Crypto Engineering Midterm Exam

April, 2023

Student Name : 吳承瑀

Student ID: 110550091

Department : 資工系

For the following Questions, please write down your answers with explanations

Question 1:

Let us see what goes wrong when a stream cipher key is used more than once. Below are eleven hex-encoded ciphertexts that are the result of encrypting eleven plaintexts with a stream cipher, all with the same stream cipher key. Your goal is to decrypt the last ciphertext and submit the secret message within it as solution.

Hint: XOR the ciphertexts together and consider what happens when a space is XORed with a character in [a-z A-Z].

Ciphertext #1:

315c4eeaa8b5f8aaf9174145bf43e1784b8fa00dc71d885a804e5ee9fa40b16349c146fb778cdf2d3aff021dfff5b403b510d0d0455468aeb98622b137dae857553ccd8883a7bc37520e06e515d22c954eba5025b8cc57ee59418ce7dc6bc41556bdb36bbca3e8774301fbcaa3b83b220809560987815f65286764703de0f3d524400a19b159610b11ef3e

Ciphertext #2:

234c02ecbbfbafa3ed18510abd11fa724fcda2018a1a8342cf064bbde548b12b07df44ba7191d9606ef4081ffde5ad46a5069d9f7f543bedb9c861bf29c7e205132eda9382b0bc2c5c4b45f919cf3a9f1cb74151f6d551f4480c82b2cb24cc5b028aa76eb7b4ab24171ab3cdadb8356f

Ciphertext #3:

32510ba9a7b2bba9b8005d43a304b5714cc0bb0c8a34884dd91304b8ad40b62b07df44ba6e9d8a2368e51d04e0e7b207b70b9b8261112bacb6c866a232dfe257527dc29398f5f3251a0d47e503c66e935de81230b59b7afb5f41afa8d661cb

Ciphertext #4:

32510ba9aab2a8a4fd06414fb517b5605cc0aa0dc91a8908c2064ba8ad5ea06a029056f47a8ad3306ef5021eafe1ac01a81197847a5c68a1b78769a3

7bc8f4575432c198ccb4ef63590256e305cd3a9544ee4160ead45aef5204
89e7da7d835402bca670bda8eb775200b8dabbba246b130f040d8ec6447
e2c767f3d30ed81ea2e4c1404e1315a1010e7229be6636aaa

Ciphertext #5:

3f561ba9adb4b6ebec54424ba317b564418fac0dd35f8c08d31a1fe9e24fe
56808c213f17c81d9607cee021dafe1e001b21ade877a5e68bea88d61b9
3ac5ee0d562e8e9582f5ef375f0a4ae20ed86e935de81230b59b73fb4302
cd95d770c65b40aaa065f2a5e33a5a0bb5dcaba43722130f042f8ec85b7c
2070

Ciphertext #6:

32510bfbacfbb9befd54415da243e1695ecabd58c519cd4bd2061bbde24e
b76a19d84aba34d8de287be84d07e7e9a30ee714979c7e1123a8bd9822
a33ecaf512472e8e8f8db3f9635c1949e640c621854eba0d79eccf52ff111
284b4cc61d11902aebc66f2b2e436434eacc0aba938220b084800c2ca4e6
93522643573b2c4ce35050b0cf774201f0fe52ac9f26d71b6cf61a711cc22
9f77ace7aa88a2f19983122b11be87a59c355d25f8e4

Ciphertext #7:

32510bfbacfbb9befd54415da243e1695ecabd58c519cd4bd90f1fa6ea5ba
47b01c909ba7696cf606ef40c04afe1ac0aa8148dd066592ded9f8774b52
9c7ea125d298e8883f5e9305f4b44f915cb2bd05af51373fd9b4af511039f
a2d96f83414aaaf261bda2e97b170fb5cce2a53e675c154c0d9681596934
777e2275b381ce2e40582afe67650b13e72287ff2270abcf73bb02893283
6fbdecfecee0a3b894473c1bbeb6b4913a536ce4f9b13f1efff71ea313c866
1dd9a4ce

Ciphertext #8:

315c4eeaa8b5f8bffd11155ea506b56041c6a00c8a08854dd21a4bbde54c
e56801d943ba708b8a3574f40c00fff9e00fa1439fd0654327a3bfc860b92f
89ee04132ecb9298f5fd2d5e4b45e40ecc3b9d59e9417df7c95bba410e9a
a2ca24c5474da2f276baa3ac325918b2daada43d6712150441c2e04f6565
517f317da9d3

Ciphertext #9:

271946f9bbb2aeadec111841a81abc300ecaa01bd8069d5cc91005e9fe4a
ad6e04d513e96d99de2569bc5e50eeeca709b50a8a987f4264edb6896fb
537d0a716132ddc938fb0f836480e06ed0fcd6e9759f40462f9cf57f45641
86a2c1778f1543efa270bda5e933421cbe88a4a52222190f471e9bd15f65
2b653b7071aec59a2705081ffe72651d08f822c9ed6d76e48b63ab15d020
8573a7eef027

Ciphertext #10:

466d06ece998b7a2fb1d464fed2ced7641ddaa3cc31c9941cf110abbf409e
d39598005b3399ccfafb61d0315fca0a314be138a9f32503bedac8067f03a
dbf3575c3b8edc9ba7f537530541ab0f9f3cd04ff50d66f1d559ba520e89a
2cb2a83

Target Ciphertext (decrypt this one):

32510ba9babebbbefd001547a810e67149caee11d945cd7fc81a05e9f85a
ac650e9052ba6a8cd8257bf14d13e6f0a803b54fde9e77472dbff89d71b5
7bddef121336cb85ccb8f3315f4b52e301d16e9f52f904

```
(base) lily@Lily-d-Mac Mitdterm % python3 q1.py
1  | The secuet message is: WhZn using a stream cipher, never use the key more than once
2  | We can aactor the number 5 with quantum computers. We can also factor the number 1
3  | Euler whuld probably enjoF that now his theorem becomes a corner stone of crypto -
4  | The nicb thing about KeeySoq is now we cryptographers can drive a lot of fancy cars
5  | The cipoertext produced bF a weak encryption algorithm looks as good as ciphertext
6  | You don t want to buy a sZt of car keys from a guy who specializes in stealing cars
7  | There aue two types of crFptography - that which will keep secrets safe from your l
8  | There aue two types of cyOtography: one that allows the Government to use brute for
9  | We can tee the point wherZ the chip is unhappy if a wrong bit is sent and consumes
10 | A (privfte-key)  encryptiPn scheme states 3 algorithms, namely a procedure for gene
```

Ans:
The secret message is: When using a stream cipher, never use the key more than once

Question 2:

Suppose you are told that the one time pad encryption of the message “attack at dawn” is “09e1c5f70a65ac519458e7e53f36” (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 “attack at dusk” under the same OTP key?

Ans: 0x9e1c5f70a65ac519458e7f13b33

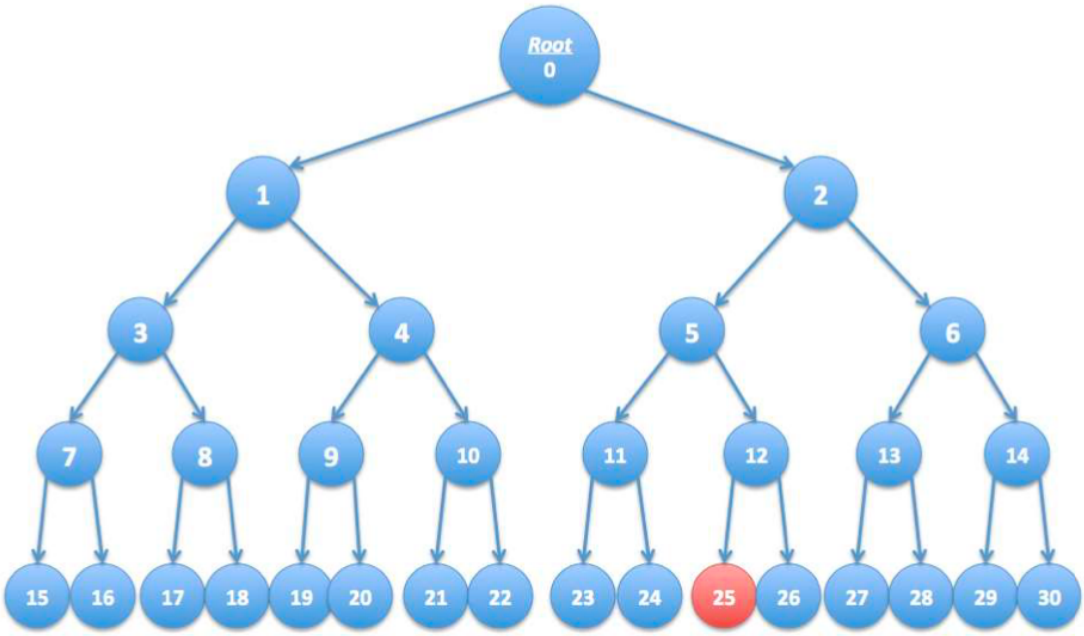
Question 3:

The movie industry wants to protect digital content distributed on DVD’s. We develop a variant of a method used to protect Blu-ray disks called AACS.

Suppose there are at most a total of n DVD players in the world (e.g. $\mathrm{n} = 2^{32}$). We view these n players as the leaves of a binary tree of height $log_2\mathrm{n}$. Each node in this binary tree contains an AES key k_i. These keys are kept secret from consumers and are fixed for all time. At manufacturing time each DVD player assigned a serial number $\mathrm{i} \in [0, \mathrm{n} - 1]$. Consider the set of nodes S_i along the path from the root to leaf number i in the binary tree. The manufacturer of the DVD player embeds in player number i the keys associated with the nodes in the set S_i. A DVD movie m is encrypted as $E(k_{root}, k) \,\|\, E(k, m)$ where k is a random AES key called a content-key and k_{root} is the key associated with the root of the tree. Since all DVD players have the key k_{root} all players can decrypt the movie m. We refer to $E(k_{root}, k)$ as the header and $E(k, m)$ as the body. In what follows the DVD header may contain multiple ciphertexts which each ciphertext is the encryption of the content-key k under some key k_i in the binary tree.

Suppose the keys embedded in DVD player number r are exposed by hackers and published on the Internet. In this problem we show that when the movie industry distributes a new DVD movie, they can encrypt the contents of the DVD using a slightly larger header (containing about $log_2 n$ keys) so that all DVD players, except for player number r, can decrypt the movie. In effect, the movie industry disables player number r without affecting other players.

As shown below, consider a tree with $n = 16$ leaves. Suppose the leaf node labeled 25 corresponds to an exposed DVD player key. Check the set of keys below under which to encrypt the key k so that every player other than player 25 can decrypt the DVD. Only four keys are needed.



(A) 14 (B) 28 (C) 11 (D) 4 (E) 6 (F) 2 (G) 26 (H) 1

We can't encrypt key *k* under any key on the path from the root to 25.
So, encrypt the highest node on the path that does not include 25.
On level 0, we can't encrypt the root, so skip it.
On level 1, we can't encrypt 2, so encrypt 1.
On level 2, we can't encrypt 5, so encrypt 6.
On level 3, we can't encrypt 12, so encrypt 11.
On level 4, we can't encrypt 25, so encrypt 26.

Ans: (C), (E), (G), (H)

Question 4:

Continuing with the previous question, if there are n DVD players, what is the number of keys under which the content key k must be encrypted if exactly one DVD player's key needs to be revoked?

(A) 2 (B) n-1 (C) log_2n (D) n/2 (E) $\sqrt{n}$

The key will need to be encrypted under one key for each node on the path from the root to the revoked leaf and there are $\log_2 n$ nodes on the path. In another word, the height of the tree is $\log_2 n$, so we need to encrypt this many nodes.

Ans: (C)

Question 5:

In the following let p be a prime. The set $Z_p = \{ x \text{ integer}, \text{ such that } 0 \leq \text{x} < \text{p}\}$ is a group with respect to addition modulo p (i.e. every element x in Z_p has an inverse $-x \in Z_p$ such that $x + (-x) = 0 \; mod \; p$. The set $Z_p^* = \{ x \text{ integer}, \text{ such that } 0 < \text{x} < \text{p}\}$ is a group with respect to multiplication modulo p (i.e. every element x in Z_p^* has an inverse $x^{-1} \in Z_p^*$ such that $xx^{-1} = 1 \; mod \; p$.

Another cipher with perfect secrecy. Consider the following cipher.

Let Z_p^* be the message space, the key space and the ciphertext space.

Alice and Bob share a key $k \in Z_p^*$ uniformly chosen at random. To send a message $m \in Z_p^*$ to Bob, Alice computes the ciphertext $c =$

$mk \ mod \ p$.

1. Prove that this cipher provides **Perfect Secrecy** using the criterium we proved in class.

Ans:
For c=mk mod p, we can have mk-c = px, x ∈ Z
→ $k-cm^{-1} = pxm^{-1}$ → $k = pxm^{-1} + cm^{-1}$ → $k = m^{-1}(px+c)$
Then, we have Pr[k ∈ K: E(k, m) = c] = 1.
For m0, m1 ∈ Z*p, |m0| = |m1|, we have
Pr[E(k, m0) = c] = Pr[E(k, m1)=c].
So, this cipher provides perfect secrecy.

2. Why one-time pad are **Perfect Secrecy** and also **Semantic Secu**re?

Ans:
Perfect Secrecy:
The key is truly random, is used only once, and is kept completely secret. These properties make it impossible for an attacker to derive any information about the plaintext from the ciphertext, without knowledge of the key. However, the one-time pad has practical limitations due to the need for a truly random and unpredictable key that is at least as long as the plaintext message.

Semantic Secure:
It produces ciphertext that is completely indistinguishable from random data, regardless of the underlying plaintext. Since the key used for encryption is truly random and is at least as long as the plaintext message, it produces a ciphertext that has no statistical correlation with the plaintext. Therefore, an attacker cannot distinguish between two ciphertexts corresponding to two different plaintexts.

3. Is the use of one-time pads susceptible to statistical analysis (especially if it is known that the plaintext is in American English)?

Ans:
No, the use of one-time pads is not susceptible to statistical analysis because it produces ciphertext that is completely indistinguishable from random data, regardless of the underlying plaintext. From the result, we will not be able to tell that the original text was from English or not nor will we be able do any statistical analysis.

4. Did public-key encryption scheme provide Perfect Secrecy? We assume there is a public-key encryption scheme (KeyGen, Enc, Dec) with perfect correctness (i.e., for all messages M and valid key-pairs (PK, SK), we have $DEC_{sk}(Enc_{pk}(\text{M})) = \text{M}$).

Ans:
No, unlike one-time pads, public-key encryption scheme provides a level of security that depends on the mathematical complexity of certain computational problems. They do not provide perfect secrecy. The security of a public-key encryption scheme relies on the difficulty of computing the private key from the public key or on the difficulty of factoring large composite numbers depending on the scheme used.

Question 6:

Predicting generators. Consider the following *congruential generator*. It uses constants $a, b \in Z_p^*$. The seed is a value $x_0 \in Z_p^*$. The i^{th} value generated is computed as $x_i = ax_{i-1} + b \; mod \; p$.

The sequence output by the generator is $S = x_0, x_1, x_2, \ldots$ Assume that an attacker knows p and witness the sequence.

1. Prove that after a short prefix (i.e. a few of the values x_i's) the attacker is able to predict the rest of the sequence (i.e. the rest of the x_j's).

Ans:
Suppose the attacker have the sequence x0, x1, x2
Then we can have the following two equations:
x1 = ax0 + b (mod p)
x2 = ax1 + b (mod p)
→ x2 – x1 = a(x1 – x0) (mod p)
→ a = (x2 – x1)/(x1 – x0) (mod p)
→ b = x1 – ax0 (mod p)
After knowing a and b along with p, the attacker can easily predict the rest of the sequence.

2. What does this say about the security of using the congruential generator as the keystream generator for a stream cipher?

Ans:
Using a congruential generator as the keystream generator for a stream cipher can be risky because it may produce output that repeats after a certain number of iterations or exhibits non-randomness, making it vulnerable to attacks. Or, if the rule of the sequence is easily breakable, then the attacker can know the keystream without a sweat as proven in (1). To mitigate these risks, it is important to carefully choose the generator parameters and use a secure initialization process or use a more secure sequence generating rule.

3. If an attacker knows constants $a, b \in Z_p^*$ and p. How many output bits $S = x_0, x_1, x_2, \ldots$ did the attacker to know to rest sequences.

Ans:
Since the attacker already knows a, b, and p, he or she just has to follow the formula x{i+1} = ax{i} + b (mod p) to calculate the rest of the sequence. In other words, as long as the attacker knows one bit of the output sequence, say x3, then the he or she can compute x0, x1, x2, x4, x5, x6, and so on.

4. However, If an attacker knows p but know nothing about constants $a, b \in Z_p^*$. In this case, How many output bits $S = x_0, x_1, x_2, \ldots$ did the attacker need to know to recover the rest sequence?

Ans:
As stated in (1), the attacker only needs 3 output sequence bits, say x0, x1, and x2 to have the following two equations:
x1 = ax0 + b (mod p)
x2 = ax1 + b (mod p)
and solve for a and b,
after solving a and b, the attacker can easily predict the rest.

Question 7:

In standard RSA the modulus N is a product of two distinct primes. Suppose we choose the modulus so that it is a product of three distinct primes, namely $\mathrm{N} = \mathrm{pqr}$. Given an exponent e relatively prime to $\varphi(N)$ we can derive the secret key as $d = e^{-1} \, mod \, \varphi(N)$. The public key (N, e) and secret key (N, d) work as before. What is $\varphi(N)$ when N is a product of three distinct primes?
(A) $\varphi(N) = pqr - 1$
(B) $\varphi(N) = (p-1)(q-1)(r+1)$
(C) $\varphi(N) = (p+1)(q+1)(r+1)$
(D) $\varphi(N) = (p-1)(q-1)(r-1)$

In RSA encryption method, N = pq and $\varphi(N) = (p-1)(q-1)$, so with 3 primes, it would be $\varphi(N) = (p-1)(q-1)(r-1)$.
Ans: (D)

Question 8:

Suppose we choose the modulus so that it is a product of three distinct primes, namely N = 105. Given an encryption key is 13 which is co-prime to $\varphi(N)$. Please find the secret key as $d = e^{-1} mod\ \varphi(N)$ using Extended Euclidean Algorithm.

Ans:
N=105, so the three primes are: 3, 5, 7.
ϕ(N) = (p-1)(q-1)(r-1) = 2*4*6 = 48.
Having e = 13 and $d \equiv e^{-1} \bmod \phi(N)$, $d \equiv 13^{-1} \bmod 48$.
→ $d \equiv 13^{-1} \bmod 48$
→ $1 \equiv 13d \bmod 48$
→ d = 37 (13*37 – 48*10 = 1)

Question 9:

An attacker intercepts the following ciphertext (hex encoded) :

20814804c1767293b99f1d9cab3bc3e7
ac1e37bfb15599e5f40eef805488281d

He knows that the plaintext is the ASCII encoding of the message “Pay Bob 100$” (excluding the quotes). He also knows that the cipher used is CBC encryption with a random IV using AES as the underlying block cipher. Show that the attacker can change the ciphertext so that it will decrypt to “Pay Bob 500$”. What is the resulting ciphertext (hex encoded)? This shows that CBC provides no integrity.

Since we only have to change the 9th character (1 -> 5) so we only have

to change the 9th character's ciphertext. Each ASCII code yields two hex codes, so the 9th character is 0xb9.
0xb9 decrypts to 1
0xb9 ⊕ ASCII(1 ⊕ 5) = 0xb9 ⊕ 0x31 ⊕ 0x35 = 0xbd

Ans:
20814804c1767293bd9f1d9cab3bc3e7
ac1e37bfb15599e5f40eef805488281d

Question 10:

Let G be a finite cyclic group (e.g., $G = \mathbb{Z}_p^*$) with generator g. Suppose the Diffie-Hellman function $DH_g(g^x, g^y) = g^{xy}$ is difficult to compute in G. Which of the following functions is also difficult to compute:
As usual, identify the f below for which the contra-positive holds:

if $f(\cdot, \cdot)$ is easy to compute then so is $DH_g(\cdot, \cdot)$.

If you can show that then it will follow that if DH_g is hard to compute in G then so must be f.

(A) $f(g^x, g^y) = g^{2xy}$
(B) $f(g^x, g^y) = (g^2)^{x+y}$
(C) $f(g^x, g^y) = \sqrt{g^{xy}}$
(D) $f(g^x, g^y) = g^{x-y}$

(B): $(g^2)^{x+y} = g^{2x} * g^{2y}$ → It is easy to solve.

(D) $g^{x-y} = g^x / g^y$ → It is easy to solve.

Ans: (A), (C) since they can't be broken down into easier computed equations.