2.9 Exercises

2.11. Point out unsymmetrical places between AES encryption and AES decryption.

2.12. Let \( K = 1234567890abcdef1234567890abcdef \) be an AES-128 encryption key, represented in hexadecimal. Calculate round key 1. \( K_1 = W[4,7] \).

2.13. Show that generating AES-128 round keys is equivalent to the following pseudo code.

```c
KeyExpansion (byte K[16], word W[44]) {
    int i;
    word temp;
    for (i = 0; i < 4; i++)
        W[i] = K[4*i, 4*i+3];
    for (i = 4; i < 44; i++) {
        temp = W[i-1];
        if ((i % mod 4 == 0))
            temp = SubWord(RotWord(temp)) ⊕ Rcon[i/4];
        W[i] = W[i-4] ⊕ temp;
    }
}
```

Here functions SubWord, RotWord, and Rcon are defined as follows. Let \( W = w_1w_2w_3w_4 \) be a word, where each \( w_j \) is a byte. Then

\[
\text{SubWord}(W) = S(w_1)S(w_2)S(w_3)S(w_4),
\]

\[
\text{RotWord}(W) = w_2w_3w_4w_1.
\]

Rcon[\( j \)] is a round constant, which is a word defined by \((RC[j], 0, 0, 0)\), with

\[
RC[j] = \begin{cases} 
02 ⊕ RC[j - 1], & \text{if } j > 1, \\
01, & \text{if } j = 1.
\end{cases}
\]

2.14. Verify the following elements in matrix \( mic(A) \) (see Equality 2.16): \( a'_{0,1} = 4d, a'_{0,2} = 9f, a'_{0,3} = d5 \).

2.15. Let \((a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}) = (8e, 4d, a1, bc)\) be the first row in the state matrix \( A \), compute the first column in matrix \( mic^{-1}(A) \); i.e., compute \((d0'_{0,0}, d0'_{1,0}, d0'_{2,0}, d0'_{3,0})\).

2.16. Let \( w_1 \) and \( w_2 \) be two 8-bit binary strings. Let \( A \) and \( B \) be two \( 4 \times 4 \) byte matrices, i.e., each element in the matrix is an 8-bit binary string. Prove the following equalities:

(a) \( \mathcal{M}(w_1 \oplus w_2) = \mathcal{M}(w_1) \oplus \mathcal{M}(w_2) \).

(b) \( mic^{-1}(A \oplus B) = mic^{-1}(A) \oplus mic^{-1}(B) \).

2.17. Let \( K = a01b2b3c4c5d67e8e99af0b0c0d1e1f \) be an AES-128 encryption key, represented in hexadecimal. Execute the first round of AES-128 on the plaintext block 0112233445566778899aabbccddeeff0. What is the state matrix \( A_2 \) after the first round?
2.18. Let $A$ be a state matrix. Show that $shr^{-1}$ and $sub^{-1}$ commute, i.e.,

$$shr^{-1}(sub^{-1}(C_i)) = sub^{-1}(shr^{-1}(C_i)).$$

2.19. Let $p(x)$ be a polynomial of degree $n$ in $GF(2^n)$. Show that

$$x^n \mod p(x) = p(x) - x^n.$$

2.20. Complete the verification of Equality 2.22.

2.21. Prove Equalities 2.29 and 2.30.

2.22. Following the construction algorithm of the AES S-Box, find the fourth element in the first row, i.e. $s_{0,3}$, in the S-Box $S$ and the fourth element in the first row, i.e. $s'_{0,3}$, in the inverse S-Box $S^{-1}$.

*2.23. Write a client-server program using socket API to implement AES-128 using an encryption key known to both sides, which is stored in a file. The client program takes a plaintext file and the encryption key file as input, encrypts the plaintext file using AES-128, and sends ciphertext blocks to the server program one block at a time. The server program uses the same encryption key from the encryption key file to decrypt the blocks it receives, one block at a time, and writes the plaintext blocks to a file.

2.24. RC5 is a block cipher with a Feistel structure. Its block size, the number of rounds, and key length may vary. In particular, RC5 takes $2w$-bit block as input, where $w \in \{16, 32, 64\}$; runs for $r$ rounds, where $r \in \{0, 1, \ldots, 255\}$; and uses $b$-byte keys, where $b \in \{0, 1, \ldots, 255\}$. It is customary to denote RC5-$w/r/b$ an RC5 encryption algorithm with parameters $w, r,$ and $b$. For example, RC5-32/12/16 takes a 64-bit block as input, runs for 12 rounds, and uses a 128-bit encryption key.

RC5 uses $r = 2r + 1$ subkeys of length $w$: $S_0, S_1, \ldots, S_{t-1}$, generated by the following algorithm. Let $K$ be a $b$-byte encryption key: $K_0, \ldots, K_{b-1}$, where $K_i$ is the $i$-th byte in $K$. Let $c$ be the smallest integer that is greater than or equal to $8b/32$. Let $L_0L_1\cdots L_{c-1}$ be a 32c-bit binary string, where each $L_i$ is a 32-bit binary string. Copy $K$ to $L$ from left to right. Pad the unoccupied locations in $L$ (if any) with 0. Let

$$S_0 \leftarrow P_w$$

For $i = 1$ to $t - 1$, let

$$S_i \leftarrow (S_{i-1} + Q_w) \mod 2^{32}$$

Let $i \leftarrow j \leftarrow A \leftarrow B \leftarrow 0$

Execute the following statements for $3 \times \max\{t, c\}$ times:

$$A \leftarrow S_i \leftarrow (S_i + A + B) \ll\ll 3$$
$$B \leftarrow L_j \leftarrow (L_j + A + B) \ll\ll (A + B)$$

$i \leftarrow (i + 1) \mod t$

$j \leftarrow (j + 1) \mod c$

where $P_w = \text{Odd}[(e - 2)2^w], Q_w = \text{Odd}[(\Phi - 1)2^w], \text{Odd}(x)$ denotes the odd number that is closest to $x$, $\Phi$ is the golden ratio $\frac{1+\sqrt{5}}{2}$, and $x\ll\ll y$ denotes the left-
circular-shift operation on $x$ for $y$ bits. In particular, $P_w$ and $Q_w$ are given below (in hexadecimal):

$$
\begin{array}{c|c|c|c}
  w & 16 & 32 & 64 \\
  P_w & b7e1 & b7e15163 & b7e151628ae2a6b \\
  Q_w & 9e37 & 9e3779b9 & 9e3779b97f4a7c15 \\
\end{array}
$$

Write a program to generate RC5 keys.

**2.25.** RC5 encryption and decryption are given below. Let $M = LR$, where $L$ and $R$ are, respectively, $w$-bit binary strings.

**RC5 encryption:**

\[
\begin{align*}
L & \leftarrow (L + S_0) \mod 2^{32} \\
R & \leftarrow (R + S_1) \mod 2^{32} \\
\text{For } i = 1 \text{ to } r, \text{ let} \\
   L & \leftarrow (((L \oplus R) \lll R) + S_{2i}) \mod 2^{32} \\
   R & \leftarrow (((L \oplus R) \lll L) + S_{2i+1}) \mod 2^{32}
\end{align*}
\]

**RC5 decryption:**

\[
\begin{align*}
\text{For } i = r \text{ down to } 1, \text{ let} \\
   R & \leftarrow (((R - S_{2i+1}) \ggg L) \oplus L) \\
   L & \leftarrow (((R - S_{2i}) \ggg R) \oplus R) \\
   L & \leftarrow (L - S_0) \mod 2^{32}
\end{align*}
\]

where $x \ggg y$ denotes the circular-right-shift operation on $x$ for $y$ bits.

(a) Prove the correctness of RC5 decryption.

(b) Write a program to implement RC5 encryption and decryption, using Exercise 2.24 to generate encryption keys. Here the plain text is an ASCII file, while the encryption keys and cipher text are stored in binary files. Note that RC5 follows the little-endian format to store binary strings (see Exercise 2.26).

**2.26.** Current computer architecture is based on 32-bit or 64-bit CPU. These Computers store information by words, and address memory locations by bytes. Thus, one word has four addressable units, whose relative addresses are 0, 1, 2, 3. Let $w = w_3w_2w_1w_0$ be a 4-byte binary string. We have two choices to store $w$ in a word: Store $w_3$ at relative address $i$, or store it at relative address $3 - i$, where $0 \leq i \leq 3$. The first choice is referred to as little-endian storage, and the latter big-endian storage.

In other words, if bytes in a 4-byte string are read from left to right, then in the little-endian storage, the first byte is stored in the location with the largest relative address in a word, the second byte is stored in the location with the second largest relative address, and so on; in the big-endian storage, the first byte is stored in the location with the smallest relative address in a word, the second byte is stored in the location with the second smallest relative address, and so on. Let $w = 08040201$
Table 2.10 Little-endian storage and big-endian storage of 08040201

<table>
<thead>
<tr>
<th>relative address</th>
<th>little-endian</th>
<th>big-endian</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>01</td>
<td>08</td>
</tr>
<tr>
<td>1</td>
<td>02</td>
<td>04</td>
</tr>
<tr>
<td>2</td>
<td>04</td>
<td>02</td>
</tr>
<tr>
<td>3</td>
<td>08</td>
<td>01</td>
</tr>
</tbody>
</table>

(hexadecimal), listed below shows how \( w \) is stored in the little-endian storage and in the big-endian storage:

For another example, on a 16-bit computer, the basic storage unit is a 2-byte memory unit, where each byte is addressable. Thus, to store UNIX we get UNIX in the big-endian storage, and we get NUXI in the little-endian storage.

Write a program that can exchange between the little-endian storage and the big-endian storage.

2.27. Show that in the CBC mode, any error occurred in one cipher block during transmission will affect the correctness of two plaintext blocks at the receiving side.

2.28. Suppose we are using AES under the CFB mode with \( s = 8 \). If a transmission error occurs in one cipher block, how many plaintext blocks will be affected at the receiving side?

2.29. For each of the following cipher-block modes, draw a block diagram for encryption and a block diagram for decryption.

(a) Electronic codebook mode (ECB).
(b) Cipher block chaining mode (CBC).
(c) Cipher feedback mode (CFB).
(d) Output feedback mode (OFB).
(e) Counter mode (CTR).

2.30. In Exercise 2.23 you have written a client-server program to encrypt and decrypt data using AES-128 under ECB. Rewrite this program using CBC, where the initial vector is a pseudorandom binary string generated by BBS.

2.31. Let \( M_1, \ldots, M_k \) be a sequence of plaintext blocks, where each \( M_i \) is \( \ell \)-bit long for \( 1 \leq i < k \), \( \ell \) is the input size of the underlying encryption algorithm \( E \), and \( M_k \) is \( q \)-bit long for \( q < \ell \). Define a ciphertext stealing mode (CTS) as follows, where \( C_0 \) is an \( \ell \)-bit initial vector and \( K \) an encryption key:

\[
C_i = E_K(M_i) \oplus C_{i-1}, \quad i = 1, \ldots, k-2,
\]
\[
C_k = p\delta_N(Z_{k-1}), \quad Z_{k-1} = E_K(Y_{k-1}), \quad Y_{k-1} = M_{k-1} \oplus C_{k-2},
\]
\[
C_{k-1} = E_K(Y_{k}), \quad Y_{k} = Z_{k-1} \oplus M_{0}0^{\ell-q}.
\]

(a) Describe how to decrypt \( C_{k-1} \) and \( C_k \), and prove the correctness of your decryption.
2.9 Exercises

(b) Draw a block diagram for encryption and a block diagram for decryption under CTS.

2.32. Alice proposes the following method to verify that she and Bob shares the same AES-128 key. Alice generates a 128-bit binary string \( r \) using BBS, encrypts \( r \), and sends the ciphertext block \( r_A = E_{K_A}(r) \) to Bob, where \( E \) is the AES-128 encryption algorithm and \( K_A \) is Alice's AES-128 encryption key. Bob decrypts \( r_A \) to get \( r' = D_{K_B}(r_A) \) and sends \( r' \) to Alice, where \( D \) is the AES-128 decryption algorithm and \( K_B \) is Bob's AES-128 encryption key. Alice checks whether \( r' = r \). If so, then \( K_A = K_B \). Is this protocol secure? Justify your answer.

2.33. Modify RC4 as follows: Shorten the array \( S \) from 256 cells in RC4 to 8 cells and replace each occurrence of 255 in RC4 with 7. This gives a simplified version of RC4. Let \( K = 0110010110000011 \) be an encryption key. Using this simplified RC4 to encrypt plaintext WITEHAT.

*2.34. Let \( M_1 = m_{11}m_{12} \cdots m_{1n} \) and \( M_2 = m_{21}m_{22} \cdots m_{2n} \) be two binary strings that are unknown to you, where each \( m_{ij} \) is a binary bit. However, you know

\[
M_1 \oplus M_2 = (m_{11} \oplus m_{21})(m_{12} \oplus m_{22}) \cdots (m_{1n} \oplus m_{2n}).
\]

Describe how you may be able to deduce \( M_1 \) and \( M_2 \).

2.35. Let \( p = 383 \) and \( q = 503 \). Show that \( p \equiv q \equiv 3 \pmod{4} \). Then let \( s = 101355 \). Write a program to implement BBS and produce the first 128 pseudorandom bits \( b_1, b_2, \ldots, b_{128} \).

2.36. The following result can be used to check whether a PRNG is sufficiently random: For any two positive integers \( x \) and \( y \), if they are selected uniformly at random, then the probability that \( \gcd(x, y) = 1 \) is equal to \( 6/\pi^2 \). Write a program to verify the randomness of the PRNG supported by the operating system of your machine.