Stream Ciphers

Math 4175

§2.7.1. Stream Cipher

We have studied so far the cryptosystems where successive plaintext elements are encrypted using the same selected key $k \in \mathcal{K}$.

Cryptosystems of this type are called block ciphers.

Another type of ciphers are called stream ciphers where we generate a stream of keys (possibly not all same) called keystream and use these keys to encrypt the successive plaintext elements.

For example, if $\mathbf{x} = x_1 x_2 \cdots$ is a plaintext to be encrypted, we generate a stream of keys $\mathbf{z} = z_1 z_2 \cdots$ where each $z_i \in \mathcal{K}$ and use it to generated the ciphered text $\mathbf{y} = y_1 y_2 \cdots = e_{z_1}(x_1) e_{z_2}(x_2) \cdots$.

§2.7.1. Stream Cipher

We will learn two types of stream ciphers:

- synchronous stream ciphers
- non-synchronous stream ciphers

A stream cipher is called a synchronous stream ciphers if each element z_i in the keystream depends only on the previous key elements $\{z_1, \cdots, z_{i-1}\}$.

A stream cipher is called a non-synchronous stream ciphers if each element z_i in the keystream depends not only on the previous key elements $\{z_1, \cdots, z_{i-1}\}$ but also on the earlier plaintext elements $\{x_1, \cdots, x_{i-1}\}$ and/or cipher text elements $\{y_1, \cdots, y_{i-1}\}$.

§2.7.1. Synchronous Stream Cipher

For example, choose first four keys z_1, z_2, z_3, z_4 randomly from $\mathbb{Z}_{26}$ and then select later keys by the formula

$$z_{4+i} = (z_i + z_{i+1}) \mod 26$$

with encryption and decryptions rules

$$e_z(x) = (x + z) \mod 26$$
$$d_z(y) = (y - z) \mod 26$$

If start the key stream with $(0 \;\; 1 \;\; 2 \;\; 3)$, then we get the keystream $(0 \;\; 1 \;\; 2 \;\; 3 \;\; 1 \;\; 3 \;\; 5 \;\; 4 \;\; 4 \;\; 8 \;\; 9 \;\; 8 \cdots)$

§2.7.1. Synchronous Stream Cipher

Now if the plaintext is

```
cryptography
```

then

$$\begin{array}{cccccccccccc}
\text{c} & \text{r} & \text{y} & \text{p} & \text{t} & \text{o} & \text{g} & \text{r} & \text{a} & \text{p} & \text{h} & \text{y} \\
2 & 17 & 24 & 15 & 19 & 14 & 6 & 17 & 0 & 15 & 7 & 24 \\
+ & + & + & + & + & + & + & + & + & + & + & + \\
0 & 1 & 2 & 3 & 1 & 3 & 5 & 4 & 4 & 8 & 9 & 8 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
2 & 18 & 0 & 18 & 20 & 17 & 11 & 21 & 4 & 23 & 16 & 6 \\
\text{C} & \text{S} & \text{A} & \text{S} & \text{U} & \text{R} & \text{L} & \text{V} & \text{E} & \text{X} & \text{Q} & \text{G}
\end{array}$$

So the ciphered text is: CSASURLVEXQG

§2.7.1. Synchronous Stream Cipher

Example: Every Vigenère Cipher is a synchronous stream cipher.

If the key word length is m with key $k = (k_1, k_2, \cdots, k_m)$, then the keystream is

$$k_1, k_2, \cdots, k_m, k_1, k_2, \cdots, k_m, k_1, k_2, \cdots,$$

and it is defined by

$$z_i = \begin{cases} k_i & \text{if } 1 \leq i \leq m \\ z_{i-m} & \text{if } i \geq m+1 \end{cases}$$

Above cipher is an example of a periodic stream cipher of period m. So Vigenère Cipher is a periodic stream cipher.

Remark: We can think of any block cipher as a special case of a stream cipher.

§2.7.1. Synchronous Stream Cipher

Now we give a formal definition of Synchronous Stream Cipher.

A synchronous stream cipher is a tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{L}, \mathcal{E}, \mathcal{D})$ together with a function g, such that the following conditions are satisfied:

1. $\mathcal{P}$ is a finite set of possible plain texts.
2. $\mathcal{C}$ is finite set of possible cipher texts.
3. $\mathcal{K}$ is a finite set of possible keys, called the keyspace.
4. $\mathcal{L}$ is a finite set called the keystream alphabet.
5. g is the keystream generator, that is, g takes a key $k \in K$ as input and generates an infinite string $z_1 z_2 \cdots$ called the keystream, where $z_i \in \mathcal{L}$ for each $i \geq 1$.
6. For each $z \in \mathcal{L}$, there is an encryption rule $e_z : \mathcal{P} \to \mathcal{C}$ $(e_z \in \mathcal{E})$ and a decryption rule $d_z : \mathcal{C} \to \mathcal{P}$ $(d_z \in \mathcal{D})$ such that $d_z(e_z(x)) = x$ for every plaintext $x \in \mathcal{P}$.

§2.7.1. Binary Alphabets

Synchronous stream ciphers are often described using binary alphabets, because it is often easier to generate keystream using a hardware.

Recall that the common base is 10 with digits $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. For example,

$$5748 = (5 \times 10^3) + (7 \times 10^2) + (4 \times 10^1) + (8 \times 10^0)$$

For example, the number 11010 in the binary system can be converted to a number in the common base 10 as follows:

$$(1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 16 + 8 + 2 = 26$$

§2.7.1. Binary Alphabets

Conversely, any number in the common base can be converted to binary system as follows:

$$118 = 2^6 + 2^5 + 2^4 + 2^2 + 2$$

So $118 \equiv 1110110$ in the binary system.

In the binary system situation, we have $\mathcal{P} = \mathcal{C} = \mathcal{L} = \mathbb{Z}_2$ and

$$e_z(x) = (x + z) \mod 2$$
$$d_z(y) = (y + z) \mod 2$$

We will consider a method for generating a synchronous keystream in the binary system.

§2.7.1. Synchronous Stream Cipher

Let us start with a key $k = (k_1, k_2, \cdots, k_m)$ of a binary m-tuple.

Let $z_i = k_i$ for $1 \leq i \leq m$ as before.

Now we generate the remaining z_i's using linear recurrence of degree m as follows:

$$z_{m+i} = \sum_{j=0}^{m-1} c_j z_{j+i} \mod 2$$

for all $i \geq 1$, where $c_0, \cdots, c_{m-1} \in \mathbb{Z}_2$ are some fixed constants.

Remark: The degree is m because each term depends on the previous m terms. It is linear because each term is a linear function of previous m terms.

§2.7.1. Synchronous Stream Cipher

The linear equation $z_{m+i} = \sum_{j=0}^{m-1} c_j z_{j+i} \mod 2$ can be expressed in matrix equation as:

$$\begin{pmatrix} z_1 & z_2 & \dots & z_m \\ z_2 & z_3 & \dots & z_{m+1} \\ \vdots & \vdots & \vdots & \vdots \\ z_m & z_{m+1} & \dots & z_{2m-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-1} \end{pmatrix} = \begin{pmatrix} z_{m+1} \\ z_{m+2} \\ \vdots \\ z_{2m} \end{pmatrix}$$

that is, for short, $AC = X$ (A is called the coefficient matrix in linear algebra)

or we can write it as $C^T A = X^T$ (as noted in the text book) by taking transpose.

§2.7.1. Synchronous Stream Cipher

Example: Suppose that $m = 5$ and

$$z_1 = 0, \quad z_2 = 1, \quad z_3 = 0, \quad z_4 = 0, \quad z_5 = 0$$

with the linear recurrence relation

$$z_{5+i} = (z_i + z_{2+i}) \mod 2$$

That is, $c_0 = 1 = c_2$ and $c_1 = c_3 = c_4 = 0$.

Then the keystream is:

$$0100001001011001111100011011101\,0100001001011001111\ldots$$

This stream repeats after $31 = 2^5 - 1$ terms.

In general, for initial key length m, we get a keystream with period $2^m - 1$, which is an advantage of this cryptosystem.

§2.7.1. Linear Feedback Shift Register (LFSR)

The above key stream can be produced efficiently in hardware using linear feedback shift register or for short LFSR (see page 37 of textbook).

For example, if the plaintext is 101100111000111 and the keystream is as in the example given previously, we have

```
101100111000111
010000100101100 1
```

$$\begin{array}{c} 101100111000111 \\ 010000100101100\,1 \\ -\,-\,-\,-\,-\,-\,-\,-\,-\,- \\ 111100011101011\,0 \end{array}$$

Decryption is accomplish by adding the key sequence to the cipher text in exactly the same way.

Exercise: Consider the recurrence $z_{4+i} = (z_{1+i} + z_i) \mod 2$ with initial condition $k_1k_2k_3k_4 = 0101$ to encrypt 1000101

§2.7.1. Linear Feedback Shift Register (LFSR)

Remark: As indicated earlier, one advantage of this encryption method is that a key with large period can be generated using very little information. The long period gives an improvement over the Vigenere method, where short period allowed us to find the key. In the above example, specifying the initial vector (0,1,0,0,0) and the coefficients (1,0,1,0,0) yielded a sequence of period 31, so 10 bits were used to produce 31 bits. It can be shown that the recurrence

$$z_{n+31} = z_n + z_{n+3} \mod 2$$

and any nonzero initial condition will produce a sequence that has period $2^{31} - 1 = 2147483647$. Therefore, 62 bits produce more than two billion bits of key.

§2.7.1. Autokey Cipher

We will now give an example of a non-synchronous stream cipher, called Autokey Cipher.

Autokey Cipher is a cryptosystem with $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathcal{L} = \mathbb{Z}_{26}$. For a key $z_1 = k \in \mathcal{K}$, we define $z_i = x_{i-1}$ for all $i \geq 2$. For $0 \leq z \leq 25$, we define

$$e_z(x) = (x + z) \mod 26$$

and

$$d_z(y) = (y - z) \mod 26$$

where $x, y \in \mathbb{Z}_{26}$.

§2.7.1. Autokey Cipher

Example: Suppose that the key $k = 8$ and the plaintext is

`rendezvous`

Convert the plaintext to a sequence of integers:

$$17 \quad 4 \quad 13 \quad 3 \quad 4 \quad 25 \quad 21 \quad 14 \quad 20 \quad 18$$

Then the keystream is as follows:

$$8 \quad 17 \quad 4 \quad 13 \quad 3 \quad 4 \quad 25 \quad 21 \quad 14 \quad 20$$

By adding the corresponding numbers modulo 26, we get:

$$25 \quad 21 \quad 17 \quad 16 \quad 7 \quad 3 \quad 20 \quad 9 \quad 8 \quad 12$$

So the ciphered text is:

`ZVRQHDUJIM`

§2.7.1. Autokey Cipher

Now the decryption is done in the reverse order such as

$$x_1 = d_8(25) = (25 - 8) \mod 26 = 17$$
$$x_2 = d_{17}(21) = (21 - 17) \mod 26 = 4$$
$$x_3 = d_4(17) = (17 - 4) \mod 26 = 13$$

and so on.

Decrypt the following with autokey cipher where key is 19:

MALVVMAFBHBUQ

§2.7.1. Autokey Cipher

$$\begin{array}{ccccccccccccc}
\text{M} & \text{A} & \text{L} & \text{V} & \text{V} & \text{M} & \text{A} & \text{F} & \text{B} & \text{H} & \text{B} & \text{U} & \text{Q} \\
12 & 0 & 11 & 21 & 21 & 12 & 0 & 5 & 1 & 7 & 1 & 20 & 16 \\
| & | & | & | & | & | & | & | & | & | & | & | & | \\
19 & 19 & 7 & 4 & 17 & 4 & 8 & 18 & 13 & 14 & 19 & 8 & 12 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
-7 & -19 & 4 & 17 & 4 & 8 & -8 & -13 & -12 & -7 & -18 & 12 & 4 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
19 & 7 & 4 & 17 & 4 & 8 & 18 & 13 & 14 & 19 & 8 & 12 & 4 \\
\text{t} & \text{h} & \text{e} & \text{r} & \text{e} & \text{i} & \text{s} & \text{n} & \text{o} & \text{t} & \text{i} & \text{m} & \text{e}
\end{array}$$

§2.7.2. Cryptanalysis of LFSR Stream Cipher

Recall that the keystream in LFSR is produced by a matrix equation (linear). So LFSR is susceptible to known plaintext attack, as in the case with the Hill Cipher.

Suppose that Oscar knows the value of m and Oscar has a plaintext string $x_1 x_2 \cdots x_n$ and the corresponding cipher text string $y_1 y_2 \cdots y_n$ for $n \geq 2m$.

Then he can compute the keystream bits $z_i = (x_i + y_i) \mod 2$ for $1 \leq i \leq n$.

Now Oscar only needs to compute $c_0, c_1, \cdots, c_{m-1}$ in order to be able to reconstruct the entire keystream.

§2.7.2. Cryptanalysis of LFSR Stream Cipher

Recall the matrix equation $AC = X$. If the coefficient matrix A is invertible (mod 2), then Oscar obtains the solution $C = A^{-1}X$. More explicitely, the matrix equation

$$\begin{pmatrix} z_1 & z_2 & \cdots & z_m \\ z_2 & z_3 & \cdots & z_{m+1} \\ \vdots & \vdots & \vdots & \vdots \\ z_m & z_{m+1} & \cdots & z_{2m-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-1} \end{pmatrix} = \begin{pmatrix} z_{m+1} \\ z_{m+2} \\ \vdots \\ z_{2m} \end{pmatrix}$$

yields

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-1} \end{pmatrix} = \begin{pmatrix} z_1 & z_2 & \cdots & z_m \\ z_2 & z_3 & \cdots & z_{m+1} \\ \vdots & \vdots & \vdots & \vdots \\ z_m & z_{m+1} & \cdots & z_{2m-1} \end{pmatrix}^{-1} \begin{pmatrix} z_{m+1} \\ z_{m+2} \\ \vdots \\ z_{2m} \end{pmatrix}$$

§2.7.2. Cryptanalysis of LFSR Stream Cipher

How can Oscar guess the value of m?

Suppose that Oscar determines the following initial segment of 12 bits of keystream: 011010111100.

Supposing that $m = 2$, then the period should be $2^2 - 1 = 3$ which does not seem to be the case. To confirm, Oscar gets the matrix equation:

$$\begin{pmatrix} z_1 & z_2 \\ z_2 & z_3 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} z_3 \\ z_4 \end{pmatrix}$$

which is

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

where the recurrence could be written as:

$$z_{2+i} = c_0 z_i + c_1 z_{1+i}$$

§2.7.2. Cryptanalysis of LFSR Stream Cipher

Previous matrix equation yields the system:

$$1 = c_0 0 + c_1 1$$
$$0 = c_0 1 + c_1 1$$

Above system of equations has unique solution: $c_0 = 1$, $c_1 = 1$, which does not correctly generate the rest of the sequence.

Conclusion: $m \neq 2$.

§2.7.2. Cryptanalysis of LFSR Stream Cipher

If $m = 3$, then the period should be $2^3 - 1 = 7$, which is not the case. To confirm, we repeat the procedure to get the following matrix form:

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

This leads to an inconsistent system.

(Hint: Use Augmented matrix and perform the row operation $R_3 + R_2 \to R_3$ and then compare R_1 and R_3).

Conclusion $m \neq 3$.

§2.7.2. Cryptanalysis of LFSR Stream Cipher

If $m = 4$, then the period should be $2^4 - 1 = 15$, which we cannot check unfortunately. But we get:

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Notice that the determinant of the matrix is 1, so one can expect that the equation has unique solution.

Using standard linear algebra techniques, we may determine:

$$c_0 = c_1 = 1 \text{ and } c_2 = c_3 = 0.$$

This agrees with the rest of the data that we have. So we have found the recurrence

$$z_{4+i} = z_i + z_{1+i}.$$