Computer Science Tripos part 2 – Security

Lecture 6 – Stream Ciphers

Ross J. Anderson

1 Introduction

In the next few lectures, we are going to look at cryptography. This is Greek for 'secret writing'; but secrecy will not be our only concern. As we already saw, there are applications (such as frequency agile radar) where we simply need to generate an unpredictable stream of bits; and there are yet others where we want to bind data items together (such as in electronic commerce, where we do not want an opponent to be able to cut and paste the payees, dates and amounts from different instruments). So we need to understand the properties of ciphers in more detail, and historically stream ciphers came first.

2 Elementary Ciphers

In *'The Twelve Caesars'*, Suetonius tells us that Julius Caesar enciphered his dispatches by writing 'D' for 'A', 'E' for 'B' and so on. If we number the letters so that 'A' = 0, 'B' = 1, through to 'Z' = 25, then we can express his encryption algorithm as *ciphertext* = *plaintext* + *'D'* (mod 26). When Augustus Caesar ascended the throne, he changed the imperial cipher system to *ciphertext* = *plaintext* + *'C'* (mod 26). In modern terminology, we would say that he changed the key from *'D'* to *'C'*.

The Arabs generalised this idea to the monoalphabetic substitution, in which a keyword is typically used to permute the cipher alphabet. We will write the plaintext in lower case letters, and the ciphertext in upper case:

```
abcdefghijklmnopqrstuvwxyz
SECURITYABDFGHJKLMNOPQVWXZ
```

`OYAN RWSGKFR AN AH RHTFANY MSOYRM OYSH SMSEAC NCMAKO`; but breaking ciphers of this kind is a straightforward pencil and paper puzzle, which you may have done in primary school. The key is that some letters, and combinations of letters, are much more common than others; in English the most common letters are e,t,a,i,o,n,s,h,r,d,l,u in that order. Artificial intelligence researchers have shown some interest in writing programs to solve monoalphabetic substitutions; using letter and digram (letter pair) frequencies alone, they typically succeed with about 600 letters of ciphertext, while smarter strategies such as guessing probable words can cut this to about 150 letters.

3 The Vigenère

Another generalisation of the Caesar cipher, and one which was widely used over the past four hundred years, is commonly ascribed to the Frenchman Blaise de Vigenère, a diplomat who served the Duke of Nevers and King Charles IX. It works by adding a key repeatedly into the plaintext using the above convention ('A' = 0, 'B' = 1, ..., 'Z' = 25, with addition carried out modulo 26):

Plain	`nowisthetimeforallgoodmen`
Key	`typetypetypetypetypetypet`
Cipher	`GMLMLRWIMGBIYMBEEJVSHBBIG`

In fact, this attribution does not really do Vigenère justice. While on assignment at the Vatican, he picked up the cryptologic lore of the day, and wrote it up in a book called *Traicté des Chiffres* which he published in 1585 and which describes a number of systems, including some which can be made somewhat more secure than the above. However, the name has stuck; this simple polyalphabetic cipher was widely used, and was widely believed to be secure until a solution was published in the nineteenth century. The addition operation has been implemented in various ways, including printed addition tables, slide rules and rotating discs.

A number of people appear to have worked out how to solve polyalphabetic ciphers, from the notorious womaniser Casanova to computing pioneer Charles Babbage. However the first published solution was in *'Die Geheimschriften und die Dechiffrier-Kunst'*, a book written in 1863 by Friedrich Kasiski, a Prussian infantry officer. He noticed that given a long enough piece of ciphertext, repeated patterns will appear at multiples of the keyword length:

Plain	`tobeornottobethatisthequestion`
Key	`runrunrunrunrunrunrunrunrunrun`
Cipher	`KIOVIEEIGKIOVNURNVJNUVKHVMGZIA`

Here, for example, we see '`KIOV`' repeated after nine letters, and '`NU`' after six. So we would guess a keyword of three letters. It follows that ciphertext letters one, four, seven and so on all enciphered under the same keyletter; so we can use frequency analysis techniques to guess the most likely values of this letter, and then repeat the process for the second and third letters of the key.

During world war one, a more sophisticated technique was developed by US cryptologist William Friedmann. This is the index of coincidence, defined as the probability that two letters in a text are the same; it is almost equal to the sum of the squares of the letter probabilities. For English, the index of coincidence (often written I_c) is about 0.065; so that when we choose two letters at random from a book (or a monoalphabtic cipher) we expect them to be the same about 6.5% of the time. For a purely random text, the probability is clearly 1/26 or about 0.038. It turns out that for a Vigenère cipher, the index of coincidence depends on the length of the keyword; and so we can estimate this length from the ciphertext letter frequencies (see exercise below).

You might not believe how many programmers re-invent variants of this simple cipher when confronted with the need to protect a file.

4 Kerkhoff's Principles

The first exposition of the engineering acpects of cryptology came in 1883 with the publication of *'La Cryptographie Militaire'* by Auguste Kerkhoffs, a Dutch philologist, who put forward six principles for evaluating the usefulness of a cipher system:

1. the system should be hard to break in practice, even if it is breakable in theory;
2. it should not be compromised if the opponent learns the technique being used. In other words, the security should reside in the choice of key;
3. the key should be easy to remember and to change;
4. the cryptograms should be capable of transmission by telegraph;
5. the apparatus or documents should be portable and operable by a single person;
6. the system should not involve a long list of rules or mental strain.

These principles should be self-evident, but there are many cases of cipher system designers breaking them, with often serious consequences. For example, the system used by the Tsar's army in 1914-17 was so complex that under the stress of battle, the signal clerks threw it away and communicated in clear; this cost the battle of Tannenberg and led ultimately to the Russsian revolution. More recently, a bank had to reissue all its customers with ATM cards and PINs, an exercise which cost millions; the reason appears to have been a cryptographic key compromise from which a system level recovery had not been provided for.

5 Playfair

By no means all early cipher systems relied on substitution. Transposition was also used, and one of the best known examples is the Playfair cipher. This was invented in 1854 by Sir Charles Wheatstone (who also invented the telegraph, the concertina and the Wheatstone bridge), and was sold to Palmerston, the

Prime Minister, by his friend Lyon Playfair, the foreign secretary, by demonstrating it on a napkin after dinner.

This cipher uses a 5 by 5 grid, in which we place the alphabet, permuted by the key word, and omitting the letter 'J':

P	A	L	M	E
R	S	T	O	N
B	C	D	F	G
H	I	K	Q	U
V	W	X	Y	Z

The plaintext is first conditioned by replacing 'J' with 'I' wherever it occurs, then dividing it into letter pairs, preventing double letters occurring in a pair by separating them with an 'x', and finally adding a 'z' if necessary to complete the last letter pair. Thus for example, 'Lord Granville's letter' becomes '`lo rd gr an vi lx le sl et te rz`'

It is then enciphered two letters at a time using the following rules:

- if two letters are in the same row or column, they are replaced by the succeeding letters. For example, '`am`' enciphers to '`LE`'
- otherwise the two letters stand at two of the corners of a rectangle in the table, and we replace them with the letters at the other two corners of this rectangle. For example, '`lo`' enciphers to '`MT`'.

We can now encipher our specimen text as follows:

Plain	lo	rd	gr	an	vi	lx	le	sl	et	te	rz
Cipher	MT	TB	BN	ES	WH	TL	MP	TA	LN	NL	NV

Variants of this cipher were used by the British army as a field cipher in world war one, and by the Americans and Germans in world war two. Of course, a modern analyst, equipped with a computer, could simply try all possible key words; and given enough ciphertext, or a few probable words, even a thoroughly mixed table can be reconstructed (see exercise). Nonetheless, Playfair is a substantial improvement on Vigenère.

6 The One Time Pad

All the above ciphers can be broken by an opponent with enough computer power to try all the possible keys. However, as you saw in the lectures on Information Theory, there is one type of cipher whose security is independent of the opponent's computational capacity – the one-time pad. In this, the key sequence is as long as the plaintext, and never repeats. The effect is that given any ciphertext, and any plaintext of the same length, there is a key which decrypts the ciphertext to the plaintext. Regardless of the amount of computation which an opponent can do, he is none the wiser, as all possible plaintexts are just as likely.

In the 1940's, Claude Shannon proved that a cipher has perfect secrecy if and only if there are as many possible keys as possible plaintexts, and every key is equally likely; so the one-time pad is the only kind of system which offers perfect secrecy. He defined the unicity distance of a cipher as the quantity of ciphertext needed in order to uniquely determine the key; thus a cipher with perfect secrecy is one whose message length is less than the unicity distance. For other ciphers, a good approximation to the unicity distance is $H(K)/D$ where $H(K)$ is the log to base 2 of the number of possible keys, and D is the redundancy of the plaintext, which for English is about 3.2 bits per character.

The one-time pad has been widely used by spies. In this application, the plaintext is usually encoded as a series of five-digit groups and then the key is added digit by digit without carry. Back in the days of the Cold War, there were 'number stations' in East Germany which transmitted five figure cipher groups to agents in the west; the agents' pads were printed on inflammable acetate film, so that they could destroy the evidence quickly if need be. A one-time pad is also used to encrypt the 'hot line' between Washington and Moscow; in this case the pad consists of a tape of random bits which are combined with the plaintext using exclusive or to give the ciphertext.

So given that a perfect cipher system exists, why does everyone not use it? Why are people still doing research in cryptography?

The first answer to this is that in many systems it would be quite impractical to ship the quantity of one-time key required. For example, in satellite TV encryption, we could consume megabits of key per second.

The second answer is that we often need more than just secrecy. We may also wish to protect data integrity and provide some assurance that a signal has not been manipulated. In this application, the one-time pad is useless.

Suppose for example that you had intercepted a message from a wartime German agent which you knew started with 'Heil Hitler', and the first ten letters of ciphertext were `DGTYI BWPJA`. This means that the first ten letters of the one-time pad were `wclnb tdefj`:

Plain	`heilhitler`
Key	`wclnbtdefj`
Cipher	`DGTYIBWPJA`

If you wanted to get him into trouble, you could change the ciphertext to `DCYTI BWPJA`:

Cipher	`DCYTIBWPJA`
Key	`wclnbtdefj`
Plain	`hanghitler`

This weakness, known as an 'attack in depth', is quite topical as it can be used against the 'Clipper' phone which the US government tried to establish as a standard for encrypted business communications. The effect of using a simple stream cipher was to impair the evidential value of intercepted conversations.

We will now look at how these two objections can be dealt with. We will first consider how a relatively small key can be expanded into a long keystream; integrity mechanisms will be discussed in later lectures.

7 Keystream Generators

We have already seen two stream ciphers – the one-time pad, which offers perfect secrecy, and the Vigenère, in which the keyword is simply repeated. Ideally, we would like to have a system which gives us something like the security of the former, while retaining something like the short key size of the latter.

The common solution is a keystream generator, also called a pseudorandom number generator. This is an algorithm which we initialise with a key and which then produces a long sequence of random looking bits. These bits should have the property that even if the opponent knows the algorithm, and even if he knows a considerable quantity of keystream, he should not be able to predict any more of it. After all, there are many applications where the opponent can observe billions of keystream bits after the fact – spread spectrum radio and satellite TV scrambling are merely two of the more obvious examples.

This means that we will want our keystream generator to pass the standard statistical tests for randomness. There are many of these, but the set of tests most commonly referred to is due to Golomb:

- there should be about the same number of ones and zeros
- about 2^{-m} of the runs (of all ones or all zeros) should have length m
- the out-of-phase autocorrelation should be a constant, i.e. prob$(s_i = s_{i+m})$ should be the same for all $m \neq 0$.

Other tests which you may come across are the serial test (the transition probabilities between consecutive bits are even) and the poker test (each m-tuple of bits has probability 2^{-m}).

However, good statistical properties are not enough. The random number generator which comes with your favourite compiler is likely to have very good statistical properties, but still be extremely easy for an opponent to predict as it typically uses a linear recurrence relation such as $r_n = ar_{n-1} + b \pmod{n}$ for suitably chosen values of a, b and n (typically $n = 32$). See volume 2 of Knuth for more details.

8 Linear Feedback Shift Registers

Another classic example of a pseudorandom number sequence with good statistical properties is the linear feedback shift register sequence. This is simply a sequence in which each bit is the exclusive-or of the same pattern of the preceding bits; it is so called because it can be easily implemented in hardware using a shift register with a suitable pattern of feedback taps. Its generating relation is also a linear recurrence relation but with $n = 2$ and of the form

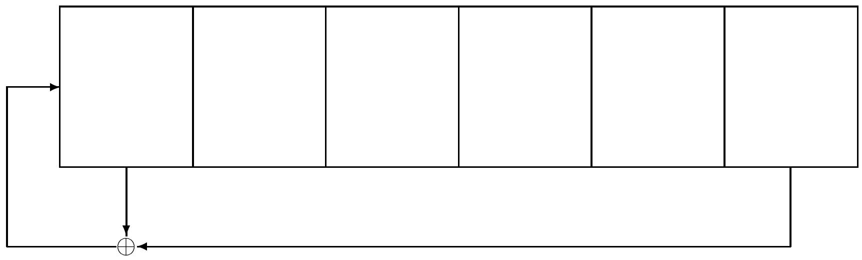
$$s_i = c_1 s_{i-1} \oplus c_2 s_{i-2} \oplus ... \oplus c_n s_{i-n} \tag{1}$$

The properties of this sequence are tied up with those of its generating polynomial, $1+c_1x+c_2x^2+...+c_nx^n$ (here we assume, without loss of generality, that $c_n = 1$). If this polynomial is *primitive* – which is similar to an integer's being prime – then the sequence will have the maximum length $2^n - 1$ (unless it is the trivial sequence of all zeros). Essentially, the k-th bit of the sequence can be got by applying the k-th power of the state transition matrix to the initial state, and the generating polynomial is also the characteristic polynomial of the transition matrix. If the sequence does not factor into subsequences, then the polynomial will not factor either. This is not a proof; for that, see Lidl and Niederreiter's *Finite Fields*.

For example, a shift register of length 6 will have a transition matrix

$$\mathcal{M} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ c_6 & c_5 & c_4 & c_3 & c_2 & c_1 \end{pmatrix}$$

And if its generating polynomial is $x^6 + x + 1$, then it can be implemented in hardware by a six stage shift register and one exclusive-or gate:



Now since $x^6 + x + 1$ is primitive, this circuit will generate two sequences: if its initial fill is 000000, then it will remain stuck at this value, but if started off from any other value, its state will cycle through each of the 63 other 6-tuples of bits exactly once (you should check this as an exercise if you are doubtful). In general a maximum length sequence, or *m-sequence*, with generating polynomial of degree n, will contain every n-tuple of bits exactly once (with the exception of the all-zero n-tuple), its statistical properties are near perfect.

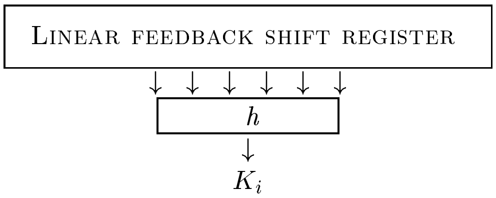
Until the late 1970's, some manufacturers of stream cipher equipment used these sequences as keystreams. However, given any n bits of the sequence, the rest of the sequence can be found by linear algebra (see exercise 4 for the case in which both the initial state and the feedback taps are unknown).

9 Shift Register Based Keystream Generators

Many of the keystream generators in use today are still based on shift registers, but with added nonlinearity. Examples are:

9.1 The nonlinear filter generator

The simplest kind of shift register based keystream generator is the nonlinear filter generator. This is based on a linear feedback shift register, whose initial state is the secret key; some of its state bits are filtered through a nonlinear function to provide the keystream.



Such systems can in principle be attacked by finding a function of the keystream which is correlated with the underlying shift register sequence (see exercises). We then have with the problem of reconstructing this sequence from what appears to be a noisy version of it:

$$\boxed{\text{LFSR}} \longrightarrow \boxed{\text{ADD NOISE}} \longrightarrow K_i$$

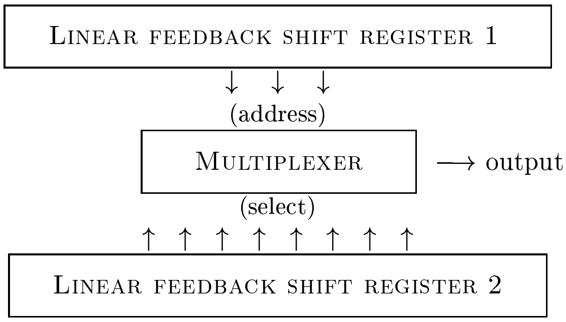
The standard shift register reconstruction algorithm by Meier and Staffelbach starts off with a guess of the shift register state and then changes bits in order to improve the fit with the feedback relations; it converges if the feedback of the shift register is fairly sparse. Newer techniques (developed by David MacKay here in Cambridge) use belief propagation methods to improve on this somewhat.

The nonlinear filter generator is still widely used. The stronger variants of it tend to have multiple shift registers combined in a very complex filter function to provide the keystream.

9.2 The multiplexer generator

This is the keystream generator used by Sky TV and many other satellite TV broadcasters to encrypt their video signals. Each line of the signal is cut and rotated at a point determined by the output of the generator.

In the generator itself, two feedback shift registers are employed, which generate sequences whose lengths are coprime. Some of the bits of register 1 are used as address lines to a multiplexer, which selects the next bit of the keystream from register 2:



It can be shown that this generator produces a very long and highly nonlinear sequence. However, there is a divide and conquer attack on it: guess the contents of register 1 and then check to see whether the observed keystream is consistent with this. If our guess for register 1 was incorrect, then we will usually find ourselves trying to fit a '1' and a '0' from the keystream into the same position in the sequence of register 2, or getting a pattern that is not consistent with the feedback polynomial.

In theory, one could use this attack to decrypt satellite TV with an effort of about 2^{30} for each key. In practice, actual breaks of these systems have involved attacks on the key management mechanism, on the protocol used to revoke customers who have stopped paying their subscriptions, or microprobing of customer smartcards to recover master keys. There is a lot more to building a secure system then just possessing a good keystream generator!

9.3 A5

GSM digital cellular telephones use an algorithm called A5 for encrypting voice traffic over the air link. It consists of three shift registers of lengths 19, 22 and 23, with sparse feedback taps, whose clocking is interlocked: the eleventh bit of each register is examined and if these are all equal then all three registers are clocked. If they are not, then the two registers whose eleventh bits are equal are clocked. The keystream is the exclusive-or of the three registers' least significant bits.

This is an example of a design which resists a naïve divide-and-conquer attack: the best currently known attack on it consists of guessing the state of two of the registers and then working back from the keystream to get the state of the third. It does have short cycles (almost half of starting states end in a cycle with length $2/3(2^{23} - 1)$) but this hardly matters in GSM, where it is used to encrypt two short packets and then re-keyed. In fact, for a given level of security, A5 uses fewer gates (when implemented in custom hardware) than other stream cipher algorithms of comparable strength – an important consideration in battery operated equipment.

Despite the fact that A5 can be broken with some effort, actual surveillance attacks on GSM tend to target the communications between the base stations, which are not protected by A5, or to involve the use of fraudulent base stations or other such trickery. Again, the cryptographic algorithm is not the weak point of the system.

Of course, the keystream generator is not sufficient in itself: we need to authenticate users to prevent billing fraud, to negotiate session keys, and to protect all this traffic on the rest of the network. We may also discuss GSM authentication and attacks on it if there is time.

10 Generators Not Based on Shift Registers

Shift register technology is still widely used in encryption hardware, but an increasing proportion of applications are implemented in software, and in this case a shift register system can be less than ideal — all the bit level manipulation can end up being rather slow on word-oriented cpu's. There have therefore been a number of proposals recently for algorithms based on nonlinear automata which utilise the computer's ability to retain a lot of state.

A widely used generator is RC4, developed by Ron Rivest at MIT and used in SSL, the protocol used to secure electronic commerce transactions in web browsers.

The key to RC4 is a permutation of 0, ..., 255 which we will write $s[0]$, ..., $s[255]$. There are also two counters, i and j, which are initialised to zero. The next byte of keystream, k, is generated by:

$i \leftarrow i+1 \pmod{256}$
$j \leftarrow j+s[i] \pmod{256}$
$\mathrm{swap}(s[i], s[j])$
$t \leftarrow s[i]+s[j] \pmod{256}$
$k \leftarrow s[t]$

There are no practical attacks published on RC4. There are short cycles, minor statistical irregularities and other features that a purist might consider undesirable, but the state space is so vast that none of them give rise to practical attacks.

11 Summary

Many programmers have invented encryption schemes which either add a password repeatedly into a file, or which repeatedly exclusive or the plaintext with the output of the random number generator to get ciphertext. Neither of these is much good: the first method had already been broken in the nineteenth century and the second can be defeated with elementary linear algebra.

We can construct a cipher with perfect secrecy using a one-time tape. The sender and receiver each possess a copy of a tape of truly random bits, and encrypt data by exclusive or'ing it a bit at a time with the plaintext to get ciphertext. Provided that the random tape is never reused, and not compromised in any way, this cipher cannot be broken.

However, in practice, we usually construct stream ciphers using a keystream generator. A good generator has the property that even given a lot of known keystream, an opponent cannot predict any more of it; in particular, a good keystream generator will pass all statistical tests. However, this is not enough, as linear feedback shift registers show.

Typical modern keystream generators are often built using various nonlinear combinations of shift register sequences. Typical attacks on them include correlation attacks, in which we try to infer the state of the underlying shift register, and divide-and-conquer attacks, in which we try to guess the contents of the various shift registers one at a time.

There is one systematic problem with additive stream ciphers – they do not provide message integrity. An opponent can alter the ciphertext so as to change the plaintext in a predictable way, and if he knows the position of some plaintext within the message, then he can alter this text to produce an undetectable forgery. This is called an attack in depth; and it still works if the cipher system is a one-time tape, whose secrecy is provable. Secrecy and authenticity are definitely not the same thing.

12 Further Reading

If you want to learn more about stream ciphers, there are two rather dated books (*'Cipher Systems'* by Beker and Piper and *'The Analysis and Design of Stream Ciphers'* by Rueppel). Recent research can be found in the proceedings of the *Crypto* and *Eurocrypt* conferences, and the workshops on Fast Software Encryption, which are published by Springer in the Lecture Notes in Computer Science series.

13 Problems

1. Are Kerkhoff's principles still relevant to modern computer based cryptography? How should we update them?

2. Show that if a text of n letters is Vigenère enciphered using a keyword of length l, then the expected number of pairs of equal letters is

$$0.065\frac{n(n-l)}{2l} + 0.038\frac{n^2(l-1)}{2l}$$

   Hence or otherwise show that the keyword length is estimated by

$$\frac{0.027n}{(n-1)I_c - 0.038n + 0.065}$$

3. The following Playfair encrypted message was sent on August 2nd, 1943, by an allied agent behind Japanese lines to report the sinking of a boat captained by John F Kennedy, the future president of the USA:

   ```
   KXJEY UREBE ZWEHE WRYTU HEYFS KREHE GOYFI WTTTU OLKSY CAJPO
   BOTEI ZONTX BYBWT GONEY CUZWR GDSON SXBOU YWRHE BAAHY USEDQ
   ```

   Solve this, given the hint that the sunken boat was PT Boat 119 and that vessel identities usually preceded naval signals. Note also that in this variant of Playfair, double letters were written in clear rather than being split up with an 'x'.

   On August 7th, a following message was sent, which contained the number and location of the survivors. Given that one of the nearby islands was called Gross Island, solve this message.

```
XELWA OHWUW YZMWI HOMNE OBTFW MSSPI AJLUO EAONG OOFCM FEXTT
CWCFZ YIPTF EOBHM WEMOC SAWCZ SNYNW MGXEL HEZCU FNZYL NSBTB
DANFK OPEWM SSHBK GCWFV EKMUE
```

Had the Japanese had been able to solve these signals, the modern history of the USA might have been quite different!

4. The Soviet ZS-23 antiaircraft gun's radar used a lag sequence to make jamming harder (a '1' bit caused the next pulse to be delayed). The sequence was generated using a 16 bit linear feedback shift register, whose key was not just the initial state but also the feedback polynomial itself. Show that 31 bits of the sequence are enough to predict it (hint: arrange these bits in a suitable matrix).

5. A keystream generator is constructed using a linear feedback shift register of length 32 and taps (32,31,30,10) with a nonlinear filter. If we write the shift register state as s_i, then the keystream is given by

$$s_1 \oplus s_2 \oplus (s_1 \oplus s_3)(s_2 \oplus s_4 \oplus s_5) \oplus (s_1 \oplus s_4)(s_2 \oplus s_3)s_5$$

   Show that this generator is insecure in that given a few hundred bits of keystream, the shift register state can be recovered much more quickly than by exhaustive search (hint: consider which 9-tuples of shift register bits give rise to which 5-tuples of keystream).

6. Find out how the random number generator in your favourite compiler works. How would you decipher a file which had been encrypted by repeatedly calling this generator and exclusive-or'ing the least significant byte of its output with the next plaintext byte? Would your attack be feasible if all you knew about the plaintext was that it was ASCII encoded?

7. Reverse engineer the file encryption or other password protection mechanisms on any personal computer software products you own. Are any of them any good?

8. How do Kerkhoff's principles apply to steganography? Which of the steganographic systems discussed in lecture 5 obey them, and which do not?