteria we list here provide only some of the necessary conditions for the security of the keystream; a keystream might well satisfy all these conditions and yet still be vulnerable to some attack.

Over the years a vast number of different considerations have been highlighted and they seem to fall into one of two camps. The first group are used to assess the appearance of the keystream; is there some imbalance to the way the sequence is generated that allows a cryptanalyst to guess the next bit with some probability better than that of random guessing?

The second group of criteria address the ability of a cryptanalyst to use the bits of the keystream he might already have, to construct a sequence that replicates the keystream. In some way we are considering the inherent complexity of the sequence and attempting to answer the question — is it hard to reproduce the sequence?

Finally, Section 4.3 describes attempts to provide a firm theoretical basis for the security offered by stream ciphers.

4.1 Appearance

4.1.1 Period

The first attribute of a sequence, and one of the most important to consider, is the length of the period. The usual model for a keystream generator is provided by a *finite state machine*.

In this model the machine is regularly clocked and at each clocking instant the internal *state* of the machine is updated in a way that is determined by its current state. At the same time some of the keystream is output. Since there are a finite number of states available it is clear that eventually some internal state will occur twice. Since the successor state and the output are determined by the current state, then from the point of repetition on, the

keystream will be identical to that produced when the same state previously occurred.

If the period of the keystream is too short then different parts of the plaintext will be encrypted in an identical way and this constitutes a severe weakness. Knowledge of the plaintext allows recovery of the corresponding portion of the keystream and the cryptanalyst can then use the fact that this portion of keystream is used elsewhere in the encryption to successfully decrypt the ciphertext. Additionally, if only the ciphertext parts are received then they can be combined to give a stream of data that equals the combination of two plaintext messages and is independent of the key. The underlying statistics of the plaintext source might then be used to derive both the plaintexts and the keystream.

During the Second World War, the Lorenz SZ-42 cipher machine was used to encrypt messages sent from German High Command to various Army Commands. According to Good [57], who worked on the cryptanalysis of this cipher, one of the biggest single advances came when a radio operator used the same keystream twice to encrypt two different messages. This allowed cryptanalysts to construct a machine which mimicked the action of the SZ-42 and paved the way for the subsequent successful cryptanalysis of this cipher.

The question of how large a period is required for a sequence is open to debate and depends on the application in mind. We note however, that with a stream cipher encrypting at a speed of 1 Mbyte/sec, a sequence with period 2^{32} will repeat itself after only 2^9 seconds or 8.5 minutes. This would not generally be considered adequate.

When the modes of use for the DES block cipher were first published [95] some flexibility was provided in one of the parameters for the OFB mode; this uses the block cipher as a keystream generator. An important parameter was allowed to vary from 8 bits through to 64. It was soon discovered that for all choices except 64 the period of the generated keystream was very likely to be around 2^{32} which as we have previously noted is inadequate. Instead the period of the keystream generated when using 64 as the parameter is around 2^{63} which is more acceptable. Consequently DES should only be used in the OFB mode with a feedback of 64 bits [29].

A good assessment of the period of the keystreams generated by a keystream generator is essential to the design of any stream cipher. Practically, the keystream should be long enough to ensure that it is overwhelmingly unlikely that the same portion of keystream is used twice during encryption.

On a technical point we note that many of the theoretical results in the

later sections are obtained by considering what is termed a *period* of the sequence. Most often the period of some sequence refers to the *number* of bits before the sequence recurs. At other times, however, a period is said to consist of p successive bits of the sequence where p is the length of the period. The context provided by the text should avoid any confusion between these two uses.

4.1.2 Statistical measures

When repeatedly tossing a fair coin one expects to see roughly as many heads as tails. In a similar fashion many other properties can be formulated to describe the appearance of a sequence that is purported to be generated by a totally random source. One of the first formulations of some basic ground-rules for the appearance of periodic pseudo-random sequences was provided by Golomb [52] and these three rules have come to be known as Golomb's postulates.

G1 The number of 1's in every period must differ from the number of 0's by no more than one.

G2 In every period, half the runs must have length one, one quarter must have length two, one eighth must have length three etc. as long as the number of runs so indicated exceeds one. Moreover, for each of these lengths, there must be equally many runs of 1's and of 0's.

G3 Suppose we have two copies of the same sequence of period p which are off-set by some amount d. Then for each d, $0 \leq d \leq p-1$ we can count the number of agreements between the two sequences, A_d, and the number of disagreements, D_d. The auto-correlation coefficient for each d is defined by $(A_d - D_d)/p$ and the auto-correlation function takes on several values as d ranges through all permissible values.

For a sequence to satisfy G3, the auto-correlation function must be two-valued.

G3 is a technical expression of what Golomb has described as the notion of independent trials: that knowing some previous value in the sequence is essentially of no help in deducing the current value. Another view of the auto-correlation function is that it is some measure of the ability of being able to distinguish between a sequence and a copy of the same sequence that has been started at some other point in the period.

A sequence satisfying G1–G3 is often termed a *pn-sequence* where *pn* stands for *pseudo–noise*. However, it is clear that these rules on their own are not sufficient to capture the full significance of a random looking sequence and a wide range of different *statistical tests* can be applied to a sequence to assess how well it fits the assertion that it was generated by a perfectly random source [4, 32, 67, 114].

We expand a little on quite what we mean when we test a keystream generator. Suppose a sequence of length p is generated at random and this finite sequence of p bits is repeated to form a periodic sequence. (Such a sequence is sometimes called a *semi-infinite* sequence.) If the p bits were generated completely at random, then any pattern of the p bits would be equally likely. In particular, the sequence consisting of p zero bits (which are then repeated) would be as likely to occur as any other. When we test the generator, we test many sequences individually and assess what proportion of the sequences generated fail the tests we apply. If the failure rate is comparable to that expected for sequences generated using a perfectly random source then we pass the generator. Now of course for cryptographic purposes even sequences generated by a perfectly random source might be wholly unsuitable for encryption, such as the example given above, and so the design of the generator should ensure that catastrophically weak sequences can never be generated.

4.2 Measures of complexity

As we mentioned previously, a great deal of work has centered on providing an adequate measure of how hard a sequence might be to replicate. The most popular technique by far, is the linear complexity; we shall describe this next. Meanwhile, attempts to develop either new techniques or more general measures of complexity have also had some success; we shall describe some of these approaches in Section 4.2.2.

4.2.1 Linear complexity

One of the most far-reaching papers in stream cipher analysis is due to Massey [77]. In this paper, an algorithm now called the *Berlekamp-Massey algorithm*, is described which identifies the shortest linear recurrence that can be used to generate a finite binary sequence. Since a linear recurrence can be implemented using a *linear feedback shift register* (see Section 6.1) this result is often described in terms of the shortest linear feedback shift register that can be used to generate a sequence.

Every sequence $s_0 s_1 \ldots$ of period p satisfies a linear recurrence of length p, namely $s_{i+p} = s_i$ for all $i \geq 0$. A sequence may additionally satisfy a shorter recurrence, that is each bit of the sequence can be defined using some linear expression which involves bits that are less than p bits away. The length of the shortest recurrence is defined to be the *linear complexity* (or *linear span*) of the sequence.

Given a finite sequence, the Berlekamp-Massey algorithm can be used to calculate this recurrence over what is mathematically termed a *field*; an example of a field is the set of binary numbers under the operations of addition and multiplication modulo two, hence its applicability to pseudo-random bit generators. An extension of the Berlekamp-Massey algorithm is provided by Reeds and Sloane [103] which acts over more general number systems; those that form what are mathematically called *rings*.

While the Berlekamp-Massey algorithm calculates the linear complexity of a finite sequence, its use can be easily extended to periodic or semi-infinite sequences. It is a very important algorithm since the linear recurrence satisfied by a sequence with linear complexity k can be efficiently calculated after observing $2k$ consecutive bits of the sequence. Since the linear recurrence also defines a linear feedback shift register it offers some indication for how difficult a sequence might be to replicate. A high linear complexity means that more of the sequence has to be observed before the recurrence can be identified and that a longer register is required to duplicate the sequence.

While a high linear complexity is a necessary condition, the following example serves to show that it is not a sufficient condition on the suitability of the keystream. Consider the periodic sequence of period p consisting of a single 1 with the remaining bits set to 0. In this case the linear complexity is p since no linear relation shorter than $s_{i+p} = s_i$ for all $i \geq 0$ will be satisfied by every bit of the sequence. However it is clear that as a keystream such a sequence is useless since $p - 1$ bits are zero.

Klapper [63] demonstrates another important consideration for the linear complexity of the keystream, namely that the sequence must have a high linear complexity not only when considered bitwise, but also when the sequence is viewed as numbers (which happen to be 0 or 1) over fields of *odd characteristic*. In particular Klapper notes that *geometric sequences* [20] might well be susceptible to this kind of analysis.

Rueppel [114] additionally proposes the use of the *linear complexity profile* in the analysis of stream ciphers. After each bit is added to the keystream the linear complexity of the sequence seen so far is calculated; the value of the linear complexity can be plotted against the number of bits that have been examined, thereby giving a 'profile' of the sequence. Rueppel proved

several very important theorems concerning the linear complexity profile and managed to obtain expressions for the expected behavior of the linear complexity profile of a sequence for which each bit is generated at random [114].

This so-called ideal linear complexity profile has been widely studied [96]. Rueppel established that the linear complexity profile for a perfectly random source closely follows the line $y = \frac{x}{2}$; a conjecture was posed specifying a class of sequences which possess the ideal linear complexity profile, that is which sequences have a profile which follows the line $y = \frac{x}{2}$ as closely as possible. This conjecture was proved by Dai [28] and a full characterization of all sequences with an ideal linear complexity profile was provided by Wang and Massey [132].

There are other algorithms for identifying the linear complexity of a binary sequence; some more practical than others. One very interesting algorithm due to Games and Chan [39, 108] is exceptionally elegant but can be used only on sequences with period 2^n for which an entire period of the sequence is known. While this appears to be an algorithm with limited general applicability the mathematical foundations were used to prove some very interesting results [40, 38] in the study of what are called *de Bruijn* sequences [15, 35]. Very recently the validity of this algorithm has been extended by Blackburn [9] to sequences with any period, although an entire period of the sequence is still required for its application.

4.2.2 Other measures of complexity

As a generalization of the linear complexity a great deal of research was completed by Jansen [58] who looked closely at algorithms for evaluating the *maximum order complexity* of a sequence. This is a generalization of the linear complexity in that the recurrence that relates bits of the sequence need no longer be linear. Thus, when given a sequence, it might be possible to identify a much shorter recurrence that can be used to recreate the sequence using a nonlinear feedback shift register instead of one which is entirely linear.

There are some very obvious relationships between the linear and the maximum order complexity. For instance the maximum order complexity is always less than or equal to the linear complexity. However the expected behavior of the maximum order complexity for a randomly generated sequence is not easily established [58]. Consequently, the practicality of the maximum order complexity in a statistical test tends to be somewhat limited though it is of significant theoretical interest.

In many ways the maximum order complexity seems to be less easily related [92, 58] to the linear complexity than to another measure of complexity called the *Ziv-Lempel complexity* (or Lempel-Ziv complexity to some commentators) [141, 142]. The Ziv-Lempel complexity and the maximum order complexity are, in fact, based on quite different principles since the Ziv-Lempel complexity provides some measure of the rate at which new patterns are generated within the sequence. The origins of the Ziv-Lempel complexity lie within the field of data compression and the observation that a random sequence cannot be significantly compressed. Unfortunately the usefulness of this measure is also currently limited since it is difficult to define a practical test statistic with which to evaluate the performance of a generator.

The reason that results on the maximum order complexity and the Ziv-Lempel complexity can be closely related is due to the fact that both complexities can be computed using what is called a *suffix tree* [32, 97]. The linear complexity cannot be computed in this way and this makes relating the linear complexity to the Ziv-Lempel complexity difficult. Jansen [58] describes a related technique for calculating the maximum order complexity using *directed acyclic word graphs*.

The use of the suffix tree points out that a measure of complexity can only be useful if it can be efficiently calculated for the sequence of interest. This is the major stumbling block with the development of the *quadratic span* [21].

The quadratic span lies between the linear complexity and the maximum order complexity since it is concerned with using quadratic recurrences to generate a sequence. At present it is only a measure of theoretical interest since there is no efficient way to calculate the quadratic span of a sequence. Consequently there are as yet few results on the calculation of the quadratic span for a binary sequence [19, 62] and even less on the expected values for a randomly generated sequence. If, however, these problems can be overcome then the quadratic span will almost certainly be another useful measure of complexity.

Finally we mention the *2-adic span* [64, 53] of a sequence. This fascinating new technique has allowed the cryptanalysis of the previously proposed *summation* stream cipher [113]. Perhaps more importantly this work shows the way toward a rigorous mathematical formalism of another class of shift registers and while much research has concentrated on the cryptanalytic potential of this technique, there are also results of interest to the designer [65, 66].

4.3 Some theoretical results

The statistical testing of a keystream reflects what some commentators have described as the *system theoretic* approach to stream cipher design [126]. The designer uses the tests that are available and if some statistical weakness in a class of sequences is discovered a new test is devised and added to the set. This is very much an ad hoc method of analysis and many prefer to see some firm theoretical foundations on which the security of stream ciphers might be based. While a great deal of theory has been established and there have been several proposals attempting to 'prove' the security provided by some keystream, there still remains a wide gulf between the work of the researchers and that of the practitioners.

Much of the theoretical work was stimulated by the work of Yao [135]. Yao succeeded in tying together the two essential concepts we consider a requisite in a keystream generator; those are the ideas of predictability and random appearance. In short, and somewhat casually, Yao showed that a pseudo-random generator could be 'efficiently' predicted if, and only if, the generator could be 'efficiently' distinguished from a perfectly random source.

Following on this result, techniques in the field of computational complexity have been used to prove results which relate the problem of predicting the next bit in a pseudo-random sequence to the difficulty of solving a 'hard' problem [11, 12, 123]. We shall look into some of the generators proposed as a result of this work in Section 7.3.

We have already seen an interesting link (provided by the Ziv-Lempel complexity [141]) between techniques used in the field of data compression and the analysis of pseudo-random sequences. There is another technique due to Maurer called the *universal statistical test* [83] which is linked to the field of data compression. This test was developed with particular interest in the use of a pseudo-random bit generator for obtaining keys for use in a symmetric-key cryptosystem.

Expressing the strength of a cryptosystem against exhaustive search in terms of the length of the key might be misleading if the keys are not chosen uniformly. In such a case an opponent can search through the keys that are more likely to occur giving an improved chance of quickly obtaining the correct key. Maurer's test evaluates the *entropy per output bit* of the generator thereby reflecting the cryptographic strength of a system when the generator is used to obtain keys in a cryptographic application.

Returning to questions of complexity, one of the earliest attempts at assessing the complexity of a sequence was provided by Chaitin [18] and Kolmogorov [70] who attempted to define the complexity of a sequence in terms

of the size of the smallest *Turing machine* that could be used to generate the sequence. The Turing machine is a simple, but powerful, conceptual computing device which is often used in the theory of computing. While considering the Turing machines' ability to reproduce a sequence leads to a theoretically interesting measure now called the *Turing-Kolmogorov-Chaitin* complexity, it is of little practical significance since there is no way to compute it [5]. The charmingly titled paper "On the Complexity of Pseudo-Random Sequences — or: If You Can Describe a Sequence It Can't be Random"[5] provides a link between the Turing-Kolmogorov-Chaitin complexity and the linear complexity. More information on the Turing-Kolmogorov-Chaitin complexity can be found in the work of Chaitin [17] and Martin-Löf [75].

5 Congruential generators

Some of the earliest practical systems were intended to act as pseudo-random number generators rather than keystream generators. While the problems are closely related, much of the motivation for pseudo-random number generation comes from problems in statistical testing and the cryptographic value of sequences generated by these techniques can often be questioned.

A *congruential generator* is often used to generate random numbers and the next number x_{i+1} in a sequence of numbers x_i is defined in the following way

$$x_{i+1} = (ax_i + b) \bmod m.$$

There are many results about the different forms of a, b and m and their inter-relation to obtain a sequence of pseudo-random numbers with large period [67].

For cryptographic use the numbers generated should not be predictable; if the modulus m is known then it is easy to solve for a and b given two consecutive numbers in such a sequence. Knuth considers a variation of this generator where the modulus m is a power of two [68] but only the high order bits of the numbers are output; this bears a striking similarity to some work of Dai [27] which is concerned with generating similar sequences using linear feedback shift registers.

Some results on congruential generators are as follows. Marsaglia [74] questions the claims of sufficient 'random behavior' for sequences produced using linear congruential generators and Reeds [102], Knuth [68], Plumstead [99], Hastad and Shamir [56] and Frieze, Kannan and Lagarias [37] have all cast considerable doubt on the cryptographic value of sequences generated using the multiplicative congruential generator. A paper by Frieze, Hastad,

Kannan, Lagarias and Shamir [36] and one by Boyar (Plumstead) [13] undermine confidence in techniques to use fragments of integers derived from linear congruences.

While the generation of such sequences can be convenient and there are analytic results which provide assurances on some of the basic properties of the sequences generated, linear congruential generators cannot be recommended for cryptographic use. Surprisingly perhaps, a paper by Lagarias and Reeds [72] implies that there might be little extra cryptographic security gained by moving to more sophisticated recurrences which involve polynomial expressions. Krawczyk [71] has extended both this work and that of Plumstead to apply a very general analysis to the problem of predicting sequences generated using different forms of polynomial recurrence relation.

6 Shift register based schemes

The vast majority of proposed keystream generators are based in some way on the use of linear feedback shift registers [4]. There are two primary reasons for this: a class of sequences they generate ideally capture the spirit of Golomb's Postulates (Section 4.1.2) and their behavior is easily analyzed using algebraic techniques.

6.1 Linear feedback shift registers

Linear feedback shift registers are very familiar to electrical engineers and coding theorists [10] and they are very suited for high speed implementations since they are easily implemented in both hardware and software. The two environments generally utilize different implementations of the linear feedback shift register, termed the *Fibonacci* and the *Galois* registers respectively, but all theoretical results of major importance are valid for both types.

A linear feedback shift register consists of a number of *stages* numbered say from left to right as $0 \ldots n-1$ with feedback from each to stage $n-1$. The contents of the n stages of a register describe its *state*. The description of the action of the register is perhaps easier for the Fibonacci register, certainly it is the most commonly described, so we shall consider the Fibonacci register here. The register is controlled by a clock and at each clocking instance the contents of stage i are moved to stage $i-1$. The contents of stage 0 are output and form part of the sequence while the new contents to stage $n-1$, which is now conceptually empty, are calculated as some linear function

of the previous contents to stages $0 \ldots n-1$, the particular function being dependent on the feedback used.

For completeness we shall briefly describe the Galois register. While each stage of the Galois register is updated according to the contents of the stage immediately to its right (as in the Fibonacci register) the feedback taps also determine whether the prior contents of stage 0 are exclusive-ored into each stage of the register. Thus, in contrast to the Fibonacci register where feedback is a function of all stages in the register and the result is stored in the rightmost stage, feedback in the Galois register is potentially applied to every stage of the register, though it is a function of only the leftmost stage.

Despite the implementation differences between these two forms of the linear feedback shift register, the important thing to note is that for a register of length n, a sequence with maximum period has period $2^n - 1$ (since there are 2^n states and the state $0 \ldots 0$ cannot occur) and satisfies Golomb's Postulates. Actually this isn't too remarkable; Golomb formed the postulates with these so-called *m-sequences* in mind and every m-sequence is a pn-sequence[2]. What is remarkable is that conditions on the generation of such m-sequences can be easily identified and this makes the analysis of these sequences particularly straightforward.

There clearly has to be a drawback to such sequences that can be easily and quickly generated and seem to have good properties of random appearance. The drawback is that they only have linear complexity n since they are generated using an n-stage linear feedback shift register. Consequently, the Berlekamp-Massey algorithm (Section 4.2) can be used on $2n$ successive bits of the output sequence to deduce the feedback and the initial state of the register used to generate the sequence.

All shift register based schemes try to exploit the good characteristics of sequences generated using linear feedback shift registers in such a way that the new sequences are not susceptible to attacks based on their linear complexity. Somewhat casually; the designers of shift register based schemes are attempting to introduce sufficient nonlinear behavior into the generation of the sequences to hinder successful cryptanalysis. As Massey is quoted [137] as saying:

> Linearity is the curse of the cryptographer.

The essential theoretical background to the study of linear feedback shift

[2] The question of whether every pn-sequence is an m-sequence (or its binary complement) was answered in 1981 when Cheng [22] discovered a pn-sequence which could not be derived from an m-sequence.

registers and related topics is laid down by the work of Selmer [122] and Zierler [139]. The work of Ward [133] is often overlooked despite the fact that many important results were derived a considerable time ago. Much work has also concentrated on producing a parallel foundation to the theory of non-linear feedback shift registers, see for example the work of Ronce [111], but the lack of a convenient and general mechanism for this analysis is a major handicap.

6.2 Combination and filter generators

When using linear feedback shift registers there are two obvious ways to generate an alternative output. The first is to use several registers in parallel and to combine their output in some (hopefully) cryptographically secure way. A generator like this is conventionally called a *combination* generator. Another alternative is to generate the output sequence as some nonlinear function of the state of a single register; such a register is termed a *filter* generator.

We have described this technique in very general terms and there is little value in listing specific choices for the functions used. Clearly, bounding the period and the linear complexity of the sequences generated are important issues [114, 79, 117]. In addition, unless great care is taken in deciding which types of function to use, these generators may be susceptible to what have been termed *correlation attacks*. We will go into more detail on this subject in Section 6.2.1.

The combination of several sequences may well involve the use of what is termed the *Hadamard product*. The product of two sequences is formed bitwise and we expect to see more 0's than 1's in a sequence formed as the product of two other sequences which had a roughly equal distribution of 0's and 1's. While this imbalance in the product sequence tends to increase the linear complexity considerably, the excessive number of 0's potentially provides a cryptographic loop-hole which the cryptanalyst can exploit. Much of the theoretical background on the linear complexity of such product sequences can be found in the work of Zierler [140] and Rueppel and Staffelbach [116]. Recently Gottfert and Niederreiter [54] proved bounds on the linear complexity of product sequences.

6.2.1 Correlation attacks

A correlation attack is a widely applicable type of attack which might be used with success on generators which attempt to combine the output from

several cryptographically weak keystream generators.

A correlation attack exploits the weakness in some combining function which allows information about individual input sequences to be observed in the output sequence. In such a case, there is a correlation between the output sequence and one of the internal sequences. This particular internal sequence can then be analyzed individually before attention is turned to one of the other internal sequences. In this way the whole generator can be deduced - this is often called a *divide-and-conquer* attack.

Correlation attacks were first introduced by Siegenthaler [121, 119, 120]. Since it is immediately clear that some combining or filter functions are more susceptible to attack than others, the idea of an m^{th}-order correlation-immune function was introduced [119]. When at least $m+1$ internal sequences must be simultaneously considered in a correlation attack the function is said to be m^{th}-order correlation-immune. In the same paper [119] Siegenthaler showed that there was an interesting trade-off between the linear complexity of the output sequence and the order of correlation immunity; greater correlation immunity meant a reduced linear complexity.

Brynielsson [16] examined how this problem might be adapted to other non-binary fields and research by Rueppel [113] showed how the use of memory could be used to separate the ideas of correlation immunity and linear complexity in the binary case. The *summation* generator [113] (see Section 6.6) introduced the idea of a combining function with memory and it was established that with this combining function it is possible to attain maximum-order correlation and maximum linear complexity simultaneously. Meier and Staffelbach [86] have provided more complete details about the correlation properties of combiners and the role of memory.

Additional work on correlation attacks, and some improvements in efficiency, can be found in [23, 87, 42, 90, 2, 91, 45]. Other interesting results have been established and we shall describe some of them in Section 6.2.3 where the issues addressed are more suitably expressed in terms of Boolean functions.

6.2.2 Two weak generators

Since they don't fall easily into any other section, we will mention here two early and simple proposals for keystream generators which use multiple registers and are susceptible to correlation attacks.

The first is the Geffe generator [41] which was later analyzed by Key [61] and also cryptanalyzed using the linear syndrome algorithm [136] (Section 6.2.4). This generator uses three linear feedback shift registers, the third

being used to 'choose' whether the bit that is output comes from the first register or the second. While such a generator has some nice properties, it is susceptible to a correlation attack. The success of correlation attacks also defeated the Pless generator [98] which used a widely available logic device, the *J-K flip-flop*, to combine the outputs from eight linear feedback shift registers.

6.2.3 Boolean functions

It is interesting to observe that with the topic of Boolean functions the design of stream ciphers and block ciphers are once again related. The interest in Boolean functions for block ciphers follows from the design of S-boxes in DES-like block ciphers [106]. Some of the conditions required for good S-box design are essentially the same as the requirements for good combining functions.

Meier and Staffelbach [85] consider a measure of the distance of an arbitrary Boolean function from the nearest linear function and introduce the idea of a *perfect nonlinear function*. A Boolean function taking n inputs is perfect nonlinear if the output changes with probability 1/2 whenever i input bits are complemented for $1 \leq i \leq n$. It so happens that the notion of perfect nonlinear functions coincides with the idea of *bent* functions [112]. These have already been well researched in other areas of mathematics and have been connected with functions used in the design of S-boxes [118].

A second issue of interest in the field of S-box design is the so-called *Strict Avalanche Criterion (SAC)* [34, 73]. A Boolean function $f(x)$ satisfies SAC if the output changes with probability 1/2 whenever exactly one of the input bits changes. This property is useful both in the design of S-boxes and in the design of combining functions. So is a generalization of SAC, m^{th}-order SAC. A function $f(x)$ satisfies m^{th}-order SAC if, when any m input bits to $f(x)$ are kept constant, the output changes with probability 1/2 when one of the remaining input bits changes.

Preneel et. al. [100] have unified these ideas with the concept of the *propagation criterion of degree k*. A Boolean function is PC of degree k if the output changes with probability 1/2 whenever i input bits are complemented for $1 \leq i \leq k$. As a consequence we have that the idea of perfect nonlinear is equivalent to PC of degree n and the property of SAC is equivalent to PC of degree 1.

Counting and constructing families of Boolean functions which satisfy various desirable cryptographic properties forms a very active area of research. In particular much work is concerned with finding an acceptable

balance between often conflicting requirements.

6.2.4 Three more attacks

In this section we briefly describe three types of attacks that have been proposed in the literature.

The *linear consistency test* [137] attempts to efficiently identify some subset of the key used for encryption. The idea is that a matrix $A(K_1)$ is devised for which the entries identify the generator being used. This matrix is parameterized by some subkey K_1 of the complete key K. With some portion of output sequence b the cryptanalyst attempts to find some x such that the matrix equation $A(K_1)x = b$ is consistent.

If a solution is found then it can be shown that provided the portion of output sequence b is large enough, the solution is unique and the correct subkey K_1 has been identified. Thus a search need only be performed on all possible subkeys K_1 until a consistent solution is found. In this way an attack relevant to the entire key K can be mounted.

The second attack uses what is termed the *linear syndrome algorithm* [136]. This attack is essentially a generalization of the work of Meier and Staffelbach [85] and relies on being able to write a fragment of captured output sequence b as $b = a + x$ were a is a sequence generated by a known linear recurrence and x is a sparse unknown sequence, where a sparse sequence consists of more 0's than 1's. This algorithm was particularly successful in the cryptanalysis of both the Geffe generator [41] (Section 6.2.2) and the stop-and-go generator [6] (Section 6.4.1).

Finally, Golić [43] has proposed the *linear cryptanalysis* of stream ciphers. An extension of earlier work [42] and related to other simultaneous work [44], this technique is potentially applicable to a wide variety of stream cipher proposals.

6.3 Multiplexers

A *multiplexer* is a logic device that selects one input from a set of inputs according to the value of another *index* input. Sequences based on the use of multiplexers were initially popular because they are relatively fast and have some nice provable properties [59]. The keystream generator is conventionally described using two sequences (often m-sequences for ease of analysis) and the multiplexer is used to combine these two sequences in a highly nonlinear way.

At each clocking instance a fixed pattern of k bits is taken from the

first sequence. These k bits are viewed as the binary representation of a number modulo 2^k and this number is then mapped using a fixed and known mapping into some other number n. (Various conditions are imposed on k and n to ensure that the mapping is sensibly defined.) The number n is used to choose some bit from the second sequence which then forms part of the output sequence. In effect, the keystream generator uses a multiplexer to select bits from the second sequence according to the values of certain bits in the first sequence.

The sequences that result generally have a large period and linear complexity [59]. However a technique known as the linear consistency attack [137] has been used to show how the choice of mapping adds little to the security of the system and it is concluded that the security of multiplexed sequences might have been previously over-estimated. Work by Daemen [26] has highlighted another possible avenue for mounting an attack on multiplexers and has undermined [25] a European Broadcast Union proposal for audio-video scrambling [31].

6.4 Clock control

Some of the earlier attempts to introduce non-linearity into the generation of the keystream use the idea of varying the rate at which a register is clocked. Recall that in the conventional interpretation of a shift register, the register is clocked regularly and the contents of the stages updated at each clocking instance. If some arrangement is devised so that the clocking of one register is in some way dependent on another register, then it seems reasonable to suppose that more complex sequences will be generated.

While it is undoubtedly the case that sequences generated by these techniques tend to be more complex than any of the constituent sequences, there is a convenient framework for their analysis and some structure is inherited in the output sequence. In fact recent theoretical work by Golić and O'Conner [46] shows that most clock control keystream generators are at least in theory susceptible to attacks termed *embedding* and *probabilistic correlation attacks*. However these techniques can not in general be readily extended into practical attacks.

While some of the simpler clock control techniques have not withstood close analysis, more involved designs seem to perform quite well. Baum and Blackburn [3] discuss a generalization of this technique and a thorough survey of clock-control techniques is provided by Chambers and Gollmann [51].

6.4.1 Stop and go with variants

Among the first investigations were those into what was termed the 'stop-and-go' generator [6]. In this simple scenario two registers were connected so that the second register was clocked if the output of the first register was a 1, otherwise the second register repeated its previous output. Sometimes this output was then exclusive-ored with the output sequence from a third register. It is not surprising that the repetition of bits in the first output sequence roughly half the time leads to poor statistical properties and unfortunate cryptographic consequences [138, 136].

Alterations can be made to this basic model by making the first, or *motor*, register into one which steps the second register twice when a 1 is output by the first register and only once otherwise; this arrangement requires that the second register can run at twice the speed of keystream output but it certainly has improved statistics. Other generator arrangements, including one by Günther [55] which has both improved statistics and a constant register to keystream rate, have been proposed and many properties have been established [127, 129, 131].

Perhaps surprisingly, it is very straightforward to establish a firm theoretical basis for the analysis of such sequences. Changing the clocking pattern of one register merely ensures that the output sequence is some *decimated* or sampled version of the original. The underlying linear algebra can then be used to establish an expression for the new sequence in terms of the old, and bounds on the period and the linear complexity can usually be readily established.

Interestingly, the operations of decimation (that is, removing bits from a sequence) and interleaving or interlacing (that is, combining sequences together) are powerful tools in the investigation of many alternative generators [107]. Often the major difficulty is that the bounds we obtain on the linear complexity of the sequences are upper bounds; for practical purposes we generally wish to obtain a lower bound. Additionally it is often difficult to establish the conditions that define when the upper bound is achieved. Some form of lower bound can sometimes be obtained by considering the period of the sequence since a lower bound on the period (perhaps obtained by using combinatorial techniques) can usually be translated into a result on the linear complexity of the sequence. Other more general theoretical results on both what are termed the *regular* and *irregular* decimation of sequences have been obtained [47].

After considerable early interest in clock-controlled registers during the mid 1980's due to the provably high periods and linear complexities of the

resultant sequences, there has been a slackening of research interest. However work has continued into *cascades* of registers (see Section 6.4.2) which are often viewed as a generalization of the stop-and-go type register arrangements.

Finally in this section we remark on two other variations. Rueppel [115] obtained bounds on the linear complexity and period of the output from a single register whose output controls its own clock. While such a register should not be used as it stands as a keystream generator, Rueppel reports [126] that several modifications have been suggested which might make it useful in a cascade of registers. Second, a generator called the *multiple-speed* generator [76] which uses two registers clocked at different speeds, has many interesting theoretical properties though it is vulnerable to the linear consistency attack [137] (Section 6.2.4).

6.4.2 Cascades

The main idea behind cascades is to extend the simple stop-and-go type arrangements of the previous section into a string of registers for which the output of the first is used in some way to control the clock of the second, the output of the second is used for the third and so on. Two major types of cascades have been studied, the first where each of the registers generates an m-sequence and the second where each of the registers is of length p, where p is prime, and there is no feedback from any intermediate stage of the register. Such registers are called *purely cycling* registers of length p.

Much of the early theoretical work on cascades took place in tandem with proposals for clock-control [49, 51]. The beauty of cascades is that they are conceptually very simple and they can be used to generate sequences with vast periods and similarly vast, and guaranteed, linear complexity [49]. They also seem to have good statistical qualities [49, 48].

However they are prone to an effect which has been termed *lock-in* [50]. A cryptanalyst might try to reconstruct the input to the last register by using the captured output sequence of the generator and guessing the relevant, but unknown, parameters for the last register. Lock-in ensures that many related guesses will suffice to allow the reconstruction of the input sequence to the last register. In this way a cryptanalyst can unravel a cascade register by register, and the net result of lock-in is a reduction in the effective key-space of the cascade generator. This can be a serious weakness in certain situations, though precautions can be taken to reduce the effectiveness of a cryptanalytic attack based on the lock-in effect.

Rather ingeniously, Chambers and Gollmann point out that if the cas-

cade is used as an encryption mechanism with plaintext used as input to the first stage (rather than the all 1 sequence when used as a keystream generator), then the effect of lock-in can be used constructively to regain synchronization after an error in transmission.

Some very recent cryptanalytic results on cascades are due to Menicocci [89]. It is claimed that there is always some correlation between the output sequence from the first register and the output of the cascade, and if this remains significant then information about the first register is leaked in the output sequence. Menicocci suggests that this might form the basis for an attack; to ensure that this effect is not exploitable, Menicocci suggests that a cascade should be at least 10 registers long.

6.5 Shrinking and self-shrinking generator

These two closely related generators have been proposed recently.

Shrinking generator

The *shrinking generator* was proposed by Coppersmith, Krawczyk and Mansour [24]. It uses techniques similar to clock-control and in fact it has been pointed out that the generator can be viewed as implementing a form of variable clock control. One result of this equivalence is that the theoretical results of Golić and O'Conner [46] (Section 6.4) are equally applicable to the shrinking generator and the self-shrinking generator.

In effect, a shrinking generator is implemented by taking two sequences that are generated in parallel. At any clocking instance a bit is output from the second sequence if the first sequence outputs a 1, otherwise nothing is output.

Like practically all of the previous schemes, when the two source sequences are m-sequences bounds on the period and linear complexity of the resultant sequence can easily be obtained. Also, like many of the sequences built out of m-sequence building blocks, the statistical appearance of the sequences is generally good.

It is interesting to apply the techniques of decimation and interleaving to these output sequences and it can be shown that they can be considered as the interleaving of many off-set copies of some m-sequence. While it is too early to decide whether this is significant to a cryptanalyst, it is clear that there is considerable underlying structure in these sequences.

On the practical front this generator is very fast though it suffers from the problem that the output rate is not regular. A buffering technique is

suggested [24] to get around this problem though it is not clear how great a problem this irregular rate might be in practice.

Self-shrinking generator

The *self-shrinking generator* is a variant of the shrinking generator and was proposed by Meier and Staffelbach at Eurocrypt '94 [88]. Instead of generating the indexing sequence and the sequence to be shrunk from two different registers, they are both derived from the same register. While this reduces the amount of space required for an implementation it does mean that the sequence is generated at roughly half the speed of the shrinking generator.

There is some duality between the shrinking and the self-shrinking generator. It is easy to verify that any shrinking generator can be implemented using some self-shrinking generator and vice versa. However the shrinking generator equivalents to the self-shrinking generators proposed by Meier and Staffelbach do not possess the same form as those proposed by Coppersmith et al. [24] and so the previous results on the shrinking register cannot be carried over.

While there appears to be considerable unexplained behavior in the sequences produced using the self-shrinking generator, Meier and Staffelbach have proved lower bounds on both the period and the linear complexity. Consequently parameters in an implementation of a self-shrinking generator can be chosen to ensure adequate performance in these regards.

6.6 Summation generator

It is well known [128] that integer addition can be used as a nonlinear combiner; the carry in integer addition is a nonlinear function of the low-order bits of the numbers being added.

Rueppel uses this fact in a generator known as the *summation* generator. Here the outputs from several shift registers are combined using a mechanism involving integer addition. This provides a combining function with good nonlinearity and high-order correlation properties [113]. Importantly it also provides an example of the role of memory in removing the trade-off between high nonlinearity and the correlation-immunity of a function (see Section 6.2.1). Though the work of Meier and Staffelbach [85] on simple summation generators, and that of Klapper and Goresky [64] more generally seems to have compromised the security offered by this particular generator, it remains a theoretically interesting technique.

7 Alternative designs

It will come as no surprise that there are several important generators and general techniques which don't really fit into the scheme of the report so far. This penultimate section includes proposals which have not been covered in this report and considers other very important and widely used techniques.

7.1 RC4

The RC4 stream cipher [104] was designed by Ron Rivest in 1987 for RSA Data Security, Inc. Like its companion block cipher RC2, RC4 is a variable-key-size cipher suitable for fast bulk encryption. It is very compact in terms of code size, and it is particularly suitable for byte-oriented processors. RC4 can encrypt at speeds of around 1 Mbyte/sec on a 33MHz machine and, like RC2, has special status by which the export approval process is considerably simplified [33].

While RC4 is a confidential and proprietary stream cipher its security does not depend on the confidentiality of the algorithm. Its design is quite distinct from the methods we have already seen and uses a random permutation during the generation of the keystream. There are no known bad keys and though there is no proof for the lower bound of the periods of the sequences generated using RC4, theoretical analysis has established that the period is overwhelmingly likely to be greater than 10^{100}. A thorough and extensive analysis into the security of RC4 [109] has found no reason to question the security offered by the RC4 keystream generator.

7.2 SEAL

SEAL, which stands for software-optimized encryption algorithm, is a recently published stream cipher designed by Rogaway and Coppersmith [110]. SEAL is described as a length-increasing pseudo-random function and this can clearly be used as a keystream generator for a stream cipher. This stream cipher is geared towards 32-bit architectures and encryption requires about five machine instructions per byte.

SEAL requires a large amount of pre-computation to initialize several large look-up tables which total approximately 3 Kbytes in size. This initialization procedure makes repeated use of the compression function which lies at the heart of the Secure Hash Algorithm [93]. The algorithm was optimized with a particular range of popular processors in mind and since these processors were among those that are more difficult to optimize for, it

is expected that an implementation will perform well on any modern 32-bit processor.

Since SEAL is so new there has not been enough time to allow for an assessment of the security offered, but it marks a welcome new addition to the different design techniques available for stream ciphers.

7.3 Number-theoretic techniques

In this section we consider some designs for keystream generators for which the ability to predict the keystream is in some way related to the solution of what is considered to be a 'hard' problem.

There are many well known examples of problems which are considered to be hard; perhaps the most commonly cited are inverting the RSA cryptosystem [105], establishing quadratic residuosity [69] and solving what is called the discrete logarithm problem [69]. The aim of the designer is to ensure that any successful method of predicting the keystream can then be used to successfully solve some difficult problem. Under the assumption that this problem is in reality intractable, this implies that the keystream cannot be efficiently predicted. The work of Yao [135] is then cited which then provides the final link to show that the keystream cannot be efficiently distinguished from a perfectly random source.

While these generators have considerable theoretical appeal, there are some considerations we should keep in mind. First, the difficulty of a problem is usually expressed using techniques in the field of study known as complexity theory. Such results are asymptotic in nature, that is they describe the difficulty of a problem in terms of an increasingly large instance of the problem. The element of provability for which we are striving, is thus asymptotic in nature and it is lost when we move to a problem instance of fixed size. Nevertheless, these techniques do provide us with a scale by which the security of a system can be quantified against a problem that is known to be difficult to solve in practice. More importantly perhaps, the number theoretic operations that these schemes use tend to be slow. As a result these keystream generators tend to have poor performance attributes. We shall merely list here some of the proposals in the literature and provide some initial references for the interested reader.

Shamir [123] relates the security of a generator to inverting the RSA cryptosystem but Blum and Micali highlight some interesting limitations [12]. Meanwhile, Blum and Micali [12] themselves propose a generator which is related to the problem of efficiently computing the discrete logarithm; Kaliski [60] provides similar work on the use of the discrete logarithm problem over

elliptic curves. Meanwhile Alexi, Chor, Goldreich and Schnorr [1] propose a scheme based on the difficulty of inverting the RSA cryptosystem. Blum, Blum and Shub [11] use the problem of deciding quadratic residuosity as the basis for the security of another keystream generator.

7.4 Other schemes

7.4.1 $1/p$ generator

The $1/p$ generator has a long history and can be traced back to the work of Dickson [30] and Knuth [67]. The pseudo-random sequence is generated by expanding the fraction $1/p$ to some base b where p and b are relatively prime. While the sequence itself has nice statistical features and certain conditions on p and b can ensure a provably large period, it has been shown [11] that this generator is insecure.

7.4.2 Knapsack generator

The security of this generator is based on what is typically called a 'hard' problem and might therefore be more consistently presented in Section 7.3. However it is also a shift-register based scheme and this makes its classification somewhat problematical.

The problem on which the generator is based, is called the knapsack problem because an analogy is often drawn between solving this problem and packing a knapsack with different sized items so that the knapsack is filled exactly. The mathematical exposition of this problem is to find some subset of a large set of numbers such that the sum of the subset equals a specified chosen target value.

In the knapsack generator [114] a set of *weights* are chosen as part of the key. The state of the register at some time instance is combined with the set of weights to give a set of integers. These are then added together using conventional integer arithmetic. Finally bits are chosen from this sum and output as part of the keystream.

The sequences produced have good period and linear complexity properties. However, it seems that the bad name acquired by other specific knapsack-based systems during the early days of public-key cryptography [124, 14] makes many people wary of any knapsack-based system. There do not appear to be, however, any results in the literature on the successful cryptanalysis of this generator.

7.4.3 PKZIP

PKZIP is a widely-used compression function that has an option allowing stream cipher encryption with a variable-length key. This cipher, however, is not secure and Biham and Kocher [8] have described an attack which will find the internal representation of the key in less than one day with a few hundred bytes of known plaintext.

7.5 Final examples

We briefly present yet more alternative approaches.

7.5.1 Randomized ciphers

We have seen cryptographers attempt to prove security against an unlimited adversary or an adversary who is unable to efficiently solve a 'hard' problem. An interesting new direction is provided by techniques which attempt to ensure that the amount of work physically required for successful cryptanalysis is too demanding. A class of stream ciphers designed with this goal in mind, have been labelled *randomized stream ciphers* by Rueppel [126]. Two schemes require massive computation or communication overheads for the legitimate users and can safely be considered impractical but the third scheme we mention is practical assuming that there is some vast public source of random bits.

The two less practical ciphers are as follows. Diffie's randomized stream cipher is described by Rueppel [126]. The plaintext is encrypted using one of 2^n randomly generated keystreams which are all sent along with the ciphertext over the communication channel. The key specifies which of the 2^n sequences the legitimate receiver should pick to use for decryption, giving a considerable advantage over any opponent. Clearly this scheme requires a considerable communication overhead.

Meanwhile Massey and Ingemarsson [78] have presented the Rip van Winkle cipher, so-called because as Massey has said

> One can easily guarantee that the enemy cryptanalyst will need thousands of years to break the cipher, if one is willing to wait millions of years to read the plaintext.

In a third scheme, Maurer considers an information-theoretic approach to the abilities of an adversary when the adversary is computationally limited [82]. Note the contrast between this concept and that of Shannon's information theory [125] where the computational power of the adversary is

assumed to be unlimited. Maurer's scheme relies on the public availability of a vast amount of random information; an example of such a source might be a satellite which continually beams randomly generated data back to earth.

While randomized ciphers are theoretically interesting, it seems that only Maurer's proposal can be viewed as being near to practical.

7.5.2 Cellular automata

Proposed by Wolfram [134] the cellular automata scheme provides a technique for generating sequences with large periods and good statistical properties. The scheme also marks a departure from shift register based schemes and as a consequence does not lend itself to the ready analysis applicable to shift register schemes.

The generator consists of n cells that are arranged in a ring. Each cell is updated at a given time instance according to some simple but non-linear rule defined in terms of adjacent cells. The sequence of values of one chosen cell defines the keystream sequence.

While the lack of a convenient framework for analysis makes cryptanalysis that much harder, it also hinders attempts to assess such basic properties of the system as the period. This is particularly the case when the theoretically interesting model of an infinite array of cells is replaced by the practical realization described above. Meier and Staffelbach [84] have analyzed this proposal and shown that the parameters originally proposed for a practical implementation do not offer adequate security. Daemen [26] has proposed another cipher based on cellular automata that is resistant to the attack of Meier and Staffelbach.

8 Conclusions

While there is no single algorithm which acts as a focus for cryptanalysis in the field of stream ciphers, the impression left by many reviews of stream cipher techniques is that an overwhelming interest has been paid to shift-register based schemes. This report is clearly no exception. Though a huge variety of schemes and different theoretical techniques are available, the reality is that the open literature is dominated with shift register based results. As we have seen, there is a close interplay between shift registers and the techniques of linear algebra and this provides much of the emphasis of research interest.

Despite the wealth of results on both the design and cryptanalysis of shift register based schemes there are numerous other approaches, each with

advantages and disadvantages. In the future we might expect to see some of the alternative approaches to stream cipher design, such as those provided by RC4 and SEAL, becoming extremely popular.

References

[1] W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr. RSA and Rabin functions: certain parts are as hard as the whole. *SIAM Journal on Computing*, 17(2):194–209, April 1988.

[2] R. Anderson. Searching for the optimum correlation attack. In *Proceedings of Leuven Algorithms Workshop, December 1994*, Springer-Verlag, Berlin. To appear.

[3] U. Baum and S. Blackburn. Clock-controlled pseudorandom generators on finite groups. In *Proceedings of Leuven Algorithms Workshop, December 1994*, Springer-Verlag, Berlin. To appear.

[4] H. Beker and F. Piper. *Cipher Systems.* Van Nostrand, London, 1982.

[5] T. Beth and Zong-duo Dai. On the complexity of pseudo-random sequences — or: If you can describe a sequence it can't be random. In J.J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology — Eurocrypt '89*, pages 533–543, Springer-Verlag, Berlin, 1990.

[6] T. Beth and F. Piper. The stop-and-go generator. In T. Beth, N. Cot, and I. Ingemarsson, editors, *Advances in Cryptology — Eurocrypt '84*, pages 88–92, Springer-Verlag, Berlin, 1984.

[7] E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard.* Springer-Verlag, New York, 1993.

[8] E. Biham and P. Kocher. A known plaintext attack on the PKZIP encryption. In *Proceedings of Leuven Algorithms Workshop, December 1994*, Springer-Verlag, Berlin. To appear.

[9] S.R. Blackburn. A generalisation of the discrete Fourier transform: an algorithm to determine the minimum polynomial of a periodic sequence. September 1993. Preprint.

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

[11] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

[12] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–863, 1984.

[13] J. Boyar (Plumstead). Inferring sequences produced by a linear congruential generator missing low-order bits. *Journal of Cryptology*, 1(3):177–184, 1989.

[14] E.F. Brickell. Breaking iterated knapsacks. In G.R. Blakley and D. Chaum, editors, *Advances in Cryptology — Crypto '84*, pages 342–358, Springer-Verlag, New York, 1985.

[15] N.G. de Bruijn. A combinatorial problem. *Nederl. Akad. Wetensch. Proc.*, 49:758–764, 1946.

[16] L. Brynielsson. On the linear complexity of combined shift register sequences. In F. Pichler, editor, *Advances in Cryptology — Eurocrypt '85*, pages 156–166, Springer-Verlag, Berlin, 1986.

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

[18] G.J. Chaitin. On the length of programs for computing finite binary sequences. *J. ACM*, 13(4):547–569, October 1966.

[19] A.H. Chan. On quadratic m-sequences. In R. Anderson, editor, *Fast Software Encryption — Cambridge Security Workshop*, pages 166–173, Springer-Verlag, Berlin, 1994.

[20] A.H. Chan and R.A. Games. On the linear span of binary sequences obtained from geometric sequences. In A.M. Odlyzko, editor, *Advances in Cryptology — Crypto '86*, pages 405–417, Springer-Verlag, New York, 1987.

[21] A.H. Chan and R.A. Games. On the quadratic spans of periodic sequences. In G. Brassard, editor, *Advances in Cryptology — Crypto '89*, pages 82–89, Springer-Verlag, New York, 1990.

[22] U. Cheng. *Properties of Sequences.* PhD thesis, University of Southern California, 1981.

[23] V. Chepyzhov and B. Smeets. On a fast correlation attack on certain stream ciphers. In D.W. Davies, editor, *Advances in Cryptology — Eurocrypt '91*, pages 176–185, Springer-Verlag, Berlin, 1991.

[24] D. Coppersmith, H. Krawczyk, and Y. Mansour. The shrinking generator. In D.R. Stinson, editor, *Advances in Cryptology — Crypto '93*, pages 22–39, Springer-Verlag, New York, 1994.

[25] J. Daemen, R. Govaerts, and J. Vandewalle. Cryptanalysis of MUX-LFSR based scramblers. In *State and Progress in the Research of Cryptography, 1993*, pages 55–61, 1993.

[26] J. Daemen. *Cipher and Hash Function Design.* PhD thesis, Katholieke Universiteit Leuven, 1995.

[27] Zong-duo Dai. Binary sequences derived from *ML*-sequences over rings. 1986. Preprint.

[28] Zong-duo Dai. Proof of Rueppel's linear complexity conjecture. *IEEE Transactions on Information Theory*, IT-32:440–443, 1986.

[29] D.W. Davies and W.L. Price. *Security for Computer Networks: An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer.* John Wiley & Sons, New York, 1984.

[30] L. Dickson. *History of the Theory of Numbers.* Chelsea Pub. Co., London, 1919.

[31] Specification of the Systems of the MAC/Packet Family. *EBU Technical Document 3258-E*, October 1986.

[32] E.D. Erdmann. *Empirical Tests of Binary Keystreams.* Master's thesis, University of London, 1992.

[33] P. Fahn. *Answers to Frequently Asked Questions About Today's Cryptography.* RSA Laboratories, September 1993. Version 2.0.

[34] R. Forré. The strict avalanche criterion: spectral properties of Boolean functions and an extended definition. In S. Goldwasser, editor, *Advances in Cryptology — Crypto '88*, pages 450–468, Springer-Verlag, New York, 1990.

[35] H. Fredricksen. A survey of full length nonlinear shift register cycle algorithms. *SIAM Journal on Applied Mathematics*, 24(2):195–221, 1982.

[36] A.M. Frieze, J. Hastad, R. Kannan, J.C. Lagarias, and A. Shamir. Reconstructing truncated integer variables satisfying linear congruences. *SIAM Journal on Computing*, 17(2):262–280, April 1988.

[37] A.M. Frieze, R. Kannan, and J.C. Lagarias. Linear congruential generators do not produce random sequences. *IEEE Symposium on Foundations of Computer Science*, 480–484, 1984.

[38] R.A. Games. There are no de Bruijn sequences of span n with complexity $2^{n-1} + n + 1$. *Journal of Combinatorial Theory*, Series A, 34:248–251, 1983.

[39] R.A. Games and A.H. Chan. A fast algorithm for determining the complexity of a binary sequence with period 2^n. *IEEE Transactions on Information Theory*, IT-29:144–146, 1983.

[40] R.A. Games, A.H. Chan, and E.L. Key. On the complexities of de Bruijn sequences. *Journal of Combinatorial Theory*, Series A, 33:233–246, 1982.

[41] P.R. Geffe. How to protect data with ciphers that are really hard to break. *Electronics*, 99–101, January 1973.

[42] J. Golić. Correlation via linear sequential circuit approximation of combiners with memory. In R.A. Rueppel, editor, *Advances in Cryptology — Eurocrypt '92*, pages 113–123, Springer-Verlag, Berlin, 1993.

[43] J. Golić. Linear cryptanalysis of stream ciphers. In *Proceedings of Leuven Algorithms Workshop, December 1994*, Springer-Verlag, Berlin. To appear.

[44] J. Golić. Intrinsic statistical weakness of keystream generators. In J. Pieprzyk and R. Safavi-Naini, editors, *Advances in Cryptology — Asiacrypt '94*, pages 91–103, Springer-Verlag, Berlin, 1995.

[45] J. Golić. Towards fast correlation attacks on irregularly clocked shift registers. In L.C. Guillou and J.J. Quisquater, editors, *Advances in Cryptology — Eurocrypt '95*, pages 248–262, Springer-Verlag, Berlin, 1995.

[46] J. Golić and L. O'Conner. Embedding and probabilistic correlation attacks on clock-controlled shift registers. In *Advances in Cryptology — Eurocrypt '94*, Springer-Verlag, Berlin. To appear.

[47] J. Golić and M.V. Zivgović. On the linear complexity of nonuniformly decimated *pn*-sequences. *IEEE Transactions on Information Theory*, IT-34:1077–1079, 1988.

[48] D. Gollmann. Correlation analysis of cascaded sequences. December 1986. Talk presented at 1st IMA Conference on Cryptography and Coding.

[49] D. Gollmann. Pseudo-random properties of cascade connections of clock controlled shift registers. In T. Beth, N. Cot, and I. Ingemarsson, editors, *Advances in Cryptology — Eurocrypt '84*, pages 93–98, Springer-Verlag, Berlin, 1985.

[50] D. Gollmann and W.G. Chambers. Lock-in effect in cascades of clock-controlled shift-registers. In C.G. Günther, editor, *Advances in Cryptology — Eurocrypt '88*, pages 331–343, Springer-Verlag, Berlin, 1988.

[51] G. Gollmann and W.G. Chambers. Clock-controlled shift registers: a review. *IEEE Journal on Selected Areas in Communications*, 7(4):525–533, May 1989.

[52] S.W. Golomb. *Shift Register Sequences.* Holden-Day, San Francisco, 1967.

[53] M. Goresky and A. Klapper. Feedback registers based on ramified extensions of the 2-adic numbers. In *Advances in Cryptology — Eurocrypt '94*, Springer-Verlag, Berlin. To appear.

[54] R. Gottfert and H. Niederreiter. A general lower bound for the linear complexity of the product of shift-register sequences. In *Advances in Cryptology — Eurocrypt '94*, Springer-Verlag, Berlin. To appear.

[55] C.G. Günther. Alternating step generators controlled by de Bruijn sequences. In D. Chaum and W.L. Price, editors, *Advances in Cryptology — Eurocrypt '87*, pages 5–14, Springer-Verlag, Berlin, 1988.

[56] J. Hastad and A. Shamir. The cryptographic security of truncated linearly related variables. In *Proceedings of the 17th ACM Symposium on Theory of Computing*, pages 356–362, 1985.

[57] F.H. Hinsley and A. Stripp, editors. *Codebreakers: The Inside Story of Bletchley Park.* Oxford University Press, 1993.

[58] C.J.A. Jansen. *Investigations on Nonlinear Streamcipher Systems: Construction and Evaluation Methods.* PhD thesis, Technical University of Delft, 1989.

[59] S.M. Jennings. *A Special Class of Binary Sequences.* PhD thesis, University of London, 1980.

[60] B.S. Kaliski Jr. *A pseudo random bit generator based on elliptic logarithms.* Master's thesis, Massachusetts Institute of Technology, 1987.

[61] E.L. Key. An analysis of the structure and complexity of nonlinear binary sequence generators. *IEEE Transactions on Information Theory*, IT-22(6):732–736, 1976.

[62] L.H. Khachaturian. The lower bound of the quadratic spans of de Bruijn sequences. *Designs, Codes and Cryptography*, 3:29–32, 1993.

[63] A. Klapper. The vulnerability of geometric sequences based on fields of odd characteristic. *Journal of Cryptology*, 7(1):33–52, 1994.

[64] A. Klapper and M. Goresky. 2-adic shift registers. In R. Anderson, editor, *Fast Software Encryption — Cambridge Security Workshop*, pages 174–178, Springer-Verlag, Berlin, 1994.

[65] A. Klapper. Feedback with carry shift registers over finite fields. In *Proceedings of Leuven Algorithms Workshop, December 1994*, Springer-Verlag, Berlin. To appear.

[66] A. Klapper and M. Goresky. Large period nearly de Bruijn FCSR sequences. In L.C. Guillou and J.J. Quisquater, editors, *Advances in Cryptology — Eurocrypt '95*, pages 248–262, Springer-Verlag, Berlin, 1995.

[67] D.E. Knuth. *The Art of Computer Programming.* Volume 2, Addison-Wesley, Reading, Mass., 2nd edition, 1981.

[68] D.E. Knuth. *Deciphering a Linear Congruential Encryption.* Technical Report 024800, Stanford University, 1980.

[69] N. Koblitz. *A Course in Number Theory and Cryptography.* Springer-Verlag, New York, 1987.

[70] A.N. Kolmogorov. Three approaches to the definition of the concept 'quantity of information'. *Problemy Peredachi Informatsii*, 1:3–11, 1965. In Russian.

[71] H. Krawczyk. How to predict congruential generators. In G. Brassard, editor, *Advances in Cryptology — Crypto '89*, pages 138–153, Springer-Verlag, New York, 1990.

[72] J.C. Lagarias and J.A. Reeds. Unique extrapolation of polynomial recurrences. *SIAM Journal on Computing*, 17(2):342–362, April 1988.

[73] S. Lloyd. Counting binary functions with certain cryptographic properties. *Journal of Cryptology*, 5(2):107–131, 1992.

[74] G. Marsaglia. Random numbers fall mainly in the planes. *Proc. N.A.S.*, 61:25–28, 1968.

[75] P. Martin-Löf. The definition of random sequences. *Inform. Contr.*, 9:602–619, 1966.

[76] J. Massey and R.A. Rueppel. Linear ciphers and random sequence generators with multiple clocks. In T. Beth, N. Cot, and I. Ingemarsson, editors, *Advances in Cryptology — Eurocrypt '84*, pages 74–87, Springer-Verlag, Berlin, 1984.

[77] J.L. Massey. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, IT-15:122–127, 1969.

[78] J.L. Massey and I. Ingemarsson. The Rip van Winkle cipher - a simple and provably computationally secure cipher with a finite key. In *Abstracts of papers, IEEE Int. Symp. Inform. Theory*, Brighton, UK, June 1985.

[79] J.L. Massey and S. Serconek. A Fourier transform approach to the linear complexity of nonlinearly filtered sequences. In Y. Desmedt, editor, *Advances in Cryptology — Crypto '94*, pages 332–340, Springer-Verlag, New York, 1994.

[80] M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology — Eurocrypt '93*, pages 386–397, Springer-Verlag, Berlin, 1994.

[81] U.M. Maurer. New approaches to the design of self-synchronizing stream ciphers. In D.W. Davies, editor, *Advances in Cryptology — Eurocrypt '91*, pages 458–471, Springer-Verlag, Berlin, 1991.

[82] U.M. Maurer. A provable-secure strongly-randomized cipher. In I.B. Damgård, editor, *Advances in Cryptology — Eurocrypt '90*, pages 361–373, Springer-Verlag, Berlin, 1991.

[83] U.M. Maurer. A universal statistical test for random bit generators. In A.J. Menezes and S.A. Vanstone, editors, *Advances in Cryptology — Crypto '90*, pages 409–420, Springer-Verlag, New York, 1991.

[84] W. Meier and O. Staffelbach. Analysis of pseudo random sequences generated by cellular automata. In D.W. Davies, editor, *Advances in Cryptology — Eurocrypt '91*, pages 186–199, Springer-Verlag, Berlin, 1992.

[85] W. Meier and O. Staffelbach. Correlation properties of combiners with memory in stream ciphers. In I.B. Damgård, editor, *Advances in Cryptology — Eurocrypt '90*, pages 549–562, Springer-Verlag, Berlin, 1991.

[86] W. Meier and O. Staffelbach. Correlation properties of combiners with memory in stream ciphers. *Journal of Cryptology*, 5(1):67–86, 1992.

[87] W. Meier and O. Staffelbach. Fast correlation attacks on certain stream ciphers. *Journal of Cryptology*, 1(3):159–176, 1989.

[88] W. Meier and O. Staffelbach. The self-shrinking generator. In *Advances in Cryptology — Eurocrypt '94*, Springer-Verlag. To appear.

[89] R. Menicocci. Short Gollmann cascade generators may be insecure. In *Proceedings of the 4th IMA Conference on Cryptography and Coding*. To appear.

[90] M.J. Mihaljević and J. Golić. A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence. In J. Seberry and J. Pieprzyk, editors, *Advances in Cryptology — Auscrypt '90*, pages 165–175, Springer Verlag, Berlin, 1990.

[91] M.J. Mihaljević. A correlation attack on the binary sequence generators with time-varying output function. In J. Pieprzyk and R. Safavi-Naini, editors, *Advances in Cryptology — Asiacrypt '94*, pages 67–79, Springer-Verlag, Berlin, 1995.

[92] S. Mund. Ziv-Lempel complexity for periodic sequences and its cryptographic application. In D.W. Davies, editor, *Advances in Cryptology — Eurocrypt '91*, pages 114–126, Springer-Verlag, Berlin, 1992.

[93] National Institute of Standards and Technology (NIST). *FIPS Publication 180: Secure Hash Standard (SHS).* May 11, 1993.

[94] National Institute of Standards and Technology (NIST). *FIPS Publication 46-2: Data Encryption Standard*. December 30, 1993.

[95] National Institute of Standards and Technology (NIST). *FIPS Publication 81: DES Modes of Operation*. December 2, 1980. Originally issued by National Bureau of Standards.

[96] H. Niederreiter. The linear complexity profile and the jump complexity of keystream sequences. In I.B. Damgård, editor, *Advances in Cryptology — Eurocrypt '90*, pages 174–188, Springer-Verlag, Berlin, 1991.

[97] L. O'Conner and T. Snider. Suffix trees and string complexity. In R.A. Rueppel, editor, *Advances in Cryptology — Eurocrypt '92*, pages 138–152, Springer-Verlag, Berlin, 1993.

[98] V.S. Pless. Encryption schemes for computer confidentiality. *IEEE Transactions on Computers*, C-26:1133–1136, Nov. 1977.

[99] J. Plumstead (Boyar). Inferring a sequence generated by a linear congruence. In *Proceedings of 23rd IEEE Symposium on Foundations of Computer Science*, pages 153–159, 1982.

[100] B. Preneel, W. Van Leekwijck, L. Van Linden, R. Govaerts, and J. Vandewalle. Propagation characteristics of Boolean functions. In I.B. Damgård, editor, *Advances in Cryptology — Eurocrypt '90*, pages 161–173, Springer-Verlag, Berlin, 1991.

[101] N. Proctor. A self-synchronizing cascaded cipher system with dynamic control of error-propagation. In G.R. Blakley and D. Chaum, editors, *Advances in Cryptology — Crypto '84*, pages 174–190, Springer-Verlag, New York, 1985.

[102] J.A. Reeds. 'Cracking' a random number generator. *Cryptologia*, 1, January 1977.

[103] J.A. Reeds and N.J.A. Sloane. Shift register synthesis (modulo m). *SIAM Journal on Computing*, 14(3):505–513, 1985.

[104] R.L. Rivest. *The RC4 Encryption Algorithm*. RSA Data Security, Inc., March 12, 1992.

[105] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[106] M.J.B. Robshaw. *Block Ciphers.* Technical Report TR - 601, RSA Laboratories, revised July 1995.

[107] M.J.B. Robshaw. *On Binary Sequences with Certain Properties.* PhD thesis, University of London, 1992.

[108] M.J.B. Robshaw. On evaluating the linear complexity of a sequence of least period 2^n. *Designs, Codes and Cryptography*, 4:263–269, 1994.

[109] M.J.B. Robshaw. *Security of RC4.* Technical Report TR - 401, RSA Laboratories. To appear.

[110] P. Rogaway and D. Coppersmith. A software-optimized encryption algorithm. In R. Anderson, editor, *Fast Software Encryption — Cambridge Security Workshop*, pages 56–63, Springer-Verlag, Berlin, 1994.

[111] C.A. Ronce. *Feedback Shift Registers.* Volume 169 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 1984.

[112] O.S. Rothaus. On bent functions. *Journal of Combinatorial Theory*, Series A, 20:300–305, 1976.

[113] R.A. Rueppel. Correlation immunity and the summation combiner. In H.C. Williams, editor, *Advances in Cryptology — Crypto '85*, pages 260–272, Springer-Verlag, New York, 1986.

[114] R.A. Rueppel. *New Approaches to Stream Ciphers.* PhD thesis, Swiss Federal Institute of Technology, Zurich, 1984.

[115] R.A. Rueppel. When shift registers clock themselves. In D. Chaum and W.L. Price, editors, *Advances in Cryptology — Eurocrypt '87*, pages 53–64, Springer-Verlag, Berlin, 1988.

[116] R.A. Rueppel and O.J. Staffelbach. Products of linear recurring sequences with maximum complexity. *IEEE Transactions on Information Theory*, IT-33(1):124–131, 1987.

[117] A. Fúster-Sabater and P. Caballero-Gil. On the linear complexity of nonlinearly filtered PN-sequences. In J. Pieprzyk and R. Safavi-Naini, editors, *Advances in Cryptology — Asiacrypt '94*, pages 80–90, Springer-Verlag, Berlin, 1995.

[118] J. Seberry, X.M. Zhang, and Y. Zheng. Nonlinearly balanced Boolean functions and their propogation characteristics. In D.R. Stinson, editor, *Advances in Cryptology — Crypto '93*, pages 49–60, Springer-Verlag, New York, 1994.

[119] T. Seigenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, IT-30(5):776–779, Sept. 1984.

[120] T. Seigenthaler. Cryptanalyst's representation of nonlinearity filtered *ml*-sequences. In F. Pichler, editor, *Advances in Cryptology — Eurocrypt '85*, pages 103–110, Springer-Verlag, Berlin, 1986.

[121] T. Seigenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, C-34(1):81–85, Jan. 1985.

[122] E.S. Selmer. *Linear Recurrence Relations over Finite Fields.* University of Bergen, Norway, 1966.

[123] A. Shamir. On the generation of cryptographically strong pseudorandom sequences. In *Proc. 8th Int. Colloquium on Automata, Languages, and Programming*, Springer-Verlag, 1981.

[124] A. Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. *IEEE Transactions on Information Theory*, IT-30(5):699–704, Sept. 1984.

[125] C.E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:657–715, 1949.

[126] G.J. Simmons, editor. *Contemporary Cryptology, The Science of Information Integrity.* IEEE, New York, 1992.

[127] B. Smeets. A note on sequences generated by clock-controlled shift registers. In F. Pichler, editor, *Advances in Cryptology — Eurocrypt '85*, pages 40–42, Springer-Verlag, Berlin, 1986.

[128] O. Staffelbach and W. Meier. Cryptographic significance of the carry for ciphers based on integer addition. In A.J. Menezes and S.A. Vanstone, editors, *Advances in Cryptology — Crypto '90*, pages 601–615, Springer-Verlag, New York, 1990.

[129] S.A. Tretter. Properties of PN^2 sequences. *IEEE Transactions on Information Theory*, IT-20:295–297, March 1974.

[130] G.S. Vernam. Cipher printing telegraph systems for secret wire and radio telegraphic communications. *J. Am. Inst. Elec. Eng.*, 55:109–115, 1926.

[131] R. Vogel. On the linear complexity of cascaded sequences. In T. Beth, N. Cot, and I. Ingemarsson, editors, *Advances in Cryptology — Eurocrypt '84*, pages 99–109, Springer-Verlag, Berlin, 1984.

[132] M.Z. Wang and J.L. Massey. The characteristics of all binary sequences with perfect linear complexity profiles. May 20–22 1986. Presented at Eurocrypt'86.

[133] M. Ward. The arithmetic theory of linear recurring series. *Trans. A.M.S.*, 35:600–628, 1933.

[134] S. Wolfram. Cryptography with cellular automata. In H.C. Williams, editor, *Advances in Cryptology — Crypto '85*, pages 429–432, Springer-Verlag, New York, 1986.

[135] A.C. Yao. Theory and applications of trapdoor functions. In *Proc. 25th IEEE Symp. Foundations Comput. Sci.*, New York, 1982.

[136] K. Zeng, C.H. Yang, and T.R.N. Rao. An improved linear syndrome algorithm in cryptanalysis with applications. In A.J. Menezes and S.A. Vanstone, editors, *Advances in Cryptology — Crypto '90*, pages 34–47, Springer-Verlag, New York, 1990.

[137] K. Zeng, C.H. Yang, and T.R.N. Rao. On the linear consistency test in cryptanalysis with applications. In G. Brassard, editor, *Advances in Cryptology — Crypto '89*, pages 167–174, Springer-Verlag, New York, 1990.

[138] K. Zeng, C.H. Yang, D.Y. Wei, and T.R.N. Rao. Pseudorandom bit generators in stream-cipher cryptography. *Computer*, 8–17, February 1991.

[139] N. Zierler. Linear recurring sequences. *J. Soc. Indust. Appl. Math.*, 7(1):31–48, 1959.

[140] N. Zierler and W.H. Mills. Products of linear recurring sequences. *Journal of Algebra*, 27:147–157, 1973.

[141] J. Ziv and A. Lempel. On the complexity of finite sequences. *IEEE Trans. Information Theory*, 22:75–81, 1976.

[142] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Information Theory*, 23(3):337–343, 1977.