Network Security Technology	Spring, 2024
Tutorial 1, Week 1 (Feb. 22)	LIU Zhen
Due Date: Feb. 26	

Questions:

1. **Please describe the KeyGen, Enc, Dec algorithms of the following ciphers, and give the key space, message space, and ciphertext space**:
   - Caesar's Cipher (50 points).
   - Simple Substitution Cipher (50 points).

Answer:

Caeasr's Cipher

$KeyGen(\square) \rightarrow k$:
Choose a random $k \leftarrow S_{key} \coloneqq \{0, \dots, 25\}$, and output k as the key.

$Enc(k, m) \rightarrow c$:
On input a key $k \in S_{key}$ and a message $m \in S_M \coloneqq \{0, \dots, 25\}$, output $c \leftarrow (m + k)\ mod\ 26$ as the ciphertext.

$Dec(k, c) \rightarrow m$:
On input a key $k \in S_{key}$ and a ciphertext $c \in S_C \coloneqq \{0, \dots, 25\}$, output $m \leftarrow (c + 26 - k)\ mod\ 26$ as the message.

$S_{key} \coloneqq \{0, \dots, 25\}$, $S_M \coloneqq \{0, \dots, 25\}$, $S_C \coloneqq \{0, \dots, 25\}$ is the key space, message space, and ciphertext space respectively, where in the message and ciphertext spaces, $0, \dots, 25$ corresponds to the letters $A, \dots, Z$ respectively.

Simple Substitution Cipher:

$KeyGen(\square) \rightarrow k$:
Choose a random permutation over letter list $\{A, B, \dots, Z\}$, say π, and output $k \coloneqq \pi$ as the key.

$Enc(k, m) \rightarrow c$:
On input a key k which is a permutation over letter list $\{A, B, \dots, Z\}$, say π, and a message $m \in S_M \coloneqq \{A, B, \dots, Z\}$, output $c \leftarrow \pi(m)$ as the ciphertext.

$Dec(k, c) \rightarrow m$:
On input a key k which is a permutation over letter list $\{A, B, \dots, Z\}$, say π, and a ciphertext $c \in S_C \coloneqq \{A, B, \dots, Z\}$, output $m \leftarrow \pi^{-1}(c)$ as the message, where π^{-1} denotes the inverse of permutation π.

The key space of Simple Substitution Cipher consists of all the permutations over letter list $\{A, B, \dots, Z\}$. The message space is $S_M \coloneqq \{A, B, \dots, Z\}$, and the ciphertext space $S_C \coloneqq \{A, B, \dots, Z\}$.

Questions:

1. **Please design a symmetric key encryption algorithm including $KeyGen$, encryption Enc, and decryption Dec based on the RC4 algorithm (using modules $RC4.Setup()$, $RC4.Initializtion()$ and $RC4.KeyStreamGeneration()$).**

Answer:

This is an open question. The following answer is just for reference.

$KeyGen(\square) \to k$:

Choose a uniform $k \in \{0,1\}^{128}$, and output k.

$Enc(k, m) \to c$:

On input a key $k \in \{0,1\}^{128}$ and a message m which has l bytes, say $m = m_1 m_2 \dots m_l$ where m_i is a byte, the algorithm proceeds as below:

(1) Parsing k as a 16-byte array, say key[18], call RC4.Setup and RC4.Initialization.
(2) Run Rc4.KeyStreamGeneration() 256 times.
(3) Read a ctr from the disk, if it does not exist on disk, set $ctr \leftarrow 0$.
(4) Call Rc4.KeyStreamGeneration() ctr times.
(5) For $i = 1, \dots, l$
    a. Call $\mathbf{KeyStreamByteSelected} \leftarrow \mathbf{Rc4.KeyStreamGeneration()}$
    b. Set $c_i \leftarrow \mathbf{KeyStreamByteSelected} \oplus m_i$
(6) Set $c \coloneqq (ctr, c_1, \dots, c_l)$
(7) Write $ctr \leftarrow ctr + l$ to disk.
(8) Output c as the ciphertext.

$Dec(k, c) \to m$:

On input a key $k \in \{0,1\}^{128}$ and a ciphertext $c \coloneqq (ctr, c_1, \dots, c_l)$, the algorithm proceeds as below:

(1) Parsing k as a 16-byte array, say key[18], call RC4.Setup and RC4.Initialization.
(2) Run Rc4.KeyStreamGeneration() 256 times.
(3) Call Rc4.KeyStreamGeneration() ctr times.
(4) For $i = 1, \dots, l$
    a. Call $\mathbf{KeyStreamByteSelected} \leftarrow \mathbf{Rc4.KeyStreamGeneration()}$
    b. Set $m_i \leftarrow \mathbf{KeyStreamByteSelected} \oplus c_i$
(5) Set $m \coloneqq (c_1, \dots, c_l)$
(6) Output m as the message.

Note: In the above description, a counter ctr is contained in ciphertext so that the receiver can synchronize its state with the sender. And the sender is responsible for maintaining the state. However, it is still difficult for the sender to make sure no repeated $\mathbf{KeyStream}$ is used,

and to achieve this, the sender has to store the counter in disk. This is a possible but not-perfect way.

2. **Suppose the key for a cipher is a l-bit binary string. To find a key by exhaustive key search, how many keys does an attacker need to test on average?**

Answer:

Without loss of generality, suppose the attacker test the key from $0 \ldots 0$ to $1 \ldots 1$ one-by-one.

The probability that the first one is the right key is $1/2^l$;

The probability that the second one is the right key is $1/2^l$;

$\cdots;$

The probability that the 2^l-th one is the right key is $1/2^l$.

On average, the number of keys that the attacker needs to test is

$1 \cdot \frac{1}{2^l} + 2 \cdot \frac{1}{2^l} + \cdots 2^l \cdot \frac{1}{2^l} = \frac{1}{2^l} \cdot (1 + 2 + \cdots 2^l) = \frac{1}{2^l} \cdot \left(\frac{(1+2^l) \cdot 2^l}{2}\right) = \frac{1}{2} + 2^{l-1} \approx 2^{l-1}$.

3. **Please analyze the DES-based symmetric encryption algorithm as shown in Figure 1, describe its key space, and compare its strengths and weaknesses to Triple DES.**

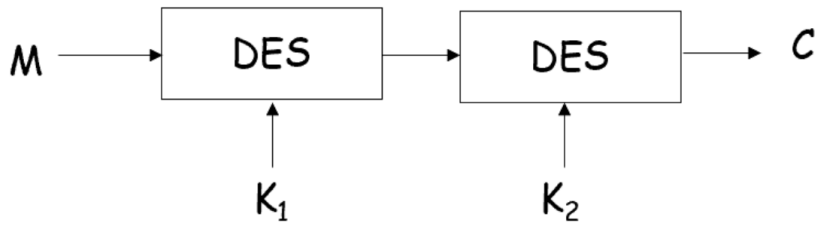

Figure 1: A DES-based algorithm.

Answer:

The key space is $\{0,1\}^{112}$.

The comparison with Triple DES (it is an open question, and the following answer is just for reference):

The above algorithm is weaker than Triple-DES against brute-force attack. In particular, given a (message, ciphertext) pair (m, c), an attacker can launch meet-in-the-middle attack as below:

(1) Run $DES.Enc(k_1, m)$ on m for each $k_1 \in \{0,1\}^{56}$. Suppose the attacker have sufficient storage store all these ciphertexts.

(2) For each $k_2 \in \{0,1\}^{56}$: run $DES.Dec(k_2, c)$, and compare the resulting message with the stored ciphertexts in Step (1).

Note that for such an attack, the size of the key space to be tested by the attacker is $2^{56} + 2^{56} = 2^{57}$, while for the Triple-DES, it is 2^{112}.

4. **As shown in Figure 2, what would happen in Triple DES if we replaced K_1 with K_3? How does this change compare to Triple DES, and what are the advantages and disadvantages?**

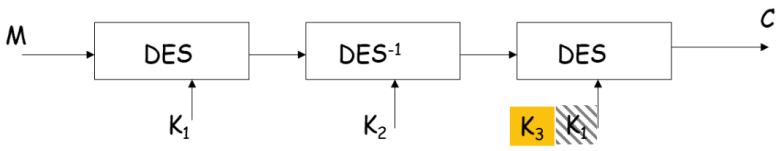

Figure 2: Triple DES.

Answer:

This is an open question, and the following answer is just for reference.

The modification makes the key space to be $\{0,1\}^{168}$.

When compared with the Tripel DES, this modification just makes the key become longer (which is a disadvantage), while does not improve security much, say the size of key space to be tested is $2^{112} + 2^{56}$ (which is only a small or tiny advantage).

Network Security Technology	Spring, 2024
Tutorial 3, Week 3 (Mar. 4)	LIU Zhen
Due Date: Mar. 11	

Questions:

1. **Given three callable algorithms of AES: $AES.KeyGen()$, $AES.Enc()$ and $AES.Dec()$. Please design an encryption algorithm to encrypt and decrypt data of arbitrary length using CBC mode.**

   $KeyGen(\lambda)$:

   $Enc(k, P)$:

   - Input:
   - Output:

   $Dec(k, C)$:

   - Input:
   - Output:

2. **Prove that: Any hash function that is collision resistant is second preimage resistant, and any hash function that is second preimage resistant is preimage resistant.**

1. Answer: (It is an open question, and the following answer is just for reference)

$KeyGen(\lambda) \to k$:

Call $k \leftarrow AES.KeyGen(\lambda)$, and output k as the key.

$Enc(k, P) \to C$:

On input a key k and a plaintext P, the algorithm proceeds as below:

1) Let $\ell := \frac{|P|}{128} + 2$ where $|P|$ is bit-length of P.
2) Extend P to $P' := P_0 \dots P_{\ell-1}$, where each P_i is 128-bit, $P_{\ell-1}$ is the binary representation of the bit-length of P (e.g., $0 \dots 01111$ denotes the bit-length of P is 15), $P_0 \dots P_{\ell-2}$ is the bit-string obtained by appending 0 to the end of bit-string of P.
3) Choose a uniform initial vector $IV \in \{0,1\}^{128}$
4) Compute $C_0 \leftarrow AES.Enc(k, IV \oplus P_0)$
5) For $i = 1, \dots, \ell - 1$, compute $C_i \leftarrow AES.Enc(k, C_{i-1} \oplus P_i)$
6) Output $C := (IV, C_0, \dots, C_{\ell-1})$ as the ciphertext.

$Dec(k, C) \to P$:

On input a key k and a ciphertext C, the algorithm proceeds as below:

1) Parse C to $C := (IV, C_0, \dots, C_{\ell-1})$ for some $\ell \geq 2$.
2) Compute $P_0 \leftarrow IV \oplus AES.Dec(k, C_0)$

3) For $i = 1, \ldots, \ell - 1$, compute $P_i \leftarrow C_{i-1} \oplus AES.Enc(k, C_i)$
4) Decode $P_{\ell-1}$ to be an integer L.
5) Output the first L bits of $P_0 P_1 \ldots P_{\ell-2}$ as the output plaintext.

2. **Answer:**

Please refer to the Sec.5.1.2 of "Introduction To Modern Cryptography" (2nd Edition)

Network Security Technology Spring, 2024
Tutorial 5, Week 5 (Mar. 19) LIU Zhen
Due Date: Mar. 19

Questions:

1. Determine whether 227 and 79 are relatively prime.
2. Find the multiplicative inverse of 79 mod 229.
3. Calculate $227^{54996213}$ mod 21 as efficient as possible.

Answer:

1. We can compute $\gcd(227,79)$ to determine whether they are relatively prime.
   As $\gcd(227,79) = \gcd(79,69) = \gcd(69,10) = 1$, we have that 227 and 79 are relatively prime.
2. We use Extended Euclidean Algorithm to compute the modular inverse.
   Note that
   $$229 = 2 \times 79 + 71$$
   $$79 = 1 \times 71 + 8$$
   $$71 = 8 \times 8 + 7$$
   $$8 = 1 \times 7 + 1$$
   We have $1 = 8 - 1 \times 7 = 8 - 1 \times (71 - 8 \times 8) = 9 \times 8 - 1 \times 71 = 9 \times (79 - 1 \times 71) - 1 \times 71 = 9 \times 79 - 10 \times 71 = 9 \times 79 - 10 \times (229 - 2 \times 79) = 29 \times 79 - 10 \times 229$, and then
   $$29 \times 79 \; mod \; 229 = 1$$
   This implies $29 \; mod \; 229 = 29$ is the multiplicative inverse of $79 \; mod \; 229$.

3. $227^{54996213} \; mod \; 21 = (227 \; mod \; 21)^{54996213} \; mod \; 21 = 17^{54996213} \; mod \; 21$
   $= 17^{54996213 \; mod \; \phi(21)} \; mod \; 21$ (since $\gcd(17,21) = 1$)
   $= 17^{54996213 \; mod \; 12} \; mod \; 21$ (since $\phi(21) = \phi(3) \cdot \phi(7) = 2 \cdot 6 = 12$)
   $= 17^9 \; mod \; 21 = 17^8 \cdot 17 \; mod \; 21$
   As
   $17^2 mod \; 21 = 16$
   $17^4 mod \; 21 = 16^2 mod \; 21 = 4$
   $17^8 mod \; 21 = 4^2 mod \; 21 = 16$
   We have
   $17^8 \cdot 17 \; mod \; 21 = 16 \cdot 17 \; mod \; 21 = 20$

Network Security Technology Spring, 2024
Tutorial 5, Week 5 (Mar. 19) LIU Zhen
Due Date: Mar. 19

Problem: Write a short survey, including attempts proposed to achieve practical and safe use of RSA encryption, as well as current industry applications of RSA-based public key encryption schemes. Provide specific algorithms, advantages, disadvantages, problems solved, and existing issues for each of these schemes.

This is an open question, and we donot provide any standard answer.

20240326第六次作业

已发布 编辑

作业问题：对照PKE的CCA模型，说明El Gamal 加密方案不是CCA安全的。
作业提交格式为PDF，命名为“姓名_学号”。请大家认真作答，谢谢！

Answer: (Note that the following is a referenced answer, and there may be other correct answers.)

In the Challenge Phase, the attacker submits two messages $m_0, m_1 \in G\ s.t. m_0 \neq m_1$, and obtains a ciphertext c^* such that $c^* := (c_1^*, c_2^*) = (g^y, h^y \cdot m_b)$ for some random $y \in Z_q$ and some random $b \in \{0,1\}$, where $g \in G$ is a group generator and $h := g^x$ for some $x \in Z_q$ is a part of the public key.

Then at the Probing Phase 2, the attacker can make use of the query to the Decryption oracle to find the value of b, as below:

(1) Choose a $y' \in Z_q$, then make a ciphertext $c' := (c_1', c_2')$ by setting $c_1' = c_1^* \cdot g^{y'}$, $c_2' = h^{y'} \cdot c_2^*$.
(2) Send c' to the challenger to query the decryption oracle, and obtain a message m'. (As $c' \neq c^*$, the challenger will decrypt this ciphertext and send back the message.)
(3) Note that actually $c_1' = g^{y+y'}, c_2' = h^{y'+y} \cdot m_b$, m' is actually m_b. The attacker can output the right value of b by comparing m' with m_0 and m_1.

20240326第七次作业

已发布 编辑

作业内容：

检索和阅读文献，写一篇报告。包括国际/行业标准中推荐使用的RSA签名的具体算法。

作业提交格式为PDF，命名为“姓名_学号”。请大家认真作答，谢谢！

This is an open question, and we donot provide any standard answer.

Network Security Technology Spring, 2024
Tutorial 7, Week 7 (Apr. 1) LIU Zhen
Due Date: Apr. 8

Questions:

1. Given a public key $pk = (N, e)$ and a specific message $m \in Z_N^*$, can you force a valid signature $\sigma \in Z_N^*$? (i.e., $Verify(pk, m, \sigma) = 1$)

   Hint: you are allowed to launch chosen-message attack, i.e., have access to the $Sign(.)$ oracle described in the security model for signature scheme.

Answer: (Note that the following is only a reference answer, and there may be other answers)

(1) The attacker chooses a message $m_1 \in Z_N^*$ and set $m_2 = m \cdot m_1 \, mod \, N$.
(2) The attacker makes Sign()-query on input m_2 and obtains a signature σ_2.
(3) The attacker makes Sign()-query on input m_1 and obtains a signature σ_1.
(4) The attacker outputs $\sigma := \sigma_2 \cdot \sigma_1^{-1} \, mod \, N$ as its forgery with respect to message m.

Note that it hold that $\sigma_1^e = m_1 \, mod \, N, \sigma_2^e = m_2 \, mod \, N$, we have $(\sigma_2 \cdot \sigma_1^{-1})^e mod \, N = \sigma_2^e \cdot \sigma_1^{-e} \, mod \, N = m_2 \cdot m_1^{-1} \, mod \, N = m \, od \, N$, which means $\sigma := \sigma_2 \cdot \sigma_1^{-1}$ satisfy $Verify(pk, m, \sigma) = 1$.