Quiz 3

111550113 謝詠晴

Problem 1

† Data compression is often used in data storage and transmission. Suppose you want to use data compression in conjunction with encryption. Does it make more sense to:

- ☑ Compress then encrypt.
- ☑ Encrypt then compress.
- ☑ The order does not matter – either one is fine.
- ☐ The order does not matter – neither one will compress the data.

1. Generally, it is more common to compress before encrypting as to have better performance. Plaintext has a large redudancy so compression can reduce the size, making less usage of random numbers. Also, compression process is more suitable for plaintext rather than random bits.

2. However, there's a latest way is to encrypt before compressing. The advantages are avoiding compression attacks or preserving encryption metadata.

3. Overall, it depends on the scenario, if efficiency is the primary concern, compressing first may
4. be a better choice, such as in communication cases ; On the other hand, if security seems more vital, encrypting first may be a suitable choice, such as in CS perspective.

Problem 2

† Let G: $\{0,1\}^s \to \{0,1\}^n$ be a secure PRG. Which of the following is a secure PRG:

- ☐ G'(k) = G(k) ‖ G(k)
- ☑ G'(k) = G(k ⊕ 1^s)
- ☐ G'(k) = G(0)
- ☐ G'(k) = G(1)
- ☐ G'(k) = G(k) ‖ 0
- ☑ G'(k_1, k_2) = G(k_1) ‖ G(k_2)
- ☑ G'(k) = reverse(G(k))
- ☑ G'(k) = $rotation_n$(G(k))

[key point : The output of G'(k) still need to be random.]

1. G'(k) = G(k) ‖ G(k) , since it concatenate the two same string G(k).
   The output may look like this for example : 0101 0101
   Therefore, the output won't be random, it'll have a obvious pattern.

2. G'(k) = G(k ⊕ 1^s)
   Since after k XOR 1^s, it's still the input of G. function, so the output will still be generated randomly.
   ex: k = 010, k⊕1^s = 101 ⇒ 此操作不會造成規律性易辨認的pattern.

3. 、4. Since output is G(0) or G(1) everytime, it is deterministic and not random.

5. $G'(k) = G(k) \,||\, 0$, since the output is always concatenate with 0 at the end.
The attacker can easily distinguish it from other random output and the output isn't fully random.

ex: let A(x): If last bit is 0 output 1, otherwise 0 $\Rightarrow \left|Pr[A(G(k))=1] - Pr[A(r)=1]\right| = |0-\frac{1}{2}| = \frac{1}{2}$

6. Concatenating two random output, the output of G'(k) is still random.

7. After reversing G(k), which is a random output, the result is still random.

8. It's similar to reversing, after rotating a random output, the result is still random.

Problem 3

Now, suppose the bank wants to split k into three pieces p_1, p_2, p_3 so that any two of the pieces enable decryption using k. This ensures that even if one executive is out sick, decryption can still succeed. To do so the bank generates two random pairs (k_1, k_1') and (k_2, k_2') as in the previous paragraph so that $k_1 \oplus k_1' = k_2 \oplus k_2' = k$. How should the bank assign pieces so that any two pieces enable decryption using k, but no single piece can decrypt?

- ☐ $p_1 = (k_1, k_2)$, $p_2 = (k_1, k_2)$, $p_3 = (k_2')$
- ☐ $p_1 = (k_1, k_2)$, $p_2 = (k_1', k_2')$, $p_3 = (k_2')$
- ☑ $p_1 = (k_1, k_2)$, $p_2 = (k_1', k_2)$, $p_3 = (k_2')$
- ☐ $p_1 = (k_1, k_2)$, $p_2 = (k_2, k_2')$, $p_3 = (k_2')$
- ☐ $p_1 = (k_1, k_2)$, $p_2 = (k_1')$, $p_3 = (k_2')$

∵ $k_1 \oplus k_1' = k_2 \oplus k_2' = k$, Only when (k_1, k_1') or (k_2, k_2') appears at the same time, we can know what k is. And the problem requirement is any two pieces can decrypt.

1. If two pieces are $P_1(k_1, k_2)$ 、$P_2(k_1, k_2)$, There's no k_1' or k_2' so it can't decrypt.

2. If two pieces are $P_2(k_1', k_2')$ 、$P_3(k_2')$ There's no k_1 or k_2 so it can't decrypt.

3. $\begin{cases} P_1(k_1, k_2) \text{ 、} P_2(k_1', k_2) \Rightarrow (k_1, k_1') \\ P_1(k_1, k_2) \text{ 、} P_3(k_2') \Rightarrow (k_2, k_2') \\ P_2(k_1', k_2) \text{ 、} P_3(k_2') \Rightarrow (k_2, k_2') \end{cases}$ $\Rightarrow$ Any two pieces can decrypt.

4. When only P_2 exists, it can decrypt on its own since it has k_2 & k_2'. However, the requirement is no single piece can decrypt.

5. If two pieces are $P_2(k_1')$ 、$P_3(k_2')$, There's no k_1 or k_2 to decrypt.

Problem 4

Let M = C = K = { 0, 1, 2, . . . , 255 } and consider the following cipher defined over (K, M, C):
E(k, m) = m + k (mod 256); D(k, c) = c - k (mod 256)
Does this cipher has perfect secrecy?

- ☐ No, there is a simple attack on this cipher.
- ☑ Yes
- ☐ No, only the One Time Pad has perfect secrecy.

$\forall\ m, c \in M, C$

if $E(k, m) = c$, $m + k \pmod{256} = c$

$k = c - m \pmod{256}$

num. of $\{k \in K,\ E(k, m) = c\} = 1$

$$\therefore \Pr_{k}[E(k, m) = c] = \frac{\text{num. of } \{k \in K,\ E(k, m) = c\}}{|K|} = \frac{1}{256}$$

It's difficult to crack c, so this cipher has perfect key.

Problem 5

† Let (E, D) be a (one-time) semantically secure cipher where the message and ciphertext space is $\{0,1\}^n$. Which of the following encryption schemes are (one-time) semantically secure?

- ☐ E'(k, m) = $E(0^n, m)$
- ☑ E'((k, k'), m) = E(k, m) ‖ E(k', m)
- ☐ E'(k, m) = E(k, m) ‖ MSB(m)
- ☑ E'(k, m) = 0 ‖ E(k, m) (i.e. prepend 0 to the ciphertext)
- ☐ E'(k, m) = E(k, m) ‖ k
- ☑ E'(k, m) = reverse(E(k, m))
- ☑ E'(k, m) = $rotation_n$(E(k, m))

1. Attacker will ask for the encryption of 0^n and 1^n. Since k is 0^n, attacker can easily distinguish between encryption of 0^n and encryption of 1^n. The secret key will then be found.
2. If attack on E', it means an attack on E, and E is semantically secure. Therefore, E' is also semantically secure.
3. Attacker will ask for encryption of 0^n and 10^{n-1}, thus can distinguish EXP(0) from EXP(1), which is not semantically secure.
4. Although prepending a 0 to the ciphertext, an attack to E' still means an attack to E, so it's still semantically secure.
5. If directly concatenate the key with the ciphertext, then attacker can use this key to decrypt.
6. After reversing, the attack is still on E so it's semantically secure.
7. After rotation, the attack is still on E so it's semantically secure.

Problem 6

Suppose you are told that the one time pad encryption of the message "attack at dawn" is 6c73d5240a948c86981bc294814d (the plaintext letters are encoded as 8-bit ASCII and the given ciphertext is written in hex). What would be the one time pad encryption of the message "defend at noon" under the same OTP key?

(PT)

"attack at dawn" ASCII DEC. → 97, 116, 116, 97, 98, 107, 32, 97, 116, 32, 100, 97, 119, 110.

DEC:

97 116 116 97 98 107 32 97 116
01100001 01110100 01110100 01100001 01100010 01101011 00100000 01100001 01110100

32 100 97 119 110
00100000 01100100 01100001 01110111 01101110

(CT)

6c73d5240a948c86981bc294814d

6 c 7 3 d 5 2 4 0 a 9 4 8 c 8 6 9 8
01101100 01110011 11010101 00100100 00001010 10010100 10001100 10000110 10011000

1 b c 2 9 4 8 1 4 d
00011011 11000010 10010100 10000001 01001101

⇓ PT XOR CT = KEY

(KEY)

00001101 00000111 10100001 01000101 01101000 11111111 10101100 11100111 11101100
00111011 10100110 11110101 11110110 00100011

"defend at noon", the representation of 8-bit ASCII code is:

(100, 101, 102, 101, 110, 100, 32, 97, 116, 32, 110, 111, 111, 110)

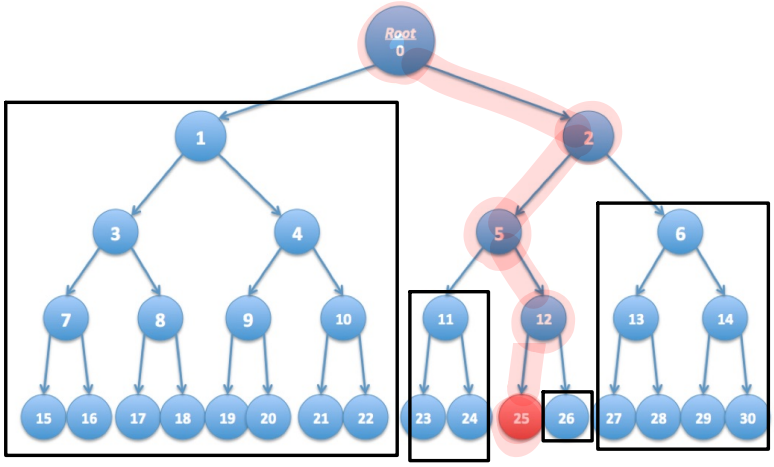
01100100 01100101 01100110 01100101 01101110 01100100 00100000 01100001 01110100
00100000 01101110 01101111 01101111 01101110

The XOR result: (using online calculator)

01101001 01100010 11000111 00100000 00000110 10011011 10001100 10000110 10011000
00011011 11001000 10011010 10011001 01001101

⇒ 69b2c720079b8c8698 1bc89a994d #

Problem 7



- ☐ 21
- ☐ 17
- ☐ 5
- ☑ 26
- ☑ 6
- ☑ 1
- ☑ 11
- ☐ 24

We need to choose four keys and their children in the tree include all numbers (0~30) except nodes that are on the path from root to 25. (0, 2, 5, 12, 25)

① We can choose 1 first since 25 is in the right side of the root while 1 is in the left.

② We must choose 26, since any number which children include 26 is the node going through path to 25, thus must include 25 too.

③ Then it's obvious the remaining two choices are 6 and 11.

After choosing these four keys, we can ensure that every player other than player 25 can decrypt the DVD.

Extra Credit

Did SHA-256 and SHA-512-truncated-to-256-bits have the same security properties? Which one is better? Please explain in detail.

Both methods offer a high level of security since they both offer collision resistance, pre-image resistance and second pre-image resistance. However, SHA-512-truncated-to-256-bits may potentially offer better resistance to certain types of attacks due to its larger internal state and message block size. And it also runs faster on 64-bit processors.

What's more SHA-256 is more vulnerable to length extension attack.