IT361: Information Security LAB

Assignment IV

Course Instructor: Dr. Dibyendu Roy Marks: 20

Instructions: Clearly write your name and roll number on the top of your C code. Program file name should be YOUR ROLL NO.c

---

Write a single C code for the following problem. Let Alice and Bob are the two users and they have agreed on cyclic group $(\mathbb{Z}_p^*, \times \mod p)$ where $p = 2^{32} - 5$ with the generator $g = 2$.

1. Alice selects two large prime numbers p_a, q_a of 32 bits (input).
2. Alice computes $n_a = p_a q_a$ and $x_a = \phi(n_a)$.
3. Inside the code Alice finds e_a such that $\gcd(e_a, x_a) = 1$.
4. Inside the code Alice computes d_a such that $e_a d_a \equiv 1 \mod x_a$.
5. Alice makes (e_a, n_a) as his public key and (d_a, p_a, q_a) as his secret key.
6. Bob selects two large prime numbers p_b, q_b of 32 bits (input).
7. Bob computes $n_b = p_b q_b$ and $x_b = \phi(n_b)$.
8. Inside the code Bob finds e_b such that $\gcd(e_b, x_b) = 1$.
9. Inside the code Bob computes d_b such that $e_b d_b \equiv 1 \mod x_b$.
10. Bob makes (e_b, n_b) as his public key and (d_b, p_b, q_b) as his secret key.
11. Alice selects a random integer $0 < A < p$ and computes Diffie-Hellman (DH) public key $K_A = g^A$ (input A, print K_A).
12. Alice signs on his public key using his RSA secret key and sends signature S_A and DH public key K_A to Bob (print S_A).
13. Bob selects a random integer $0 < B < p$ and computes Diffie-Hellman (DH) public key $K_B = g^B$ (input B, print K_B).
14. Bob signs on his public key using his RSA secret key and sends signature S_B and DH public key K_B to Alice (print S_B).
15. Alice verifies the S_B if verification passes then he computes $K = g^{AB}$ (print K).
16. Bob verifies the S_A if verification passes then he computes $K = g^{AB}$ (print K).
17. Alice now selects a message of 64 bits and encrypts the message M_A using the below mentioned Symmetric-Enc algorithm (input M_A).
18. Print the above generated ciphertext C (print C).
19. Alice xor the least significant 32 bits of M_A with the most significant 32 bits of M_A and K to generate the MAC, MAC_A (print MAC_A).
20. Alice sends C, MAC_A to Bob.
21. Bob decrypts C using K and recovers M_A after that verifies the MAC_A.
22. Print the output from the MAC verification algorithm from Bob's side.

Symmetric-Enc:

It is a 16 round Feistel Network. For a 64-bit plaintext P and a 32-bit key K the encryption will produce a 64-bit ciphertext. The key-scheduling algorithm and the round function are described below.

1. Key scheduling algorithm will generate the 16 many 32-bit round keys K_i, $0 \leq i \leq 15$ as follows.
   - K_i is the left circular rotation on $(S_1(Y_0)||S_1(Y_1)||S_1(Y_2)||S_1(Y_3))$ for i times. Here $K = Y_0||Y_1||Y_2||Y_3$ and $\text{len}(Y_i) = 8$ bits. $S_1 : \{0,1\}^8 \rightarrow \{0,1\}^8$ is the S-box described below and $S_1(X)$ is computed according to the discussion in the class.
2. The round function f is defined as follows $f : \{0,1\}^{32} \times \{0,1\}^{32} \rightarrow \{0,1\}^{32}$.
   - $f(R_i, K_i) = S(R_i \oplus K_i)$
   - $S : \{0,1\}^{32} \rightarrow \{0,1\}^{32}$
   - $S(X) = (S_1^{-1}(x_0) \parallel S_1^{-1}(x_1) \parallel S_1^{-1}(x_2) \parallel S_1^{-1}(x_3))$ where $X = x_0 \parallel x_1 \parallel x_2 \parallel x_3$, each x_i is of 8 bits and $S_1 : \{0,1\}^8 \rightarrow \{0,1\}^8$ is the S-box described is described below. $S_1(X)$ is computed according to the discussion in the class.

S_1 :

0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16