Lecture 4:
Stream ciphers
Block cipher confidentiality modes of operation
Stallings: Ch 6, Ch 3
Stream ciphers

- Stream ciphers are generally faster than block ciphers, especially when implemented in hardware.
- Stream ciphers have less hardware complexity.
- Stream ciphers can be adapted to process the plaintext bit by bit, or word by word, while block ciphers require buffering to accumulate the full plaintext block.
- Synchronous stream ciphers have no error propagation; encryption is done character by character with keys $K_i$ that are independent of the data
  \[ C_i = E_{K_i}(P_i) \]
- Function E is simple, the function which computes the key sequence is complex
- Example: Vigenère cipher, One Time Pad
  \[ C_i = (P_i + K_i) \mod 26 \]
Stream cipher encryption

**SENDER**

(\text{Secret key}, \text{Initial value}) \rightarrow \text{Key stream}

(\text{Key stream}, \text{Message}) \rightarrow \text{Ciphertext}

**RECEIVER**

(\text{Secret key}, \text{Initial value}) \rightarrow \text{Key stream}

(\text{Ciphertext}, \text{Key stream}) \rightarrow \text{Message}

The initial value can be public or secret, but it must not repeat during the lifetime of the secret key.

This is the operation of the basic, so called synchronous stream cipher.
Synchronous stream cipher: encryption

- Initial state
- Key
- State
- $K_i$
- Keystream
- Message $P_i$
- Ciphertext $C_i$
- xor

Diagram:

- Initial state arrow to State
- Key arrow to State
- State arrow to Keystream
- Keystream arrow to xor
- xor arrow to Message
- xor arrow to Ciphertext
- Ciphertext arrow to state update
- state update arrow back to State
Synchronous stream cipher: decryption

Initial state → State

Key

State update

$K_i$ → Keystream

Ciphertext $C_i$ → xor

Message

$P_i$
Stream ciphers: Security

- Known plaintext gives known key stream. Chosen plaintext gives the same but nothing more.
- Chosen ciphertext attack may be a useful method for analysing a self-synchronising stream cipher.
- The attacker of a stream cipher may try to find one internal state of the stream cipher to obtain a functionally equivalent algorithm without knowing the key.
- Distinguishing a key stream sequence from a truly random sequence allows also the keystream to be predicted with some accuracy. Such attack is also called prediction attack.

Requirements:
- Long period
- The initial state value can be public or secret, but it must not repeat during the lifetime of the secret key.
- Given a fixed initialisation value, the stream cipher generates a different keystream for each different key.
Stream ciphers: Designs

Linear feedback shift register (LFSR)
– LFSRs are often used as the running engine for a stream cipher.

Stream cipher design based on LFSRs uses a number of different LFSRs and nonlinear Boolean functions coupled in different ways. Three common LFSR-based types of stream cipher can be identified:
– Nonlinear combination generators: The keystream is generated as a nonlinear function of the outputs of multiple LFSRs
– Nonlinear filter generators: The keystream is generated as a nonlinear function of stages of a single LFSR.
– Clock controlled generators: In these constructions, the necessary nonlinearity is created by irregular clocking of the LFSRs. The GSM encryption algorithm A5/1 is an example of a stream cipher of this type.
The taps are defined by giving the feedback polynomial
\[
f(x) = x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \ldots + c_1x + c_0
\]
for all \( t \geq n \).
LFSR: Example

NOTE: Assume now that everything is binary, that is, in bits. Sums are taken mod 2. (Non-binary LFSRs exist.)

\[ f(x) = x^4 + x^3 + 1 \]

\[ \Rightarrow c_0 = c_3 = 1 \text{ and } c_1 = c_2 = 0 \]

Let us take this as an initial state: 0 0 1 1

Then the next state is this: 1 0 0 1

And so on:

\[
\begin{array}{cccc}
0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 \\
\end{array}
\]

For how long it goes? ... ... ... ... ...
LFSR statistical properties

The maximum length of the cycle for an LFSR of length $n$ is $2^n - 1$. With maximum cycle the LFSR produces a sequence of length $2^n - 1$.

A maximum length sequence has ideal statistical properties:

- $2^{n-1} - 1$ zeroes and $2^{n-1}$ ones
- One string of ones of length $n$; one string of zeroes of length $n-1$
- Also ones and zeroes occur in about equally many pairs, triples ... , and so on.

A maximum length sequence (m-sequence) is achieved using a so-called primitive polynomials. For a source of primitive polynomials see:

http://fchabaud.free.fr/English/default.php?COUNT=1&FILE0=Poly
Autocorrelation function

• The spectral properties of a periodic sequence can be analyzed using a number of transforms related to discrete Fourier transform. One such transform is the Autocorrelation function $C(k), k \in \mathbb{N}$, defined as follows:

• Let $N$ be the length of the cycle (period) of the sequence $s_0, s_1, \ldots, s_i, \ldots$. Then

$$C(k) = \frac{1}{N} \sum_{i=0}^{N-1} (2s_i - 1)(2s_{i+k} - 1), k \in \mathbb{N}$$

• Clearly, $C(k) = C(N - k)$, for all $k \in \{0, 1, \ldots, N\}$. 
Golomb’s randomness postulates

**R1**: In the cycle of the sequence the number of 1-bits differs from the number of 0-bits by at most 1.

**R2**: In the cycle of the sequence, at least $\frac{1}{2}$ of the runs have length 1, at least one $\frac{1}{4}$ have length 2, at least $\frac{1}{8}$ have length 3, etc., as long as the number of runs so indicated exceeds 1. Moreover, for each of these lengths, there are (almost) equally many gaps and blocks.

**R3**: Let $N$ be the length of the cycle (period) of the sequence $(s_i)$. The autocorrelation function is two-valued. That is, for some integer $K$:

$$C(k) = \begin{cases} 
1, & \text{if } k = 0, \\
\frac{K}{N}, & \text{if } 1 \leq k \leq N-1 
\end{cases}$$

**Note**: In general the autocorrelation function takes more than two values

**Definition**: A binary sequence which satisfies Golomb’s randomness postulates is called a pseudo-noise or a pn-sequence.
Consider the sequence with cycle length 15:

0 1 1 0 0 1 0 0 0 1 1 1 1 0 1

R1: The number of 0-bits is 7, the number of 1-bits is 8
R2: The sequence has eight runs:
   4 runs of length 1 (2 gaps and 2 blocks)
   2 runs of length 2 (1 gap and 1 block)
   1 run of length 3 (1 gap)
   1 run of length 4 (1 block)
R3: The autocorrelation function $C(k)$ takes on two values
   $C(0) = 1$ and $C(k) = -1/15$, for $k \neq 0$
Combination generator

Example: Threshold generator

\[ t(x_1, x_2, x_3) = 1, \text{ if at least two of the inputs are equal to 1} \]
\[ 0, \text{ otherwise} \]
Clock Controlled generators

A clocking sequence is derived. The clocking sequence determines how the LFSRs are shifted

Example: A5/1

Clock bits are read. The LFSRs which are in majority, are shifted

Example: Shrinking generator
If the $cl_i = 0$, then $x_i$ is dropped


RC4

Register of 256 octets initialised using the key. Counter $i$ is set to zero. Then:

\[
\begin{array}{c|c|c}
S(i) & S(j) \\
\hline
\end{array}
\]

\[j = S(i)\]

$S(i)$ are $S(j)$ swapped

\[k = (j + S(j)) \mod 256\]

\[\text{output} = S(k)\]

\[i = (i + 1) \mod 256\]
4.2 Block cipher confidentiality modes of operation

Block ciphers are used in different modes of operation.

- AES modes of operation:
  - ELECTRONIC CODEBOOK MODE (ECB)
  - CIPHER BLOCK CHAINING (CBC)
  - CIPHER FEEDBACK (CFB)
  - OUTPUT FEEDBACK (OFB)
  - COUNTER MODE (CTR)

standardized by NIST, Special Publication 800-38A, see:

DES algorithm not secure any more (small key size), enhancement
- Triple DES Special Publication 800-67
Electronic Code Book (ECB) Mode: Encryption

\[ P_i \rightarrow \text{Block cipher} \rightarrow C_i \]

Key: \( K \)

Encryption block diagram:

- Input: \( P_i \)
- Key: \( K \)
- Cipher: Block cipher
- Output: \( C_i \)
ECB encryption
Electronic Code Book Mode: Decryption

\[ K \rightarrow \text{Block cipher decryption} \rightarrow C_i \rightarrow P_i \]
Cipher Block Chaining (CBC) Mode: Encryption
Output Feed Back (OFB) Mode

Synchronous Key Stream Generator:
Identical for encryption and decryption
Cipher Feed Back (CFB) Mode: Encryption

Self-Synchronising Stream Cipher: Decryption device is identical, only $P_i$ and $C_i$ change places.
Counter Mode

Synchronous Key Stream Generator

IV

Counter=0  Counter=1  Counter=2  Counter = length - 1

K → AES encryption  K → AES encryption  K → AES encryption  K → AES encryption

\( K_0 = K[S[0]...K[S[127]] \)
\( K_1 = K[S[128]...K[S[256]] \)
\( K_2 = K[S[256]...K[S[384]] \)
UMTS Encryption algorithm f8

A nonstandard hybrid mode: CTR + OFB

\[ CT[i] = PT[i] \oplus KS[i] \]

\( BLKCTR = 0 \)

\( BLKCTR = 1 \)

\( BLKCTR = 2 \)

\( BLKCTR = n \)

\( CK \rightarrow \) KASUMI

\( CK' \rightarrow \) KASUMI

\( CK \rightarrow \) KASUMI

\( CK \rightarrow \) KASUMI

\( CK \rightarrow \) KASUMI

\( KS[0] \ldots KS[63] \)

\( KS[64] \ldots KS[127] \)

\( KS[128] \ldots KS[191] \)
Triple DES (TDEA)

DES algorithm is not strong any more (small key size)

Double DES with two different keys $K_1$ and $K_2$ not good either (security not more than single DES) due to the Meet-in-the-Middle Attack (see next slide):

Triple DES Special Publication 800-67, see http://csrc.nist.gov/publications/nistpubs/index.html

Triple DES with two keys

$$C = E_{K_1} (D_{K_2} (E_{K_1} (P)))$$

reduces to single DES if we select $K_1 = K_2$. In this manner compatibility with legacy applications can be achieved.
Meet in the Middle

Double DES with two different keys $K_1$ and $K_2$ is not good: security is not (essentially) more than single DES due to the Meet-in-the-Middle Attack. Such attack can be launched when the attacker has two known plaintext-ciphertext pairs $(P, C)$ and $(P', C')$ where encryption is done with the same keys values $K_1$ and $K_2$. Then the attacker has

$$C = E_{K_2} (E_{K_1} (P)) \text{ and } C' = E_{K_2} (E_{K_1} (P'))$$

or what is the same:

$$D_{K_2} (C) = E_{K_1} (P) \text{ and } D_{K_2} (C') = E_{K_1} (P').$$
Meet in the Middle …

Now we make a table $T$ with a complete listing of all possible pairs $K_2, D_{K_2}(C)$ as $K_2$ runs through all possible $2^{56}$ values. The table has $2^{56}$ rows with 120 bits on each row. We make one more column to this table, and fill it with $K_1$ values as follows: For each $K_1$ we compute the value $E_{K_1}(P)$ and search in the table $T$ for a match $D_{K_2}(C) = E_{K_1}(P)$. For each $K_2$ we expect to find a (almost) unique $K_1$ such that such a match occurs. Now we go through all key pairs $K_1, K_2$ suggested by table $T$, and test against the equation $D_{K_2}(C') = E_{K_1}(P')$ we have based on the second plaintext – ciphertext pair $(P',C')$. The solution is expected to be unique. The size of table $T$ is $2^{56} (56 + 64 + \sim 56$ bits) $< 2^{64}$ bits, which is the memory requirement of this attack. The number of encryptions (decryptions) needed is about $4 \cdot 2^{56} = 2^{58}$. 