[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy
Scribed by: Aniruddh Modi (202151022)

Winter 2023-2024
Lecture 10 11 (Week 6)

1 Lecture 10

1.1 One Time Pad

We define a psuedo random function F as a function, whose output looks like they were generated using some random activity (like flipping a coin) and when we have all the parameters K and $|V$ (secret key and initialization vector).

$$F(K, |V) = Z_i \in \{0, 1\}$$

We can then get a random binary string by calling this function for some series of time.

For OTP encryption/decryption, we have a secret key and a public initialization vector. Suppose plaintext is M and ciphertext is C then :

1.1.1 Encryption

$$F(K, |V) = Z_i$$

$$C_i = M_i \oplus Z_i$$

1.1.2 Decryption

$$F(K, |V) = Z_i$$

$$C_i = M_i \oplus Z_i$$

1. If we choose K randomly and keep it secret, then F is completely indistinguishable from a random bit generator.
2. Length of output bits will be >>> length of K, that is, the period of F is very large than the length of K.

1.1.3 Stream Cipher

Stream cipher is of two types,

1. Synchronous : Key stream is generated independently of the plaintext and ciphertext bits.

   State Update function $S_{i+1} = f(S_i, K)$
   Keystream generator $Z_i = g(S_i, K)$
   Ciphertext Generator $C_i = h(z_i, m_i)$

S_0 is the initial state and may be determined by K and $|V$

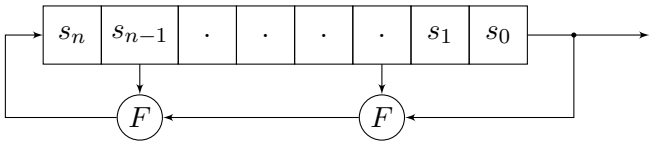
2. Self Synchronus/Asynchronys : Key stream is generated using some set of the plaintext and key.

   State Update function $\sigma_{\{}i+1\} = f(\sigma_i, K)$
   Keystream generator $Z_{\{}i\} = g(\sigma_i, K)$
   Ciphertext Generator $C_{\{}i\} = h(z_i, m_i)$

   where $\sigma_0 = (C_{i-t}, C_{i-t+1}, ..., C_{i-1})$

1.2 LFSR

Linear feedback shift register (LFSR) is a register which we will be using to generate pseudo random numbers.



$L : \{0,1\}^n \to \{0,1\}$

$$L_a = (a_0 \cdot s_0 \oplus a_1 \cdot s_1 \oplus ... \oplus a_{n-1} \cdot s_{n-1}$$

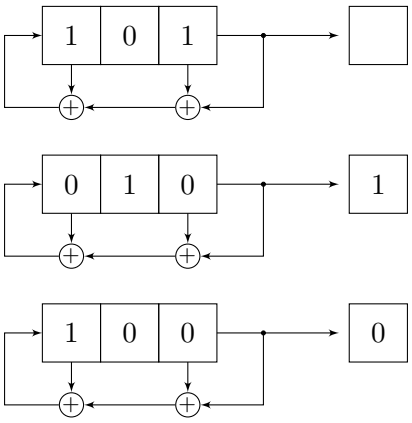
$$L = L_a \oplus a_n$$

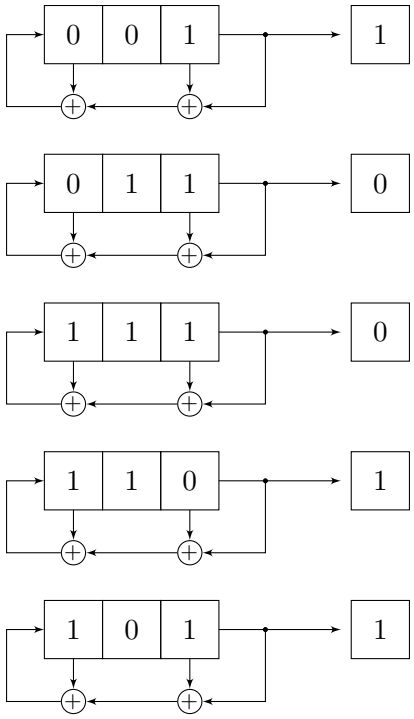
If $a_n = 0, L = L_a$ and L is called linear. On the other hand if $a_n \neq 0. L \neq L_a$ and L is called affine

1.2.1 Linear function L

$$L(X) \oplus L(Y) \oplus L(X \oplus Y) = 0$$

1.2.2 Example





1.2.3 Period of LFSR

If we start with a non zero state and reach it again agter m steps, then its period is m. If suppose there were n bits and $m = 2^n - 1$, then it is called a full periodic LFSR.

1.2.4 Connection polynomial of LFSR

Let $f(x) = 1 + c_1x + x_2x^2 + ... + c_nx^n$. Then, the $L = c_1S_{n-1} \oplus c_2s_{n-2} + ... + c_ns_0$. Now, given that $f(x) \in \mathbb{F}_2[x]$ and $def(f(x)) \leq n$.

1. If $f(x)$ is a primitive polynomial, then the period of this LFSR will be full.
2. If $f(x)$ is irreducible, then period of LFSR will divide $2^n - 1$.
3. If it is reducible, then the period will be different for different states.

2 Lecture 11

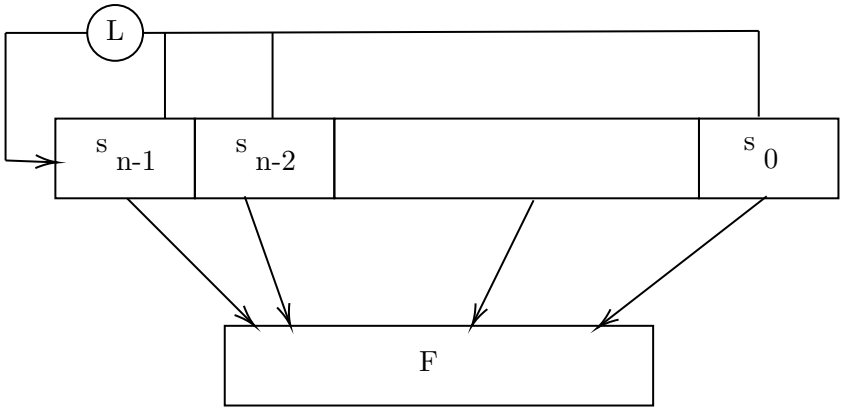
Now let $K = (K_0, \ldots, K_{n-1})$. If we know any message bit and all the keystream bits, then we can decipher the secret key.

2.0.1 Known Plaintext Attack

$Z_i = m_i \oplus C_i$ where $0 \leq 1 \leq n - 1$
$Z_0, Z_1, \ldots Z_{n-1}$
$Z_0 = K_0, Z_1 = K_1, \ldots, Z_{n-1} = K_{n-1}$
If we have keystream bits and then we be able to prepare a system of linear equations.

2.0.1.1 LFSR with non linear filter function: $F : \{0,1\}^L \to \{0,1\}$

State update of LFSR, using linear feedback and shifting.



State update function of Non linear filter LFSR is α

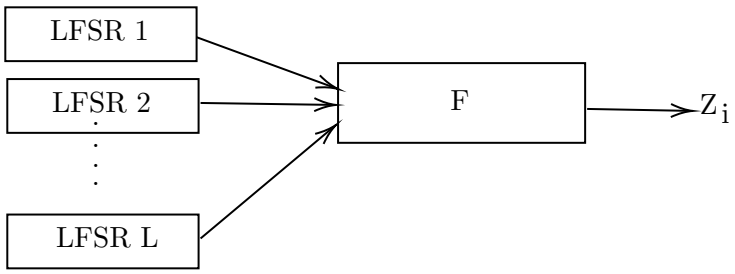
$$S_{t+1} = \alpha(S_t)$$

$$Z_{t+1} = F(S_{t+1})$$

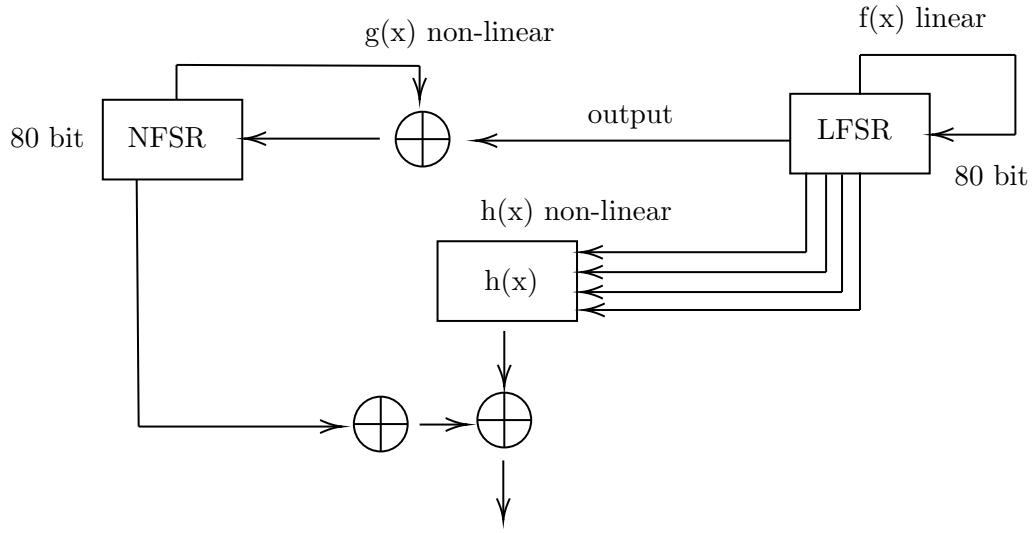
$$L = c_{n-1}s_0 \oplus c_{n-2}s_1 \ldots \oplus c_0 s_{n-1}$$

$$S^{t+1} = \begin{bmatrix} s_0^{t+1} \\ s_1^{t+1} \\ \vdots \\ s_{n-1}^{t+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ c_{n-1} & c_{n-2} & \ldots & c_0 \end{bmatrix} \begin{bmatrix} s_0^t \\ s_1^t \\ \vdots \\ s_{n-1}^t \end{bmatrix}$$

2.0.1.2 LFSR with combiner function: $F : \{0,1\}^L \to \{0,1\}$



2.0.1.3 Non-linear feedback bit shift register Feedback function will be non-linear Grain:



2.1 Hash Function

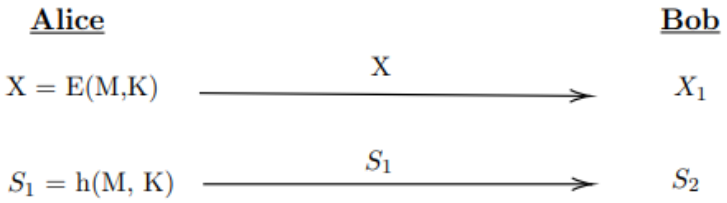
It is a mapping from one set to another set.

$$h : A \rightarrow Bh(X) = Y$$

Every hash function has following properties :

1. It X is altered to X‘, then $h(X$‘) will be completely different from $h(X)$.
2. Given Y, it is practically infeasible to find X s.t. $h(X) = Y$
3. Given X and $Y = h(X)$, it is practically infeasible to find X‘ s.t. $h(X) = h(X$‘)

Let us consider this scenario where Alice is encrypting some message using symmetric key and sending it to Bob



Now, we want to verify that the message is coming from Alice and it has not been altered. Suppose if Bob receives the altered cipher text $\tilde{X}$. Then on decryption using key K, Bob will not get the message M. So we need to ensure that the cipher text is coming from Alice and has not been altered. For that we use hash function.

If $h(X_1, K) = S_2$, then Bob accepts X_1.

Since for hash function, if the input is changed even slightly, there is huge change in output. So if X_1 would have been altered even on a single bit, the output will not satisfy. Also, since we know input X_1 and the output of $h(M, K) = S_2$, still because of the properties of hash function, we still

cannot determine message m and it is secure. This is known as the Message Authentication Code. We are able to verify :

- Whether X is altered during communication.
- Whether S1 is altered during communication.

2.1.1 Definition

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

1. P is the set of all possible messages
2. S is the set of all possible message digests or authentication tags(all output)
3. K is the Key space.
4. For each $K_1 \in k$, there is a hash function h_{K_1} such that :

$$h_{K_1} : P \to S$$

where, $|P| \geq |S|$. More interestingly, $|P| \geq 2 \times |S|$

2.1.2 Pre-image finding problem

$$h : P \to S$$

Given $y \in s$ find $x \in p$ such that $h(x) = y$.

If for h, if you can not find a preimage in a feasible time, then h is called preimage resistant hash function.

2.1.3 Second Pre-image finding problem

$$h : P \to S$$

Given $x \in p$ and $h(x)$, find $x' \in p : x' \neq x$ and $h(x) = h(x')$.

If for h, if you can not find the second-preimage in a feasible time, then h is called second preimage resistant hash function.

2.1.4 Collision finding problem

$$h : P \to S$$

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

If for h, if you can not find collision in feasible time, then h is called collision resistant hash function.

2.1.5 Ideal hash function

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

$$h : X \to Y; |Y| = M$$

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

$X_0 = \{x_1, x_2, \ldots, x_Q\}$
$E_i :$ event $h(x_i) = y; 1 \leq i \leq Q$

$$\Pr[E_i] = \frac{1}{M}$$
$$\Pr[E_i^C] = 1 - \frac{1}{M}$$
$$\Pr[E_1 \cup E_2 \cup \cdots \cup E_Q] = 1 - \Pr[E_1^C \cap E_2^C \cap \cdots \cap E_Q^C] = 1 - \prod_{i=1}^{Q} \Pr[E_i^C] = 1 - \left(1 - \frac{1}{M}\right)^Q$$
$$\therefore \Pr[E_1 \cup E_2 \cup \cdots \cup E_Q] = 1 - \left[1 - \binom{Q}{1}\frac{1}{M} + \binom{Q}{2}\frac{1}{M^2} + \binom{Q}{3}\frac{1}{M^3} + \ldots\right] \approx 1 - \left[1 - \binom{Q}{1}\frac{1}{M}\right]$$
$$\therefore \Pr[E_1 \cup E_2 \cup \cdots \cup E_Q] = \frac{Q}{M}$$

$$\Pr[\text{preimage finding}] \approx \frac{Q}{M}$$

Complexity of finding preimage $= O(M)$