**[CS304] Introduction to Cryptography and Network Security**

Course Instructor: Dr. Dibyendu Roy — Winter 2022-2023
Scribed by: Priyansh Vaishnav (202151120) — Lecture - 10 & 11 (Week 6)

LECTURE 10 — Date: - 20th Feb 2024

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

## Important Points in Stream Cipher:

1. The length of the key ($K$) should be greater than or equal to the length of the message ($M$).
2. You cannot use the same key to encrypt different messages.
3. $F(K, IV) = \mathbb{Z} \in \{0, 1\}$
   $\mathbb{Z}_0, \mathbb{Z}_1, \mathbb{Z}_2, \ldots, \mathbb{Z}_{n-1} \oplus m_0, m_1, m_2, \ldots, m_{n-1} = c_0, c_1, c_2, \ldots, c_{n-1}$
   ($\mathbb{Z}_0, \mathbb{Z}_1, \mathbb{Z}_2, \ldots, \mathbb{Z}_{n-1}$ will be a random-looking string.)
4. If you are using the same input key, it will generate the same $\mathbb{Z}_i$.
5. $F(K, IV)$ if $K$ is selected randomly and is kept secret, then the output $z_0, z_1, z_2, \ldots, z_{n-1}$ will be indistinguishable from a bit string generated by using a random bit generator.
6. $F(K, IV) = \mathbb{Z}_i, \quad 0 \leq i \leq n-1$ is a pseudorandom bit generator. In this case, the length of the output bits will be greater than the length of $K$.
7. $F(K, IV)$ if we modify at least one bit of $IV$ or $K$,
   then there will be an unpredictable change in the output.

## Synchronous Stream Cipher

In this cipher, a sequence of pseudorandom bits used for encryption and decryption is generated independently of the plaintext and ciphertext. This means that the keystream is solely determined by the secret key and an initialization vector (IV), and it is not influenced by the data being encrypted or decrypted

### State Update Function:

The state update function defines how the internal state evolves over time. It is expressed as:

$$S_{i+1} = f(S_i, k)$$

where $S_i$ is the current state, $k$ is the secret key, and $S_{i+1}$ is the updated state.

### Keystream Generation Function:

The keystream generation function produces a pseudorandom keystream based on the current state. It is represented as:

$$Z_i = g(S_i, k)$$

where $Z_i$ is the generated keystream, $S_i$ is the current state, and $k$ is the secret key.

### Ciphertext Generation Function:

The ciphertext generation function combines the keystream with the plaintext to produce the encrypted ciphertext:

$$C_i = h(Z_i, M_i)$$

where $C_i$ is the ciphertext, $Z_i$ is the keystream, and $M_i$ is the plaintext. The function $h$ represents the encryption operation.

Here, $S_0$ is the initial state, which may be determined from the secret key $k$ and an Initialization Vector (IV).

## Self-Synchronous Stream Cipher

A self-synchronizing stream cipher, also known as an asynchronous stream cipher. In this type of cipher, keystream bits are generated as a function of the secret key and a fixed number of previous ciphertext bits.

### State Representation:

The state $\sigma_i$ at time $i$ is defined as a tuple of the previous $t$ ciphertext bits:

$$\sigma_i = (C_{i-t}, C_{i-t+1}, \ldots, C_{i-1})$$

### State Update Function:

The state is updated at each iteration using the state update function $f$ with the secret key $k$ and an Initialization Vector (IV):

$$\sigma_{i+1} = f(\sigma_i, k, IV)$$

Here, $\sigma_0$ is the non-secret initial state, defined as:

$$\sigma_0 = (C_{-t}, C_{-t+1}, \ldots, C_{-1})$$

### Keystream Generation Function:

The keystream $z_i$ is generated based on the updated state $\sigma_{i+1}$ and the secret key $k$:

$$z_i = g(\sigma_{i+1}, k)$$

### Ciphertext Generation Function:

The ciphertext $c_i$ is generated by combining the keystream $z_i$ with the plaintext $M_i$ using the function $h$:

$$c_i = h(z_i, M_i)$$

## LFSR (Linear Feedback Shift Register)

Let $S_i \in \{0,1\}$ represent the binary elements of the shift register.

The shift register at time $t=0$ is given by:

$$S_0 = S_{n-1}S_{n-2}\dots S_0$$

At time $t=1$, the shift register becomes:

$$S_1 = *S_{n-1}S_{n-2}\dots S_1 \longrightarrow S_0$$

The feedback bit is calculated as:

$$S_n = L(S_0, S_1, \dots, S_{n-1}) = L(S_0)$$

At time $t=2$, the shift register becomes:

$$S_2 = *S_nS_{n-1}\dots S_2 \longrightarrow S_1$$

The feedback bit is calculated as:

$$S_{n+1} = L(S_1)$$

Given below are a few points to note regarding LSFR:

- $L(S_0, \dots, S_{n-1}) = S_n \in \{0,1\}$
- $L : \{0,1\}^n \longrightarrow \{0,1\}$
- $L_a = a_0S_0 \oplus a_1S_1 \oplus \dots \oplus a_{n-1}S_{n-1},\ a_i \in \{0,1\}$
- $L = a_0S_0 \oplus a_1S_1 \oplus \dots \oplus a_{n-1}S_{n-1} + a_n,\ a_i \in \{0,1\}$
- If $a_n = 0,\ L = L_a$, if $a_n = 1,\ L \neq L_a$

If $L(X) \oplus L(Y) \oplus L(X \oplus Y) = 0$, then $L$ is linear $\forall X, Y \in \{0,1\}^n$

- $L_1(x,y) = x \oplus y$
  $= L_1(X) \oplus L_1(Y) \oplus L_1(X \oplus Y)$
  $= (x_1 \oplus x_2) \oplus (y_1 \oplus y_2) \oplus ((x_1 \oplus y_1) \oplus (x_2 \oplus y_2))$
  $= 0$

- $L_2(x,y) = 1 \oplus x \oplus y$
  $= L_2(X) \oplus L_2(Y) \oplus L_2(X \oplus Y)$
  $= (1 \oplus x_1 \oplus x_2) \oplus (1 \oplus y_1 \oplus y_2)$
  $= (1 \oplus x_1 \oplus y_1 \oplus x_2 \oplus y_2)$
  $= 1$

**Period of an LSFR**

Let S0 be a non zero initial state, then S0 will be repeated after some m clocking of the LFSR. This m is called the period of the LSFR. **Example**

L = S0 ⊕ S1

For 3 bit LSFR: Period = $2^3$ - 1 = 7

An n-bit LFSR will be called full periodic if the period of the LFSR is $2^n$ -1.

LFSR is completely depends on Linear Feedback function.

---

LECTURE 11 Date: - 23th Feb 2024

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

## Non-Linear Feedback Bit Shift Register (NFSR)

The function $t = f(x) + f(y) + f(x, y)$ produces output values of 0 or 1. It is called non-linear due to variable output.

## Hash Function

$$h : A \to B$$

$$h(X) = Y$$

Given below are a few points regarding Hash Functions:

1. If $X$ is altered, then $h(x)$ will be completely different.
2. Given $X$, it is practically infeasible to find $x$ such that $Y = h(x)$.
3. Given $X$ and $Y = h(x)$, it is practically infeasible to find $X$ such that $h(x) = h(x')$.

$$\begin{tikzcd}
\text{Alice} & & \text{Bob} \\
X = E(M, K) \arrow[rr, "X"] & & X_1 = \mathrm{Dec}(X, K) \\
S_1 = h(M, K) \arrow[rr, "S_1"] & & S_1
\end{tikzcd}$$

If $h(x_1, K) = S_2$, then Bob accepts $X_1$.
We are able to check:

- Whether $X$ is altered during communication.
- Whether $S_1$ is altered during communication.

## Definition

A hash family is a four-tuple $(P, S, K, H)$ where the following conditions are satisfied:

- P is the set of all possible messages.
- S is the set of all possible message digests or authentication tags.
- K is the key space.
- For each $K_i \in K$, there is a hash function $h_{k_i} \in H$ such that $h_{k_i} : P \rightarrow S$.
- Here, $|P| \geq |S|$, and $|P| \geq 2 \times |S|$.

## Keyed Hash Function

If a key is involved in the computation of the hashed value, then that hash function is called a keyed hash function.

## Unkeyed Hash Function

If a key is not involved in the computation of the hash function, it is called an unkeyed hash function.

## Problem - 1

$$h : P \rightarrow S$$

Given $y \in S$, find $x \in P$ such that $h(x) = y$.

### Pre-Image Finding Problem

This problem is known as the pre-image finding problem. If you cannot find the pre-image of a hash function $h$ in feasible time (i.e., computationally hard), then the hash function $h$ is known as a preimage-resistant hash function.

## Problem - 2

$$h : P \rightarrow S$$

Given $x \in P$ and $h(x)$, find $x' \in P$ such that $x' \neq x$ and $h(x') \neq h(x)$.

### Second Pre-Image Finding Problem

If finding the second pre-image is computationally hard for $h$, then $h$ is known as a second pre-image resistant hash function.

## Problem - 3

$$h : P \rightarrow S$$

Find $x, x' \in P$ such that $x \neq x'$ and $h(x) = h(x')$.

**Collision Finding Problem**

The hash function $h$ is known as a collision-resistant hash function if finding such $x$ and $x'$ is computationally hard for $h$.

## Ideal Hash Function

Let $h : P \to S$ be a hash function. $h$ will be called an Ideal Hash Function if given $x \in P$, to find $h(x)$ either you have to apply $h$ on $x$ or look into a table corresponding to $h$ (hash table).

## 0.1 Pre-Image Finding Algorithm

$$h : X \to Y$$

Choose any $X_0 \subseteq X$ such that $|X_0| = Q$. For each $x \in X_0$, compute $y_x = h(x)$. If $y_x = y$, return $x$.

$$
\begin{aligned}
&\text{Pr[the above algorithm returns correct pre-image]} \quad \Longrightarrow \quad \text{gives you the complexity}\\
&X_0 = \{x_1, x_2, \ldots, x_Q\}\\
&E_i : \text{event } h(x_i) = y; \quad 1 \leq i \leq Q\\
&\Pr[E_i] = \frac{1}{M}\\
&\Pr[E_i] = 1 - \frac{1}{M}\\
&\Pr[E_1 \cup E_2 \cup E_3 \ldots \cup E_Q]\\
&= 1 - \Pr[E_1^c \cap E_2^c \cap E_3^c \ldots \cap E_Q^c]\\
&= 1 - \frac{Q}{\prod_{i=1} Pr[E_i^c]}\\
&= 1 - (1 - \frac{1}{M})\\
&= 1 - [1 - Q_1\frac{1}{M} + Q_2\frac{1}{M} + \ldots] \simeq 1 - [1 - Q_1\frac{1}{M}]\\
&= \frac{Q}{M}
\end{aligned}
$$

$$\text{Pr[Pre-image finding]} \simeq \frac{Q}{M}$$
$$\text{Complexity of finding pre-image} = O(M)$$