[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy
Scribed by: Dipean Dasgupta (202151188)

Winter 2023-2024
Lecture 10,11 (Week 6)

1 Stream Ciphers:

One significant class of symmetric-key encryption systems is represented by stream ciphers. They are, in a way, extremely basic block ciphers with a block length of one. Their ability to alter the encryption transformation for every plain text(PT) symbol being encrypted is what makes them valuable. Because they do not propagate faults, stream ciphers are useful in scenarios where transmission failures are highly likely. They can also be utilized in situations when data processing needs to be done one symbol at a time (such as when the equipment has limited data buffering or no memory).
The encryption is done bitwise. (one bit at a time.)

$M = M_0 \dots m_l \quad m_i \in 0, 1$
$ENC(M, K) = e(M_0, Z_0), e(M_1, Z_1) \dots e(M_l, Z_l)$

$M = m_0 m_1 \dots m_l \quad m_i \in 0, 1$
$K = k_0 k_1 \dots k_l$
$C = m \oplus k = (m_0 \oplus k_0)(m_1 \oplus k_1) \dots (m_l \oplus k_l)$

Encryption: $C_i = m_i \oplus k_i$

Decryption: $M = C \oplus K$
Rules:

- Same key can't be reused to encrypt.
- len(Message) $\leq$ len(K) . One Time Padding have to be implemented.
- F(F,IV)= $Z_i \qquad 0 \leq 1 \leq n$
  (Length(o/p bits) > length(K)); K $\rightarrow$ Secret IV
- F(K,IV_1)=$Z_{i(1)} \qquad 0 \leq i \leq n-1$
  F(K,IV_2)=$Z_{i(2)} \qquad 0 \leq i \leq n-1$

Types of Stream Cipher:

- Synchronous Stream Cipher
- Asynchronous/Self Synchronous

1.1 Synchronous Stream Cipher:

Synchronous stream cipher creates a ciphertext stream of equal length by working on plaintext one bit or byte at a time. Stream ciphers use a key and an initialization vector (IV) to create a keystream that encrypts plaintext constantly, often one bit or byte at a time, in contrast to block ciphers, which encrypt fixed-size blocks of plaintext all at once. The ciphertext is then created by combining this keystream with the plaintext via an XOR operation.
Properties:

- keystream generated
- index of PT bits and CT bits
- State update function: $S_{i+1} = f(S_i, K)$
- Keystream generation : $Z_i = g(S_i, K)$
- Ciphertext generation: $C_i = h(Z_i, M_i)$

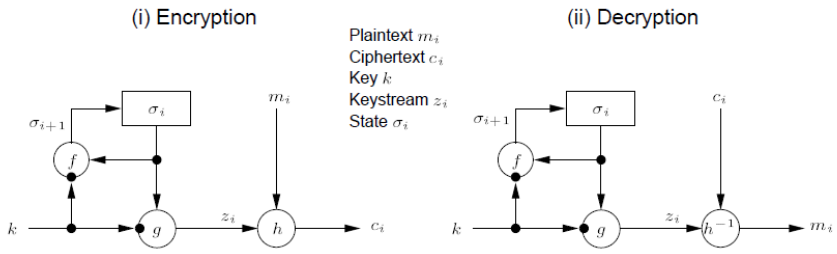

Figure:Synchronous Stream Cipher
Source:HAC

Criteria for Synchronization: In order to enable correct decryption, a synchronous stream cypher requires that the sender and recipient be synchronised using the same key and operating at the same place (state) inside that key.
Decryption fails if ciphertext digits are added or removed during transmission, causing synchronisation to be lost. This can only be fixed by using extra re-synchronization procedures. Re-initialization, putting unique markers at regular intervals in the ciphertext, or, assuming there is sufficient redundancy in the plaintext, attempting every keystream offset are methods for regaining synchronization.

No Error propagation The decoding of other ciphertext digits remains unaffected by the modification (but not deletion) of a ciphertext digit during transmission.

Active Attacks: Since of attribute (i), an active adversary may potentially be noticed by the decryptor if they insert, remove, or replay ciphertext digits since it instantly throws the system out of synchronization.
Due to property (ii), an active adversary may be able to alter specific ciphertext numbers and be aware of the precise impact those changes have on the plaintext. This demonstrates the need for additional procedures to assure data integrity and data origin authentication.

1.2 Self Synchronous Stream Cipher:

An external clock signal is not necessary for the encrypting and decryption procedures to remain synchronized when using a Self-Synchronous Stream Cipher (SSSC). The transmitter and recipient of a classical synchronous stream cipher must synchronize for them to produce and use the same keystream for encryption and decryption. A second clock signal or other synchronization techniques are frequently needed for this synchronization.
On the other hand, the encryption and decryption procedures in a self-synchronous stream cipher are internally synchronized. This indicates that no external synchronization signals are used in the generation of the keystream; instead, it is based only on the key and maybe an initialization vector (IV).
Properties:

- keystream generated (Key function, prev, CT bits)
- State function: $\sigma_i = f(C_{i-t}, C_{i-t+1} \dots C_{i-1})$
- State Update : $\sigma_{i+1} = E(\sigma_1, K)$
- Keystream generation : $Z_i = g(\sigma_i, K)$
- Ciphertext generation: $C_i = h(Z_i, M_i)$

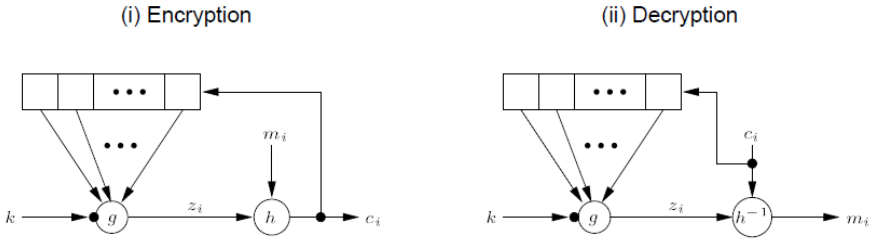

Figure:Self Synchronous Stream Cipher
Source:HAC

Linear Feedback Shift Register: A large number of the keystream generators that have been proposed in the literature use linear feedback shift registers, or LFSRs. LFSRs have several advantages, including:
1. their hardware implementation is well-suited;
2. their production of sequences with large periods
3. their production of sequences with good statistical features; and
4. their structure makes it easy to analyse them using algebraic techniques.

An L-length linear feedback shift register (LFSR) is made up of L stages (also known as delay elements), numbered $1 \dots L|1$ each of which can store one bit and have one input and one output, in addition to a clock that regulates data flow. The following actions are taken during each unit of time:

- The output of stage 0 is included in the output sequence;
- The content of stage i is transferred to stage $i|1$ for every i, $1 < i < L - 1$; and

- The new content of stage L 1 is the feedback bit S_j, which is determined by summing up, modulo 2, the previous contents of a fixed subset of stages $0, 1 \ldots L-1$

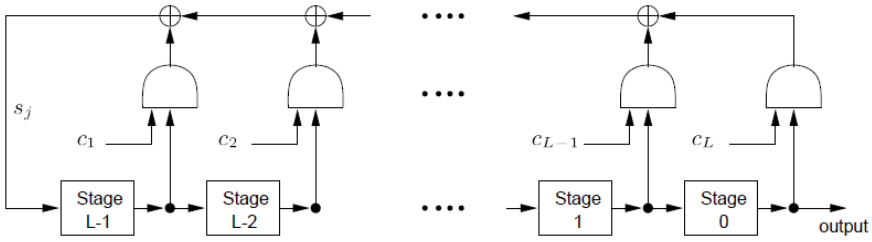

Figure:Linear Feedback Stream Cipher
Source:HAC

In Figure, the LFSR $(L, C(D))$,is represented as follows: $C(D) = 1+C_1D+C_2D^2+\cdots+C_LD^L \in \mathbb{Z}_2[D]$ is the polynomial of connections .
If $C_L = 1$, or L is the degree of $C(D)$, then the LFSR is considered non-singular. The initial state of the LFSR is denoted by $[s_{L-1}, \ldots, s_1, s_0]$ if the initial content of stage i is $s_i \in \{0, 1\}$ for each i, $0 \leq i \leq L-1$.

The following recursion can be used to uniquely determine the output sequence $s = s_0, s_1, s_2, \ldots$ if the initial state of the LFSR in Figure is $[s_{L-1}, \ldots, s_1, s_0]$.

$$s_j = (C_1 s_{j-1} + C_2 s_{j-2} + \cdots + C_L s_{j-L}) \mod 2 \quad \text{for } j > L.$$

An LFSR $(L, C(D))$ is periodic in every output sequence (i.e., for all conceivable initial states) if and only if the degree of the connection polynomial $C(D)$ is L.
Not every sequence of output from an LFSR $(L, C(D))$ is periodic if $C(D)$ has a degree smaller than L, which is known as singularity. But in the end, the output sequences are periodic; that is, the sequences that result from starting with a given finite number of words ignored are periodic. It is assumed that all LFSRs are non-singular.

Periods of LFSR output sequences
Let $C(D) \in \mathbb{Z}_2[D]$ be an L-degree connection polynomial.

1. Each of the $2^L - 1$ non-zero initial states of the non-singular LFSR $(L, C(D))$ generates an output sequence with period equal to the smallest positive integer N such that $C(D)$ divides $1 + D^N$ in $\mathbb{Z}_2[D]$ if $C(D)$ is irreducible over $\mathbb{Z}_2$. It is consistently true that N is a divisor of $2^L - 1$.

2. Each of the $2^L - 1$ non-zero initial states of the non-singular LFSR $(L, C(D))$ yields an output sequence with a maximum feasible period of $2^L - 1$ if $C(D)$ is a primitive polynomial.

When C(D) is contained in $\mathbb{Z}_2[D]$, When $(L, C(D))$ is a maximum-length LFSR, is a primitive polynomial of degree L.
An m-sequence is the result of a maximum-length LFSR with a non-zero beginning state.

LFSR with Nonlinear Filter Function:
The LFSR's output is subjected to the nonlinear filter function. To improve the created sequence's unpredictability and cryptographic qualities, nonlinear alterations are introduced. Numerous mathematical structures, including Boolean functions, modular arithmetic operations, and other nonlinear mappings, can be used to express this function.

$$f : \{0,1\}^l \rightarrow \{0,1\} \quad Boolean function(nmlinear)$$

For n-bit LFSR;
$Output$ bits $X_i \rightarrow$ keystream bits Z_i
$m_i \oplus Z_i = C_i \rightarrow$ ciphertext bits
State Update function of LFSR is α
$S_{t+1} = \alpha(S_t)$
$Z_{t+1} = f(S_{t+1})$
$t^{th} State = (S_{n-1})^t, (S_{n-2})^t, (S_0)^t$

$$L = C_{n-1}S_0 \oplus C_{n-2}$$

LFSR with combiner Function:
The LFSR's output bits and perhaps additional inputs are fed into the combiner function, which combines them to yield the desired output. The complexity of this function can vary and it can entail bitwise operations (like XOR), arithmetic operations, or other mathematical transformations. In general, using multiple LFSRs in parallel is one way to break the linearity that LFSRs are known for. These keystream generators are known as nonlinear combination generators, and f is known as the combining function. The keystream is formed as a nonlinear function f of the outputs of the component LFSRs. The rest of this paragraph shows that in order for the function f to be resistant to these specific types of cryptographic assaults, it needs to meet a number of requirements.

Nonlinear feedback shift registers
Especially in stream cyphers, a Nonlinear Feedback Shift Register (NFSR) is an essential aspect of many cryptographic systems. The feedback mechanism is nonlinear, which sets it apart from the Linear Feedback Shift Register (LFSR). The purpose of NFSRs is to generate pseudorandom bit sequences that are utilized as encryption keystreams.

$f : \{0,1\}^l \rightarrow \{0,1\}$
$t = f(x) + f(y) + f(x+y)$
$for\ t = 0,$
$f(x_0, x_1, x_2) = x_0 + x_1 + x_2$

2 Hash Function:

In the field of cryptography, a hash function is a mathematical process that accepts an input, or "message," and outputs a fixed-length string of bytes, usually in the form of a hash value or hash code. Often called a digest, the output is a distinct representation of the input data. Fast and effective hash functions offer a safe and dependable means of confirming the integrity of data, authenticating communications, and creating digital signatures.

Properties:

- Deterministic: A hash function always yields the same result for the same input.
- Fixed Output Size: Regardless of the size of the input, a hash function's output has a fixed length.
- Efficient: Hash functions ought to be computationally effective so that massive amounts of data may be processed quickly.
- Pre-image Resistance: It should be computationally impossible to determine the original input given a hash result.
- Collision Resistance: Finding two distinct inputs that result in the same hash value ought to be challenging.
- Avalanche Effect: A notable variation in the hash value should follow even a slight alteration in the input.
- Non-reversible: Reversing the hash algorithm to get the original input from the hash value should be computationally impractical.

A hash family is a four-tuple (X, Y, K, H), where the following conditions are satisfied:
1. X is a set of possible messages.
2. Y is a finite set of possible message digests or authentication tags (or just tags)
3. K, the keyspace, is a finite set of possible keys.
4. For each $k \in K$, there is a hash function $h_k \in H$. Each $h_k : X \to Y$.

While Y is always a finite set in the definition above, may not always be a finite or set. The function is sometimes referred to as a compression function if X is a finite set and $X > Y$. In this case, we'll assume the more favourable circumstance. $|X| > 2|3|$.

A function $h : X \to Y$, where X and Y are the same is an unkeyed hash function. An unkeyed hash function can be conceptualised as a hash family where $|K| = 1$, or one with a single potential key. The output of an unkeyed hash function is commonly referred to as a "message digest," while the output of a keyed hash function is referred to as a "tag."

If $h(x) = y$, then a pair $(x, y) = X \times Y$ is considered legitimate under a hash function h. In this case, h may be an unkeyed or keyed hash function. In this chapter, we mainly cover techniques to stop an opponent from creating specific kinds of valid pairs.

Let $F_{X,Y}$ denote the set of all functions from X to Y. Suppose that $|X| = N$ and $|Y| = M$. Then it is clear that $|F_{X,Y}| = M^N$. (This follows because, for each of the N possible inputs $x \in X$, there are M possible values for the corresponding output $h(x) = y$.) Any hash family F consisting of functions with domain X and range Y can be considered to be a subset of $F_{X,Y}$, i.e., $F \subseteq F_{X,Y}$. Such a hash family is termed an (N, M)-hash family.

Security of Hash Functions:
Assume that the hash function $h : X- > Y$ is unkeyed. Define $y = h(x)$, given $x \in X$. It is desirable in many cryptographic applications of hash functions that the only method to generate a valid pair (x, y) is to compute $y = h(x)$ by applying the function h to x first.
In total, three problems are defined; if a hash function is to be considered secure, it should be the case that these three problems are difficult to solve.

Problem 1: Preimage
Instance: A hash function $h : X \to Y$ and an element $y \in Y$.
Find: $x \in X$ such that $h(x) = y$.
The issue Preimage asks whether an element $x \in X$ can be found such that h(x) = y given a (possible) message digest y. A value x of that kind would be a preimage of y.A pair (x, y) is legitimate if Preimage can be solved for a given $y \in Y$. One term for a hash function that is one-way or preimage resistant is that it cannot be solved effectively using Preimage.

Problem 2: Second Preimage
Instance: A hash function $h : X \to Y$ and an element $x \in X$.
Find: $x' \in X$ such that $x' \neq x$ and $h(x') = h(x)$.
The Second Preimage problem asks if $x' \neq x$ can be discovered such that h(x') = h(x), given a message x. The goal is to identify a value x' that would be a second preimage of y. Here, we start with x, which is a preimage of y. Keep in mind that (x', h(x)) is an acceptable pair if this can be accomplished. It is common to refer to a hash function as second preimage resistant when it is incapable of being solved efficiently for Second Preimage.

Problem 3: Collision
Instance: A hash function $h : X \to Y$.
Find: $x, x' \in X$ such that $x' \neq x$ and $h(x') = h(x)$.
The issue The collision problem asks if there is any pair of different inputs, x, x', such that h(x') = h(x). Two legitimate pairs, (x, y) and (x', y), where y = h(x) = h(x'), are produced as a solution to this problem. There are several situations in which we would like to prevent this kind of thing from happening. It's common to refer to a hash function as collision-resistant when Collision cannot be addressed effectively.