MAS433 Cryptography: Tutorial 3
Block Cipher, Stream Cipher, Hash Function, MAC
Solution

Problem 1. Modes of operation of block cipher

1.1. Consider those five modes of operation: ECB, CBC, CFB, OFB, CTR. If there is error in a ciphertext block (it is not the last ciphertext block), how many plaintext blocks would be decrypted wrongly?

Solution.

ECB mode: One plaintext block is decrypted wrongly
CBC mode: Two plaintext blocks are decrypted wrongly
CFB mode: Two plaintext blocks are decrypted wrongly
OFB mode: One plaintext block is decrypted wrongly
CTR mode: One plaintext block is decrypted wrongly
(Remarks: This tutorial question is on the error propagation of each mode of operation)

1.2. A mode of operation operates as follows: $C_0 = IV$, $C_i = E_K(P_i) \oplus C_{i-1}$ for $i \geq 1$. Is this mode of operation better than ECB mode?

Solution.

From $C_i = E_K(P_i) \oplus C_{i-1}$, we obtain

$$C_i \oplus C_{i-1} = E_K(P_i)\,. \tag{1}$$

Let $C'_i = C_i \oplus C_{i-1}$, from (1), we obtain:

$$C'_i = E_K(P_i)\,.$$

So this mode of operation is almost equivalent to the ECB mode.

1.3. A mode of operation operates as follows: $P_0 = IV_1$, $C_0 = IV_2$, $C_i = E_K(P_i \oplus C_{i-1}) \oplus P_{i-1}$ for $i \geq 1$. How to decrypt the ciphertext? If there is error in a ciphertext block (it is not the last ciphertext block), how many plaintext blocks would be decrypted wrongly?

Solution.

Decryption: $P_i = E_K^{-1}(C_i \oplus P_{i-1}) \oplus C_{i-1}$

Error propagation: If there is error in C_i, then all the P_{i+j} ($j \geq 0$) would be decrypted wrongly.
(Remarks: You may need to draw the decryption diagram to see how the error propagates.)

Problem 2. Meet-in-the-middle attack on block cipher
Apply the meet-in-the-middle attack to three-key Triple-DES. How to reduce the complexity of the attack to 2^{112} DES operations, and 2^{56} memory (we consider each unit of memory as 128 bits here) ?
Solution Outline.
Denote the encryption of three-key Triple-DES as $C = E_{K_3}(E_{K_2}^{-1}(E_{K_1}(P)))$, where K_1, K_2, K_3 are independent 56-bit keys.
Step 1. From a plaintext-ciphertext pair (P, C), we can try all the possible keys of K_3 to decrypt C. For a guessed value $K_{3,i} = i$, we denote the decrytped value as $A_i = E_i^{-1}(C)$. We obtain a table T with each element being (i, A_i) for $0 \leq i \leq 2^{56} - 1$.
Step 2. Sort the table T according to the values of A_i. Denote the table after sorting as T'. Each element in the table T' is denoted as $T'[i] = (A_t, t)$, and the value of A_t increases as the value of i increases. (The sorting is needed for fast tale lookup: given an A_x, finding the value of x. The cost of sorting n elments is about $O(n\log n)$.)
Step 3. Then we try to guess all the possible values of (K_1, K_2). For each guessed value of $(K_{1,x}, K_{2,y}) = (x, y)$, compute $E_{K_{2,y}}^{-1}(E_{K_{1,x}}(P))$. If $E_{K_{2,y}}^{-1}(E_{K_{1,x}}(P))$ is in table T', i.e., if $E_{K_{2,y}}^{-1}(E_{K_{1,x}}(P)) = A_z$ with (A_z, z) being an element in Table T', then we know that $(K_{1,x}, K_{2,y}, K_{3,z}) = (x, y, z)$ may be the key being used in the Triple-DES. There are $2^{112} \times 2^{56} \times 2^{-64} = 2^{104}$ possible values of (K_1, K_2, K_3) being left at the end of this step.
Step 4. For all the 2^{104} possible keys being found in Step 3, we do not store those values. We test those keys immediately with another two plaintext-ciphertext pairs (P', Q') and (P'', Q''), and determine the correct key.
Computation complexity: The computational complexity of Step 1 is 2^{56} DES decryptions. In average, we need to search 2^{111} possible values of (K_1, K_2) for finding a correct key, thus the computational complexity of Step 3 is about $2 \times 2^{111} = 2^{112}$ DES encryptions, and that of Step 4 is $2^{103} + 2^{104-64} \approx 2^{103}$ DES encryptions. The total complexity is about $2^{56} + 2^{112} + 2^{103} \approx 2^{112}$ DES encryptions. (The sorting complexity in Step 2 is negligible comparing to the total complexity.)
Data complexity: The data complexity of Step 1 is 2^{56} memory, with each memory unit being 64+56=120 bits.

Problem 3. Solve over-defined algebraic equations
3.1. In a system of algebraic equations over GF(2), if there are n binary variables, and the highest degree of monomials is d, then what is the maxi-

mal number of different monomials in this system of equations? How many such equations are needed in order to solve this system of equations through linearization? (Hint: for a monomial over GF(2), $x^2y^3z^8 = xyz$, i.e., the degree is 3)

Solution Outline.

For n binary variables, there are $\sum_{i=1}^{d} \binom{n}{i}$ possible monomials with degree not greater than d. To linearize this system of equations, about $\sum_{i=1}^{d} \binom{n}{i}$ equations are needed.

3.2. There are several nonlinear equations over GF(7),

$$\begin{aligned} x_1 + x_2 + x_1x_2 + x_2x_3 &= 1 \\ x_1 + x_3 + x_1x_2 &= 0 \\ x_1 + x_2x_3 &= 6 \\ x_2 + x_3 + x_1x_2 &= 1 \\ x_2 + x_3 &= 3 \end{aligned}$$

How to linearize the above over-defined equations?

Solution.

Let $y_i = x_i$ for $1 \le i \le 3$. Let $y_4 = x_1x_2$, $y_5 = x_2x_3$. After substitution, the above equations become:

$$\begin{aligned} y_1 + y_2 + y_4 + y_5 &= 1 \\ y_1 + y_3 + y_4 &= 0 \\ y_1 + y_5 &= 6 \\ y_2 + y_3 + y_4 &= 1 \\ y_2 + y_3 &= 3 \end{aligned}$$

(If we solve the above system of linear equations, the solution is: $x_1 = 3$, $x_2 = 4$, $x_3 = 6$.)

Problem 4. Stream cipher

4.1. The A5/1 stream cipher consists of a 64-bit state. If we generate a long keystream from A5/1 using a key and an IV, what would be the estimated period of the keystream? (Hint: Is the state of A5/1 updated in an invertible way? Why? Can we apply the birthday paradox here?)

Solution Outline. The state of A5/1 is updated (through the clock control & clocking of the registers) in an non-invertible way. The reason is that for a given state, we are not certain which registers are clocked in the previous step.

As we learned in the Rho method (birthday attack), for a noninvertible function f with output length n, if we compute $x_{i+1} = f(x_i)$, then the sequence x_i $(i = 0, 1, 2, 3, ...)$ would form a cycle with period about $O(2^{n/2})$ for some

$i \geq O(2^{n/2})$.
Denote the state of A5/1 at time t as S_t, and the state update function as F. Then we see that $S_{t+1} = F(S_t)$. Thus for $t \geq O(2^{64/2})$, we would expect that the state of A5/1 would form a cycle with period about $t \geq O(2^{64/2})$. Since the same keystream bits would be generated from the same stream cipher states, the keystream would form a cycle with period about $O(2^{32})$ for $t \geq O(2^{32})$.
(Remarks: The above analysis gives only a rough estimation of the period of the keystream of A5/1.)

4.2. For the RC4 stream cipher, will two elements in the table S become identical? Why? How many possible values a table in RC4 can take?
Solution Outline. Two elements in the table S of RC4 would never be identical. The reason is that those 256 elements are initialized to different values (i.e., $S[i] = i$ for $0 \leq i \leq 255$), and the table is updated only through swapping the elements.
A table in RC4 can take 256! possible values.
(Remarks: Be precisely, a table in RC4 takes about 256! possible values.)

Problem 5. Birthday Attack
5.1. Randomly select four persons. What is the probability that two of these four persons have the same birthday?
Solution.
$p = 1 - (1 - 1/365)(1 - 2/365)(1 - 3/365) = 0.0164$

5.2. Randomly select 32 persons. What is the probability that two of these 32 persons have the same birthday?
Solution.
$p = 1 - e^{\frac{-Q(Q-1)}{2M}} = 1 - e^{\frac{-32\times(32-1)}{2\times365}} = 0.743$
(Remarks: The accurate probability is 0.75335, close to the estimated probability.)

Problem 6. Hash Function
6.1. For a hash function based on a modified Merkle-Damgard construction in which only '0' bits are padded to the end of the message, how to find a collision with low complexity?
Hint. Consider the ambiguity of the last partial message block.

6.2. For SHA-1, the message length is formatted as a 64-bit word, and it is padded to the message. If the message length is 200 bits, how many compression function operations are needed to compress this message? How many compression function operations are needed if the message lengthens are 0 bit, 1 bit, 447 bits, 448 bits, 511 bits, 512 bits, 960 bits?

Solution.
0-bit message: 1 compression function operation
1-bit message: 1 compression function operation
447-bit message: 1 compression function operation
448-bit message: 2 compression function operations
511-bit message: 2 compression function operations
512-bit message: 2 compression function operations
960-bit message: 3 compression function operations

6.3. Here is a method to protect the login passwords on a computer: when we create a user account in a computer system (Windows, Linux or UNIX), the login password of a user is hashed, and the hashed value (instead of the password) is stored on the computer. When a user login to the system, the computer hashes the password, and compares the hashed value with the stored hash value. With this approach, it becomes difficult to recover the passwords if an attacker (or even administrator) gains access to the computer.
6.3.1. What property of hash function is used in this password protection scheme?
Solution.
primage resistance

6.3.2. Suppose that SHA-1 is used in this password protection scheme, what is the complexity to recover an equivalent password if the hashed value of an password is known to an attacker? (Here the equivalent password means that this password is hashed to the same value as the hash value being stored on a computer, but this password may or may not be the same as the original password.)
Solution outline.
Depending on the length and randomness of the password. For a weak password (such as a password being chosen as a english word), the complexity to recover the password is low. If a password is sufficiently long and random, the complexity is about 2^{160}.
6.4. In a hash function based on Merkle-Damgard construction, the compression function is given as $H_i = E_{m_i}(H_{i-1})$, where $E_k()$ denotes the encryption of an ideal block cipher with n-bit block size.
6.4.1. What is the complexity to find a preimage of this hash function? (Hint: meet-in-the-middle attack)
Solution Outline.
The complexity is $2^{n/2}$.

6.4.2. What is the complexity to find a second-preimage of this hash function?
Solution Outline.

The complexity is $2^{n/2}$.

6.4.3. What is the complexity to find a collision of this hash function?
Solution Outline.
The complexity is $2^{n/2}$.

6.5. In a hash function based on a modified Merkle-Damgard construction in which the IV is not fixed (i.e., an user is allowed to set the value of IV arbitrarily), the compression function is given as $H_i = E_{m_i}(H_{i-1})$, where $E_k()$ represents the encryption of an ideal block cipher with n-bit block size.
6.5.1. What is the complexity to find a preimage of this hash function?
Solution Outline.
The complexity is 1.

6.5.2. What is the complexity to find a second-preimage of this hash function?
Solution Outline.
The complexity is 1.

6.5.3. What is the complexity to find a collision of this hash function?
Solution Outline.
The complexity is 2.

6.6. In a hash function based on a modified Merkle-Damgard construction in which the IV is not fixed (i.e., an user is allowed set the value of IV arbitrarily), the compression function uses the Davies-Meyer structure with an ideal block cipher with n-bit block size.
6.6.1. What is the complexity to find a preimage of this hash function?
Solution Outline.
The complexity is 2^n.

6.6.2. What is the complexity to find a second-preimage of this hash function?
Solution Outline.
The complexity is 2^n.

6.6.3. What is the complexity to find a collision of this hash function?
Solution Outline.
The complexity is $2^{n/2}$.

Problem 7. Message Authentication Code
7.1. After the CBC-encryption, denote the last ciphertext block as C_n. An MAC algorithm generates the authentication tag as: $t = E_{K_A}(C_n)$, where K_A is the secret key for MAC algorithm, and it is different from the encryp-

tion key. How to attack this MAC algorithm?
Solution Outline. In this scheme, encryption is applied, but the authentication is only applied to protect the last ciphertext block. When the ciphertext is sent together with this message authentication tag, the ciphertext blocks (except for the last one) can be modified arbitrarily (if we consider that ciphertext stealing technique is not used here).

7.2. Denote the message as $M = m_1 \| m_2 \| m_3 \| \cdots \| m_n$. An MAC algorithm generates the authentication tag as: $t = E_K(m_1) \oplus E_K(m_2) \oplus \cdots \oplus E_K(m_n)$. How to attack this MAC algorithm?
Solution Outline. This authentication scheme does not protec the order of the message blocks.

7.3. CMAC
7.3.1. What are the differences between CBC encryption and CBC-MAC?
Solution Outline. As explained in the lecture slides.

7.3.2. Why the CBC-MAC is insecure?
Solution Outline. As explained in the lecture slides.

7.3.3. What are the differences between CMAC and CBC-MAC?
Solution Outline. As explained in the lecture slides.

7.3.4. In CMAC, why the value of $K1$ is not simply set as $E_k(0)$?
Solution Outline. Suppose that a message consists of two blocks with '0' bits. If the value of $K1$ is simply set as $E_k(0)$, the message authentication tag gives $K1$.

7.4. HMAC
7.4.1. Consider an MAC algorithm constructed from a hash function using the key-prefix method. How to attack this MAC algorithm?
Solution Outline. As explained in the lecture slides.
7.4.2. HMAC-SHA-1 is the HMAC constructed from SHA-1. It is now widely used for secure internet communication. For HMAC-SHA-1, how many compression function operations are needed to generate the authentication tag of an 800-bit message? How many compression function operations are needed if the message lengths are 0 bit, 960 bits?
Solution.
0-bit message: 4 compression function operations
800-bit message: 5 compression function operations
960-bit message: 6 compression function operations

7.5. Unconditionally secure MAC
In an unconditionally secure MAC, the message is given as $m_1, m_2, m_3, ...m_n$,

the one-time key is given as $k_0, k_1, k_2, ...k_n$, where $m_i \in GF(97)$, and $k_i \in GF(97)$. The message authentication tag is generated as

$$t = (k_0 + \sum_{i=1}^{n} m_i \times k_i) \bmod 97\,.$$

7.5.1. The above unconditionally secure MAC is used for authenticating message. What is the probability that a message can be modified without being detected?
Solution.
$p = \frac{1}{97}$

7.5.2. If k_0 is not randomly generated, and its value is '12' or '67' with equal probability. For a short message with only one block m_1, what is the probability that a message can be modified without being detected?
Solution.
$p = \frac{1}{2}$

7.5.3. If the message is given as $m_1, m_2, m_3, ...m_n$, the one-time key is given as $k_0, k_1, k_2, ...k_n$, where $0 \leq m_i < 194$, and $0 \leq k_i < 194$. The message authentication tag is generated as $t = (k_0 + \sum_{i=1}^{n} m_i \times k_i) \bmod 194$. Is there efficient attack against this authentication scheme?
Solution outline.
Notice that $(97 \times k_x \bmod 194)$ is 0 or 97 with probability 1/2. If an attacker knows that $m_i = 97$, the attacker can change its value to 0, and this modification will be detected with probability 0.5.

Problem 8. Initialization Vector
8.1 How to choose the initialization vectors for CBC mode, CFB mode, OFB mode, CTR mode, synchronous stream cipher, asynchronous stream cipher, hash function, CBC-MAC and CMAC?
Solution outline.
CBC mode: use random IVs
CFB mode: different IVs should be used for the same key
OFB mode: different IVs should be used for the same key
CTR mode: different IVs should be used for the same key
synchronous stream cipher: different IVs should be used for the same key
asynchronous sream cipher: different IVs should be used for the same key
hash function: IV is set as a (fixed) constant
CBC-MAC: IV is set as 0
CMAC: IV is set as 0

8.2 How would the security be affected if identical IVs are used with the same secret key in the cryptosystems in Problem 8.1?

Solution outline.
CBC mode: There is no randomness being introduced to the encryption of the first block, thus the first block can be attacked in the same way as the attack on ECB mode. If there is a lot of redundancy in the first plaintext block, then an attacker can collect enough plaintext-ciphertext pairs of the first block, then recover the first plaintext blocks of other ciphertexts by comparing the ciphertext blocks with the plaintext-ciphertext pairs being collected. (We call it dictionary attack.)
CFB mode: The value of $E_K(IV)$ can be recovered once we know the value of the first block of a plaintext. Then all the first blocks of the plaintexts can be recovered.
OFB mode: the same keystream is used to encrypt more than one message, insecure
CTR mode: the same keystream is used to encrypt more than one message, insecure
synchronous stream cipher: the same keystream is used to encrypt more than one message, insecure
asynchronous stream cipher: insecure, similar to the CFB mode
hash function: IV is set as a (fixed) constant (and no secret key)
CBC-MAC: IV is set as 0
CMAC: IV is set as 0