[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy
Scribed by: Tailor Sarthak Kalpesh (202151169)

Winter 2023-2024
Lecture (Week 5)

AES - Advanced Encryption Standard:

1. **AES-128:**
   - The AES-128 variant utilizes a block size of 128 bits and employs a total of 10 rounds in its encryption process. It utilizes a secret key size of 128 bits for encryption and decryption operations.
   - The round function remains the same throughout the first 9 rounds: $f_1 = f_2 = \ldots = f_{n-1}$. However, the final round function f_{n-1} differs from the last round function f_n.
2. **AES-192:**
   - The AES-192 variant utilizes a block size of 128 bits and employs a total of 12 rounds in its encryption process. It utilizes a secret key size of 192 bits for encryption and decryption operations.
   - Similar to AES-128, the round function remains the same throughout the first 11 rounds: $f_1 = f_2 = \ldots = f_{n-1}$. However, the final round function f_{n-1} differs from the last round function f_n.
3. **AES-256:**
   - The AES-256 variant utilizes a block size of 128 bits and employs a total of 14 rounds in its encryption process. It utilizes a secret key size of 256 bits for encryption and decryption operations.
   - Similar to AES-128 and AES-192, the round function remains the same throughout the first 13 rounds: $f_1 = f_2 = \ldots = f_{n-1}$. However, the final round function f_{n-1} differs from the last round function f_n.

Round Function of AES:

- The round function of AES consists of several operations, typically including:
  1. **SubByte:** A substitution operation that maps a byte from the input to a new byte using a substitution table.
  2. **ShiftRow:** A permutation operation that shifts the rows of the input state matrix by different offsets.
  3. **MixColumn:** A linear transformation that operates on the columns of the state matrix, mixing the bytes to provide diffusion.
- In each round of AES, except for the final round, these operations are applied sequentially to the state matrix.

Sub Byte Operation:

- **Input:** Divide the message X into 16 parts, with each part containing an 8-bit message, i.e., $X = x_0 * x_1 \dots x_{15}$, where $sizeof(x_i) = 8$ bits.
- **Notation:** Let $S : \{0,1\}^8 \to \{0,1\}^8$.
- **Description:** The Sub Byte operation substitutes each byte of the input matrix X with a corresponding byte from the S-box S. The S-box transformation is applied to each byte independently.
- **Steps:**
  1. Initialize S-box: $(C_7C_6C_5C_4C_3C_2C_1C_0) \leftarrow (01100011)$ (Hexadecimal: 63)
  2. Apply S-box substitution: $S(s_{ij}) = (a_7a_6a_5a_4a_3a_2a_1a_0)$
  3. Compute each b_i using modular addition: $b_i = (a_i + a(i+4)\%8 + a(i+5)\%8 + a(i+6)\%8 + a(i+7)\%8) \pmod 2$
  4. Update input matrix with substituted bytes: $(b_7b_6b_5b_4b_3b_2b_1b_0) = s_{ij}$

Multiplicative Inverse Calculation:

- **Objective:** Find the multiplicative inverse of a polynomial $P(x)$ modulo $x^8+x^4+x^3+x+1$.
- Let $S : \{0,1\}^8 \to \{0,1\}^8$ be the S-box operation.
- Let $X = (a_7a_6a_5a_4a_3a_2a_1a_0)$ be an 8-bit message.
- Define the polynomial $P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7$ with degree less than 8.
- **Process:** Compute the multiplicative inverse $p(x) * q(x) = 1 \pmod{x^8+x^4+x^3+x+1}$ using the Extended Euclidean Algorithm.

Shift Rows Operation:

- **Input:** The input is a 4x4 matrix S with elements from $\{0,1\}^8$.
- **Description:** In the Shift Rows operation, the second row of the matrix is circularly shifted left by 1 position, the third row is shifted left by 2 positions, and the fourth row is shifted left by 3 positions.
- **Process:**

$$\begin{pmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{10} & S_{11} & S_{12} & S_{13} \\ S_{20} & S_{21} & S_{22} & S_{23} \\ S_{30} & S_{31} & S_{32} & S_{33} \end{pmatrix} \to \begin{pmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{11} & S_{12} & S_{13} & S_{10} \\ S_{22} & S_{23} & S_{20} & S_{21} \\ S_{33} & S_{30} & S_{31} & S_{32} \end{pmatrix}$$

- **Output:** The output is the modified matrix after performing the circular shifts on the rows.

Mix Column Operation:

- **Input:** The input is a 4x4 matrix S with elements from $\{0,1\}^8$.
- **Description:** In the Mix Column operation, each column of the matrix is multiplied by a fixed polynomial and then added together with modular arithmetic.

- **Process:** Consider the columns $c \in \{0, 1, 2, 3\}$ for $i = 0$ to 3. The general formula for each element S'_{ij} is:

$$S'_{ij} = (x*S_{i,j}+(x+1)S_{(i+1) \pmod 4,j}+1*S_{(i+2) \pmod 4,j}+1*S_{(i+3) \pmod 4,j}) \pmod{x^8+x^4+x^3+x+1}$$

$$\begin{pmatrix} S'_{00} & S'_{01} & S'_{02} & S'_{03} \\ S'_{10} & S'_{11} & S'_{12} & S'_{13} \\ S'_{20} & S'_{21} & S'_{22} & S'_{23} \\ S'_{30} & S'_{31} & S'_{32} & S'_{33} \end{pmatrix} = \begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix} \times \begin{pmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{10} & S_{11} & S_{12} & S_{13} \\ S_{20} & S_{21} & S_{22} & S_{23} \\ S_{30} & S_{31} & S_{32} & S_{33} \end{pmatrix}$$

- **Output:** The output is the modified matrix S' after performing the Mix Column operation.

Key Scheduling:

- **Input:** Key K of length 128 bits, divided into 16 subkeys of 8 bits each.
- **Constants:**
  - $Rcon[1] = 01000000$
  - $Rcon[2] = 02000000$
  - $Rcon[3] = 04000000$
  - $Rcon[4] = 08000000$
  - $Rcon[5] = 10000000$
  - $Rcon[6] = 20000000$
  - $Rcon[7] = 40000000$
  - $Rcon[8] = 80000000$
  - $Rcon[9] = 1B000000$
  - $Rcon[10] = 36000000$
- **Algorithm:**
  1. For $i = 0$ to 3, set $W[i] = (key[4i] + key[4i + 1] + key[4i + 2] + key[4i + 3])$.
  2. For $i = 4$ to 43:
     - Set $temp = W[i - 1]$.
     - If $i \mod 4 == 0$, then $temp = \text{SUBWORD(ROTWORD(temp))} \oplus Rcon[i/4]$.
     - Set $W[i] = W[i - 4] \oplus temp$.

Subkeys:

- $k_1 = W[0]||W[1]||W[2]||W[3]$
- $k_2 = W[4]||W[5]||W[6]||W[7]$
- $k_{11} = W[40]||W[41]||W[42]||W[43]$

Modes of Operation:

1. **ECB (Encryption Codebook):**
   - **Encryption:** $C_i = \text{Enc}(x_i, K)$ for $1 \leq i \leq t$, $C = C_i, \ldots, C_t$.

- **Decryption:** $x_i = \text{Dec}(C_i, K)$ for $1 \leq i \leq t$.

2. **CBC (Cipher Block Chaining):**

- **Encryption:** $C_0 = IV$ (Public), $C_j = \text{Enc}(C_{j-1} \oplus x_j, K)$ for $1 \leq j \leq t$.
- **Decryption:** $C_0 = IV$ (Public), $x_j = \text{Dec}(C_j, K) \oplus C_{j-1}$ for $1 \leq j \leq t$.

3. **CFB (Cipher Feedback):**

- **Initialization:** $C_0 = IV$ (Public).
- **Encryption:** $C_j = x_j \oplus \text{Enc}(C_{j-1}, K)$.
- **Decryption:** $C_0 = IV$ (Public), for $1 \leq j \leq t$, $x_j = C_j \oplus \text{Enc}(C_{j-1}, K)$.

4. **OFB (Output Feedback):**

- **Initialization:** $C_0 = IV$ (Public).
- **Encryption:** $C_j = \text{Enc}(C_{j-1}, K)$ for $1 \leq j \leq t$, C_j is discarded.
- **XOR with plaintext:** $C_j = x_j \oplus C_j$ for $1 \leq j \leq t$.
- **Decryption:** $C_0 = IV$ (Public), for $1 \leq j \leq t$, $x_j = C_j \oplus \text{Enc}(C_{j-1}, K)$.

Proof:

Let $[M, C, K]$ be the message, ciphertext, and key bits, respectively. $M \in \{0, 1\}$ with probabilities $Pr[M = 0] = p$ and $Pr[M = 1] = 1 - p$.

$C = \text{Enc}(m, k) = m \oplus K$, where $C \in \{0, 1\}$.

$$\begin{aligned} Pr[C = 0] &= Pr\left[\{\{m = 0, k = 0\} \cup \{m = 1, k = 1\}\right] \\ &= Pr[m = 0, k = 0] + Pr[m = 1, k = 1] \\ &= Pr[m = 0] \times Pr[k = 0] + Pr[m = 1] \times Pr[k = 1] \\ &= \left(\frac{1}{2}\right) \times (p) + \left(\frac{1}{2}\right) \times (1 - p) \\ &= \frac{1}{2} \end{aligned}$$

$K \in \{0, 1\}$ with probabilities $Pr[K = 0] = \frac{1}{2}$ and $Pr[K = 1] = \frac{1}{2}$.

So, $Pr[C = 0] = \frac{1}{2}$ and $Pr[C = 1] = \frac{1}{2}$, indicating that C is randomized and can be 1 or 0 with equal probability, hence C is not dependent on M.

Now, let's find the probability of message M given ciphertext:

$$\begin{aligned} Pr[M = m_1 \| C = ch_1] &= Pr[M = m_1, C = ch_1] \\ &= Pr[C = ch_1 \| M = m_1] \times Pr[M = m_1] \\ &= \left(\frac{1}{2}\right) \times Pr[M = m_1] \\ &= Pr[M = m_1] \times \left(\frac{1}{2}\right) \end{aligned}$$

$$\therefore Pr[M = m_1 \| C = ch_1] = \frac{1}{2}$$

Note:

1. We cannot use the same key to encrypt different messages. Proof: $C_1 = M_1 \oplus K$, $C_2 = M_2 \oplus K$, $C_1 \oplus C_2 = M_1 \oplus M_2$, thus we can infer some information about messages M_1 and M_2.

2. $Len(k) \geq Len(M)$.

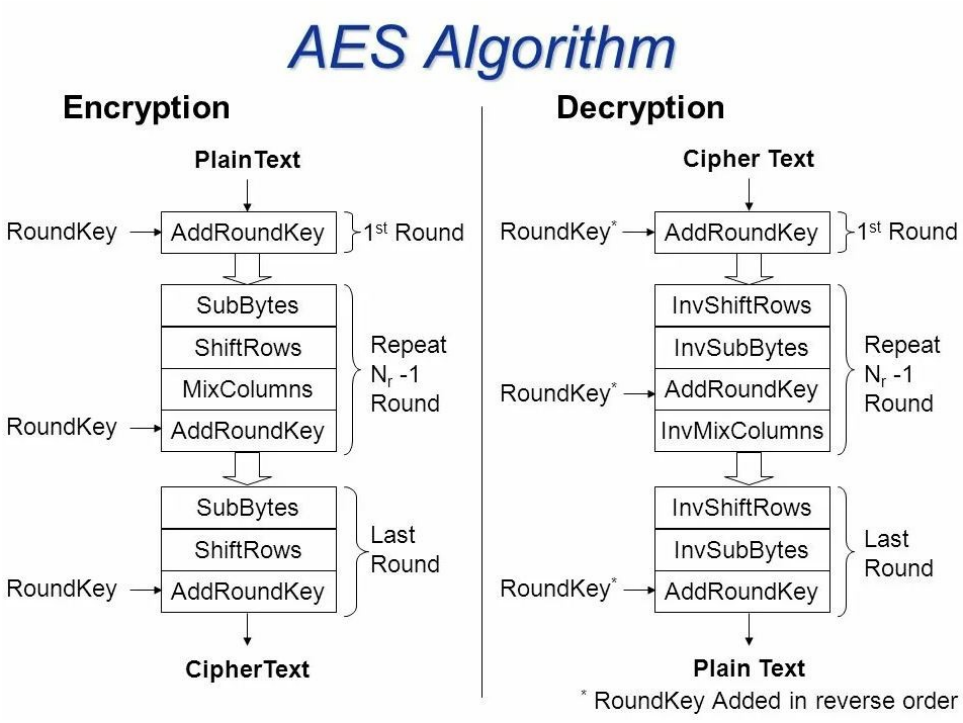

Figure 1: AES