Chapter 2

Stream Ciphers

Further Reading: [Sim92, Chapter 2]

2.1 Introduction

Remember classification:

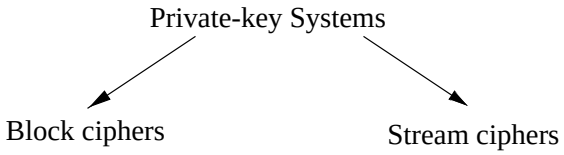

Figure 2.1: Private-key cipher classification

Block Cipher: $\bar{Y} = y_1, y_2, \ldots, y_n = e_k(x_1), e_k(x_2), \ldots, e_k(x_n)$,
e.g. the key does not change with every block
Stream Cipher: $\bar{Y} = y_1, y_2, \ldots, y_n = e_{z_1}(x_1), e_{z_2}(x_2), \ldots, e_{z_n}(x_n)$
with the "keystream" $= z_1, z_2, \ldots, z_n$

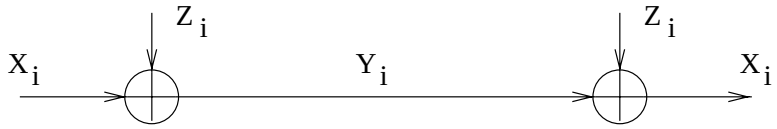

Figure 2.2: Most Popular Encryption/Decryption Function

Most popular en/decryption function: modulo 2 addition

Assume: $x_i, y_i, z_i \in \{0, 1\}$

$y_i = e_{z_i}(x_i) = x_i + z_i \bmod 2 \rightarrow$ encryption
$x_i = e_{z_i}(y_i) = y_i + z_i \bmod 2 \rightarrow$ decryption

Remarks:

1. Developed by Vernam in 1917 for Baudot Code on teletypewriters.

2. The modulo 2 operation is equivalent to a 2-input XOR operation.

   Why are encryption and decryption identical operations? Truth table of modulo 2 addition:

a	b	$c = a + b \bmod 2$
0	0	$0 + 0 = 0 \bmod 2$
0	1	$0 + 1 = 1 \bmod 2$
1	0	$1 + 0 = 1 \bmod 2$
1	1	$1 + 1 = 0 \bmod 2$

.

   $\Rightarrow$ modulo 2 addition yields the same truth table as the XOR operation.

3. Encryption and decryption are the same operation, namely modulo 2 addition (or XOR).

   Why? We show that decryption of ciphertext bit y_i yields the corresponding plaintext

bit.

Decryption: $y_i + z_i = \underbrace{(x_i + z_i)}_{\text{encryption}} + z_i = x_i + (z_i + z_i) \equiv x_i \bmod 2$.

Note that $z_i + z_i \equiv 0 \bmod 2$ for $z_i = 0$ and for $z_i = 1$.

Example: Encryption of the letter 'A' by Alice.

'A' is given in ASCII code as $65_{10} = 1000001_2$.

Let's assume that the first key stream bits are $\rightarrow z_1, \ldots, z_7 = 0101101$

Encryption by Alice:	plaintext x_i:	1000001	= 'A'	(ASCII symbol)
	key stream z_i:	0101101		
	ciphertext y_i:	1101100	= 'l'	(ASCII symbol)
Decryption by Bob:	ciphertext y_i:	1101100	= 'l'	(ASCII symbol)
	key stream z_i:	0101101		
	plaintext x_i:	1000001	= 'A'	(ASCII symbol)

2.2 One-Time Pad and Pseudo-Random Generators

Definition 2.2.1 *Unconditional Security*

A cryptosystem is unconditionally *secure if it cannot be broken even with infinite computational resources.*

Definition 2.2.2 *One-time Pad (OTP)*

A cryptosystem developed by Mauborgne based on Vernam's stream cipher consisting of:

$|\mathcal{P}| = |\mathcal{C}| = |\mathcal{K}|$,

with $x_i, y_i, k_i \in \{0, 1\}$.

encrypt $\rightarrow e_{k_i}(x_i) = x_i + k_i \bmod 2$.

decrypt $\rightarrow d_{k_i}(y_i) = y_i + k_i \bmod 2$.

Theorem 2.2.1 *The OTP is unconditionally secure if keys are only used once.*

Remarks:

1. OTP is the only provable secure system:

$$\begin{aligned} y_0 &= x_0 + K_0 \bmod 2 \\ y_1 &= x_1 + K_1 \bmod 2 \\ &\vdots \end{aligned}$$

   each equality is a linear equation with 2 unknowns.

   $\Rightarrow$ for every y_i, $x_i = 0$ and $x_i = 1$ are equally likely.

   $\Rightarrow$ holds only if $K_0, K_1, \ldots$ are not related to each other, i.e., K_i must be generated trully randomly.

2. OTP are impractical for most applications.

Question: Can we "emulate" a OTP by using a short key?

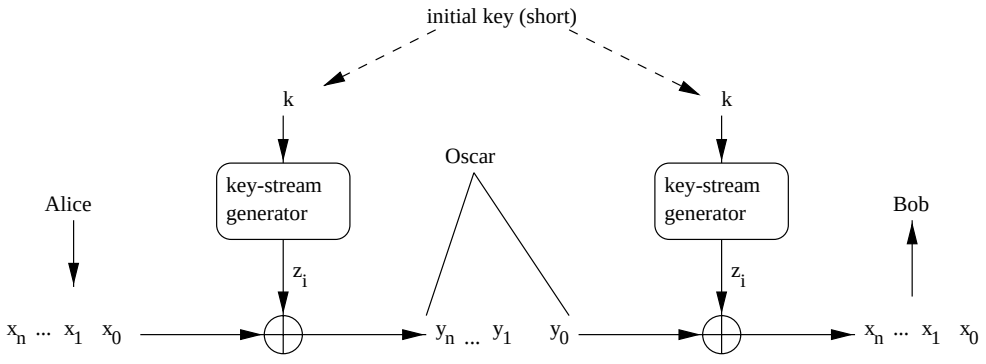

Figure 2.3: Stream cipher model

Classification by key-stream generator:

a) "synchronous stream cipher"

$z_i = f(k) \rightarrow$ pseudo-random generator (PRG).

b) "asynchronous stream cipher"

$z_i = f(k, y_{i-1}, y_{i-2}, \ldots, y_{i-N}) \rightarrow$ feedback of cipher.

c) The key issue is that Bob has to 'match' the exact z_i to get the correct message. In order to do this, both key-stream generators have to be synchronized.

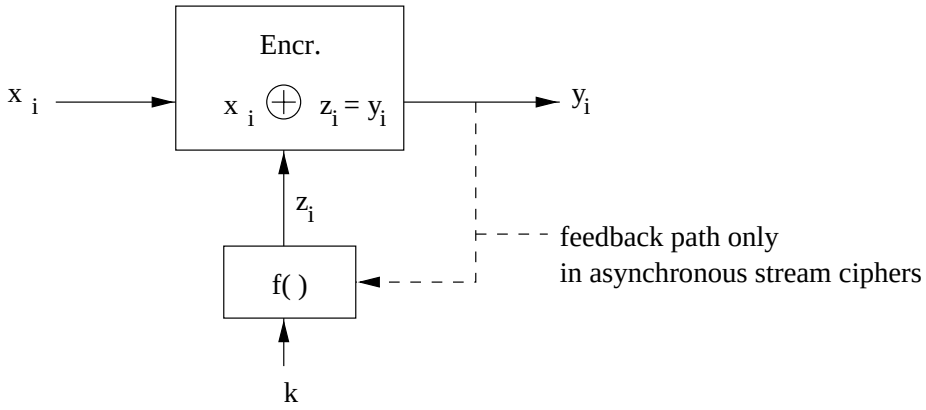

Figure 2.4: Asynchronous stream cipher

It is important to note that key stream generators must not only possess good statistical properties, which is true for other pseudo-random generatores as well, but they must also be *cryptographically secure*:

Definition 2.2.3 *Cryptographically secure pseudo-random generators*
A pseudo random generator (key stream generator) is cryptographically secure *if it is unpredictable. That is, given the first n output bits of the generator, it is computatinally infeasible to compute the bits* $n+1, n+2, \ldots$

2.3 Synchronous Stream Ciphers

The keystream $z_1, z_2, \ldots$ is a pseudo-random sequence which depends only on the key.

2.3.1 Linear Feedback Shift Registers (LFSR)

An LFSR consists of m storage elements (flip-flops) and a feedback network. The feedback network computes the input for the "last" flip-flop as XOR-sum of certain flip-flops in the shift register.

Example: We consider an LFSR of degree $m = 3$ with flip-flops K_2, K_1, K_0, and a feedback path as shown below.

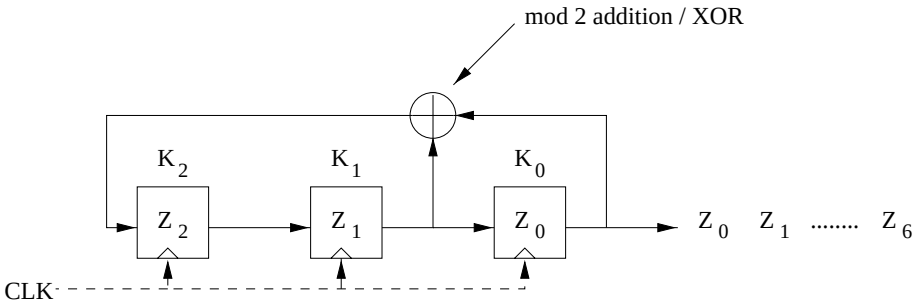

Figure 2.5: Linear feedback shift register

K_2	K_1	K_0
1	0	0
0	1	0
1	0	1
1	1	0
1	1	1
0	1	1
0	0	1
1	0	0

Mathematical description for keystream bits z_i with z_0, z_1, z_2 as initial settings:

$z_3 = z_1 + z_0 \bmod 2$

$z_4 = z_2 + z_1 \bmod 2$

$z_5 = z_3 + z_2 \bmod 2$

$\vdots$

general case: $z_{i+3} = z_{i+1} + z_i \bmod 2;\ i = 0, 1, 2, \ldots$

Expression for the LFSR:

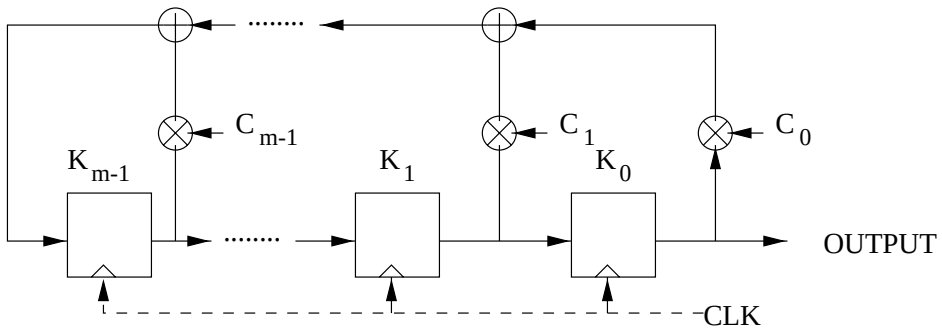

Figure 2.6: LFSR with feedback coefficients

$C_0, C_1, \ldots, C_{m-1}$ are the *feedback coefficients*. $C_i = 0$ denotes an open switch (no connection), $C_i = 1$ denotes a closed switch (connection).

$$z_{i+m} = \sum_{j=0}^{m-1} C_j \cdot z_{i+j} \bmod 2;\ C_j \in \{0, 1\};\ i = 0, 1, 2, \ldots$$

The entire key consists of:

$k = \{(C_0, C_1, \ldots, C_{m-1}), (z_0, z_1, \ldots, z_{m-1}), m\}$

Example:

$k = \{(C_0 = 1, C_1 = 1, C_2 = 0), (z_0 = 0, z_1 = 0, z_2 = 1), 3\}$

Theorem 2.3.1 *The maximum sequence length generated by the LFSR is* $2^m - 1$.

Proof:

There are only 2^m different states $(k_0, \ldots, k_m)$ possible. Since only the current state is known to the LFSR, after 2^m clock cycles a repetition must occur. The all-zero state must be excluded since it repeats itself immediately.

Remarks:

1.) Only certain configurations $(C_0, \ldots, C_{m-1})$ yield maximum length LFSRs.

For example:

if $m = 4$ then $(C_0 = 1, C_1 = 1, C_2 = 0, C_3 = 0)$ has length of $2^m - 1 = 15$

but $(C_0 = 1, C_1 = 1, C_2 = 1, C_3 = 1)$ has length of 5

2.) LFSRs are sometimes specified by polynomials.

such that the $P(x) = x^m + C_{m-1}x^{m-1} + \ldots + C_1 x + C_0$.

Maximum length LFSRs have *"primitive polynomials"*.

These polynomials can be easily obtained from literature (Table 16.2 in [Sch93]).

For example:

$(C_0 = 1, C_1 = 1, C_2 = 0, C_3 = 0) \Longleftrightarrow P(x) = 1 + x + x^4$

2.3.2 Clock Controlled Shift Registers

Example: *Alternating stop-and-go generator.*

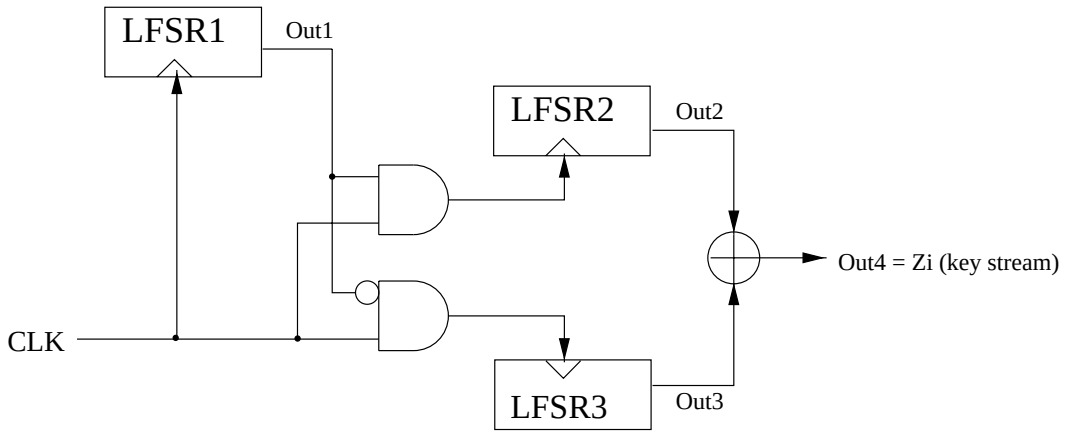

Figure 2.7: Stop-and-go generator example

Basic operation:

When $Out_1 = 1$ then LFSR2 is clocked otherwise LFSR3 is clocked.

Out_4 serves as the keystream and is a bitwise XOR of the results from LFSR2 and LFSR3.

Security of the generator:

- All three LFSRs should have maximum length configuration.
- If the sequence lengths of all LFSRs are relatively prime to each other, then the sequence length of the generator is the product of all three sequence lengths, i.e., $L = L_1 \cdot L_2 \cdot L_3$.
- A secure generator should have LFSRs of roughly equal lengths and the length should be at least 128: $m_1 \approx m_2 \approx m_3 \approx 128$.

2.4 Attacks

2.4.1 Known Plaintext Attack Against LFSRs

Assumption:

For a known plaintext attack, we have to assume that m is known.

Idea:

This attack is based on the knowledge of some plaintext and its corresponding ciphertext.

i) Known plaintext $\rightarrow x_0, x_1, \ldots, x_{2m-1}$.

ii) Observed ciphertext $\rightarrow y_0, y_1, \ldots, y_{2m-1}$.

iii) Construct keystream bits $\rightarrow z_i = x_i + y_i \bmod 2; i = 0, 1, \ldots, 2m-1$.

Goal:

To find the feedback coefficients C_i.

Using the LFSR equation to find the C_i coefficients:

$$z_{i+m} = \sum_{j=0}^{m-1} C_j \cdot z_{i+j} \bmod 2; C_j \in \{0, 1\}$$

We can rewrite this in a matrix form as follows:

$$\begin{array}{llll} i=0 & z_m & = & C_0 z_0 + C_1 z_1 + \ldots + C_{m-1} z_{m-1} \quad \bmod 2. \\ i=1 & z_{m+1} & = & C_0 z_1 + C_1 z_2 + \ldots + C_{m-1} z_m \quad \bmod 2. \\ \vdots & \vdots & \vdots & \vdots \quad \vdots \\ i=m-1 & z_{2m-1} & = & C_0 z_{m-1} + C_1 z_m + \ldots + C_{m-1} z_{2m-2} \quad \bmod 2. \end{array} \tag{2.1}$$

Note:

We now have m linear equations in m unknowns $C_0, C_1, \ldots, C_{m-1}$. The C_i coefficients are constant making it possible to solve for them when we have $2m$ plaintext-ciphertext pairs.

Rewriting Equation (2.1) in matrix form, we get:

$$\begin{bmatrix} z_0 & \dots & z_{m-1} \\ \vdots & & \vdots \\ z_{m-1} & \dots & z_{2m-2} \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ \vdots \\ c_{m-1} \end{bmatrix} = \begin{bmatrix} z_m \\ \vdots \\ z_{2m-1} \end{bmatrix} \mod 2 \tag{2.2}$$

Solving the matrix in (2.2) for the C_i coefficients we get:

$$\begin{bmatrix} c_0 \\ \vdots \\ c_{m-1} \end{bmatrix} = \begin{bmatrix} z_0 & \dots & z_{m-1} \\ \vdots & & \vdots \\ z_{m-1} & \dots & z_{2m-2} \end{bmatrix}^{-1} \cdot \begin{bmatrix} z_m \\ \vdots \\ z_{2m-1} \end{bmatrix} \mod 2 \tag{2.3}$$

Summary:

> By observing $2m$ output bits of an LFSR of degree m and matching them to the known plaintext bits, the C_i coefficients can exactly be constructed by solving a system of linear equations of degree m.

$\Rightarrow$ **LFSRs by themselves are extremely un-secure!** However, combinations of them such as the Alternating stop-and-go generator **can** be secure.