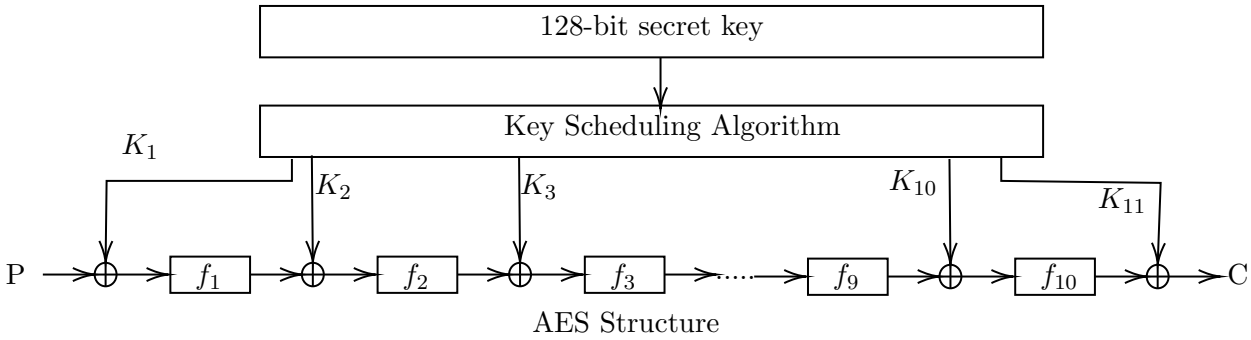
[CS309] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy
Scribed by: Kunj Thakkar (202251142)

Autumn 2024-2025
Lecture (Week 6)

1 Advanced Encryption System (AES-128)



AES Structure

AES, a type of encryption called Rijndael, has become the most popular way to encrypt data since the year 2001. It is simple and efficient, making it suitable for many different uses, such as secure communication and disk encryption.

1.1 Key Scheduling Algorithm

The AES-128 process creates 11 keys from a 128-bit key. Each key is 128 bits long and is made up of 44 smaller parts, each 32 bits long.

- **ROTWORD**: perform a cyclic shift on a 32-bit word
- **SUBWORD**: Apply substitution box on each of 4 8-bit of 32 bit word
- **XOR with Constannts**

1.1.1 constants

1. RCON[1] = 0x01000000
2. RCON[2] = 0x02000000
3. RCON[3] = 0x04000000
4. RCON[4] = 0x08000000
5. RCON[5] = 0x10000000
6. RCON[6] = 0x20000000
7. RCON[7] = 0x40000000
8. RCON[8] = 0x80000000
9. RCON[9] = 0x1b000000
10. RCON[10] = 0x36000000

1.1.2 key expansion process

Algorithm 1: AES-128 Key Expansion

```
for i = 4 to 43 do
    temp = w[i − 1];
    if i mod 4 = 0 then
        temp = SUBWORD(ROTWORD(temp)) ⊕ RCON[i/4];
    end
    w[i] = w[i − 4]⊕ temp;
end
```

2 AES Decryption Process

The AES decryption process essentially reverses the steps applied during encryption.

2.1 Inverse Round Functions

For rounds 1 to 9, the inverse round function is defined as:

$$f_i^{-1} = \text{InvSubBytes}(\text{InvShiftRows}(\text{InvMixColumns}(S)))$$

In the final round:

$$f_{10}^{-1} = \text{InvSubBytes}(\text{InvShiftRows}(S))$$

2.2 Inverse Operations

- **InvSubBytes**: Reverses the byte substitution using the inverse S-box.
- **InvShiftRows**: Rotates the rows of the state matrix to the right.
- **InvMixColumns**: Multiplies the state by the inverse MixColumns matrix.

3 Stream Cipher

Stream ciphers encrypt data at the level of individual bits, processing one bit at a time. Given a message $M = m_0 m_1 ... m_l$, where each bit $m_i \in \{0, 1\}$ represents the plaintext, encryption involves combining each plaintext bit m_i with a corresponding keystream bit k_i generated by the cipher, yielding the ciphertext C:

$$C = M \oplus K = (m_0 \oplus k_0)(m_1 \oplus k_1)...(m_l \oplus k_l)$$

where $\oplus$ represents the bitwise XOR operation.

Encryption: $C_i = m_i \oplus K_i$

Decryption: $m_i = C_i \oplus K_i$

3.1 Shannon's Notion of Perfect Secrecy

Shannon's theorem of perfect secrecy provides a rigorous foundation in cryptography, defining perfect security as a property where the ciphertext reveals no information about the original plaintext, regardless of the adversary's computational power.

Mathematically, Shannon's theorem is expressed as:

$$Pr[M = m_1] = Pr[M = m_1 | C = CH_1]$$

This states that the probability of a specific plaintext m_1 is the same, even after observing a particular ciphertext CH_1. In other words, the ciphertext does not provide any additional information about the likelihood of m_1.

To illustrate this, consider binary variables m and k, where $m \in \{0, 1\}$ represents a plaintext bit, and $k \in \{0, 1\}$ represents the key bit. Assuming that the probability of m being 0 is P and the key bit k is equally likely to be 0 or 1, we can calculate the probabilities of the resulting ciphertext C as follows:

$$\begin{aligned}
Pr[C = 0] &= Pr[M = 0 \wedge K = 0] + Pr[M = 1 \wedge K = 1] \\
&= P \times \frac{1}{2} + (1 - P) \times \frac{1}{2} \\
&= \frac{1}{2} \\
Pr[C = 1] &= Pr[M = 0 \wedge K = 1] + Pr[M = 1 \wedge K = 0] \\
&= P \times \frac{1}{2} + (1 - P) \times \frac{1}{2} \\
&= \frac{1}{2}
\end{aligned}$$

This shows that the ciphertext C is uniformly distributed. Therefore, even if the plaintext has some bias, the ciphertext remains randomized, maintaining perfect secrecy.

3.2 Conditions for Perfect Secrecy

For a stream cipher to achieve perfect secrecy, the following conditions must be met:

1. **Unique Keys:** Each message must be encrypted with a distinct key. Reusing a key for multiple messages can lead to information leakage, as patterns in the ciphertext may reveal parts of the plaintext.

2. **Key Length vs. Message Length:** The length of the key ($|K|$) must be at least as long as the message ($|M|$). If the key is shorter than the message ($|K| < |M|$), extending the key by repeating bits introduces vulnerabilities, making the plaintext more susceptible to attacks.

Adhering to these principles ensures that the stream cipher provides perfect secrecy, meaning no information about the plaintext can be inferred from the ciphertext, even by an adversary with significant computational resources.

For instance, the Vernam cipher, also known as the **One-Time Pad (OTP)**, achieves perfect secrecy by using each bit of the key exactly once. However, due to the difficulty of securely distributing keys as long as the messages, OTP is considered impractical for widespread use.

4 Block Cipher Modes of Operation

For encrypting data larger than the block size, specific operational modes are used. Let's examine two key modes:

4.1 Electronic Codebook (ECB) Mode

In ECB mode, each block undergoes independent encryption:

$$\begin{aligned} \text{Encryption:} \quad & C_i = \text{Enc}(P_i, K) \quad \forall i \in \{1, \ldots, n\} \\ \text{Decryption:} \quad & P_i = \text{Dec}(C_i, K) \quad \forall i \in \{1, \ldots, n\} \end{aligned}$$

Key Weakness: Identical plaintext blocks result in identical ciphertext blocks.

4.2 Cipher Block Chaining (CBC) Mode

CBC mode integrates the previous ciphertext block when encrypting the current block:

$$\begin{aligned} \text{Encryption:} \quad & C_i = \text{Enc}(P_i \oplus C_{i-1}, K) \quad \forall i \in \{1, \ldots, n\} \\ \text{Decryption:} \quad & P_i = \text{Dec}(C_i, K) \oplus C_{i-1} \quad \forall i \in \{1, \ldots, n\} \end{aligned}$$

Here, C_0 represents the initialization vector (IV).

5 Linear Feedback Shift Registers (LFSRs)

5.1 Basic Structure

An LFSR is a digital circuit with n bit registers and a feedback function. It's used to generate pseudorandom bit sequences.

5.2 Components and Operation

1. **Registers**: A series of n bit storage units $(s_0, s_1, \ldots, s_{n-1})$.
2. **Feedback Function**: A linear function f that determines the next state.
3. **Output**: Usually taken from the rightmost register (s_0).

5.3 Example: 5-bit LFSR

Consider a 5-bit LFSR with initial state $(1, 0, 0, 1, 1)$.

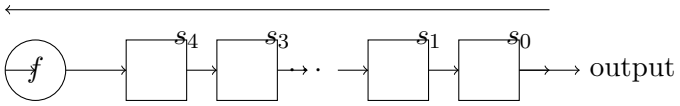


Figure 1: 5-bit LFSR Structure

5.4 Feedback Mechanism

The feedback function is typically a linear combination of register values:

$$s_n = c_1 s_{n-1} \oplus c_2 s_{n-2} \oplus \cdots \oplus c_n s_0$$

where c_i are binary coefficients (0 or 1).

5.4.1 Sample Feedback Function

For our 5-bit example:

- Coefficients: $c_1 = 1, c_2 = 0, c_3 = 0, c_4 = 1$
- Resulting function: $s_4 = s_0 \oplus s_3$

5.5 State Evolution

Starting with $(1, 0, 0, 1, 1)$, the LFSR evolves:

1. Output: 1 (value of s_0)
2. New state: $(1, 1, 0, 0, 1)$

This process continues, generating a sequence of outputs and states.

5.6 Cycle Behavior

LFSRs eventually enter a repeating cycle of states. The length of this cycle is a key characteristic of the LFSR.

5.7 Connection Polynomial

The LFSR can be described by a polynomial:

$$p(x) = 1 + c_1 x + c_2 x^2 + \cdots + c_n x^n$$

For our example: $p(x) = 1 + x + x^4$

This polynomial is crucial for analyzing the LFSR's properties and the sequence it generates.

6 Hash Function

A hash function is a computational algorithm that converts an input (or 'message') into a fixed-length string of bytes. The output, typically called a 'digest', is unique for each distinct input. We can formally express a hash function as:

$$h : A \to B \tag{1}$$

Here, h represents the hash function, A is the set of all possible inputs (messages), and B is the set of all possible outputs (hash values). For an input X, the output is denoted as:

$$h(X) = Y \tag{2}$$

where Y is the resulting hash value.

6.1 Key Characteristics of Hash Functions

A robust hash function exhibits several crucial properties that make it suitable for cryptographic applications:

1. Avalanche Effect: A minor alteration in the input (even a single bit) should produce a significantly different output. For instance, if X is changed to X', the resulting hash values $h(X)$ and $h(X')$ should be vastly different.

2. Pre-Image Resistance: Given a hash value Y, it should be computationally unfeasible to find any input X such that $h(X) = Y$.

3. Second Pre-Image Resistance: Given an input X and its hash value $Y = h(X)$, it should be impractical to find a different input X' such that $h(X) = h(X')$.

6.2 Secure Communication Scenario

Consider a secure communication scenario between Alice and Bob:

$$\begin{aligned} X &= E(M, K) \\ S_1 &= h(M, K) \\ S_2 &= h(X_1, K) \end{aligned}$$

Where: - M is the original message - K is the encryption key - E is the encryption function - S_1 is the hash of M combined with K - S_2 is the hash of the encrypted message X_1 with K

Bob can verify the message integrity by computing $h(X_1, K) = S_2$.

6.3 Formal Definition of Hash Function Family

A hash family is defined as a four-tuple (P, S, K, H), where:

1. P is the set of all possible input messages
2. S is the set of all possible message digests or authentication tags
3. K is the key space
4. For each $K_i \in K$, there exists a hash function h_{K_i} such that:

$$h_{K_i} : P \to S \tag{3}$$

where $|P| \geq |S|$ and ideally, $|P| \geq 2 \times |S|$

6.4 Hash Function Categories

Hash functions can be classified into two main types:

6.4.1 Keyed Hash Function

Incorporates a secret key into the hash computation, providing additional security.

6.4.2 Unkeyed Hash Function

Relies solely on the input message to generate the hash value.

6.5 Critical Hash Function Problems

Three key problems in hash function analysis:

1. **Pre-Image Finding Problem:** Given $h : P \to S$ and $y \in S$, find $x \in P$ such that $h(x) = y$.
2. **Second Pre-Image Finding Problem:** Given $h : P \to S$, $x \in P$, and $h(x)$, find $x' \in P$ such that $x' \neq x$ and $h(x') = h(x)$.
3. **Collision Finding Problem:** Find two distinct inputs x and x' in P such that $x \neq x'$ and $h(x) = h(x')$.

6.6 Ideal Hash Function Properties

An ideal hash function $h : P \to S$ should:

1. Produce a unique hash value for every unique input
2. Be computationally infeasible to derive the input from its hash output
3. Be infeasible to find two different inputs producing the same hash output
4. Require direct application of h on x or consultation of a precomputed hash table to find $h(x)$

6.7 Algorithms for Hash Function Problems

6.7.1 Pre-Image Finding Algorithm

Algorithm 2: Pre-Image Finding Algorithm

Input: Hash function h, target hash value y, set X, subset size Q
Output: Pre-image x such that $h(x) = y$, or failure
Choose any $X_0 \subseteq X$ such that $|X_0| = Q$;
foreach $x \in X_0$ **do**
    Compute $y_x = h(x)$;
    if $y_x = y$ **then**
        return x;
return *Failure*;

The probability of finding a pre-image using X_0 is:

$$\Pr[\text{pre-image finding}] \approx \frac{Q}{M}$$

Complexity (pre-image finding) $= O(M)$

6.7.2 Collision Finding Algorithm

Algorithm 3: Collision Finding Algorithm

Input: $X_0 \subseteq X - \{x\}$ with $|X_0| = Q$
Output: A pair of elements $\{x, x'\}$ such that $h(x) = h(x')$ and $x \neq x'$
foreach $x' \in X_0$ **do**
    Compute $y_x = h(x)$ and $y_{x'} = h(x')$;
    if $y_x = y_{x'}$ **then**
        return $\{x, x'\}$;

The collision probability ϵ is given by:

$$\epsilon = 1 - \left(\frac{M-1}{M} \cdot \frac{M-2}{M} \cdot \ldots \cdot \frac{M-(Q-1)}{M} \right)$$

Solving for Q:

$$Q = \sqrt{2 \ln \left(\frac{1}{1-\epsilon} \right) \sqrt{M}}$$

Hence, Q is approximately $\sqrt{M}$.