COMP7906A Introduction to Cyber Security

Assignment 1

Wang Dingrui

3036196908

September 25, 2024

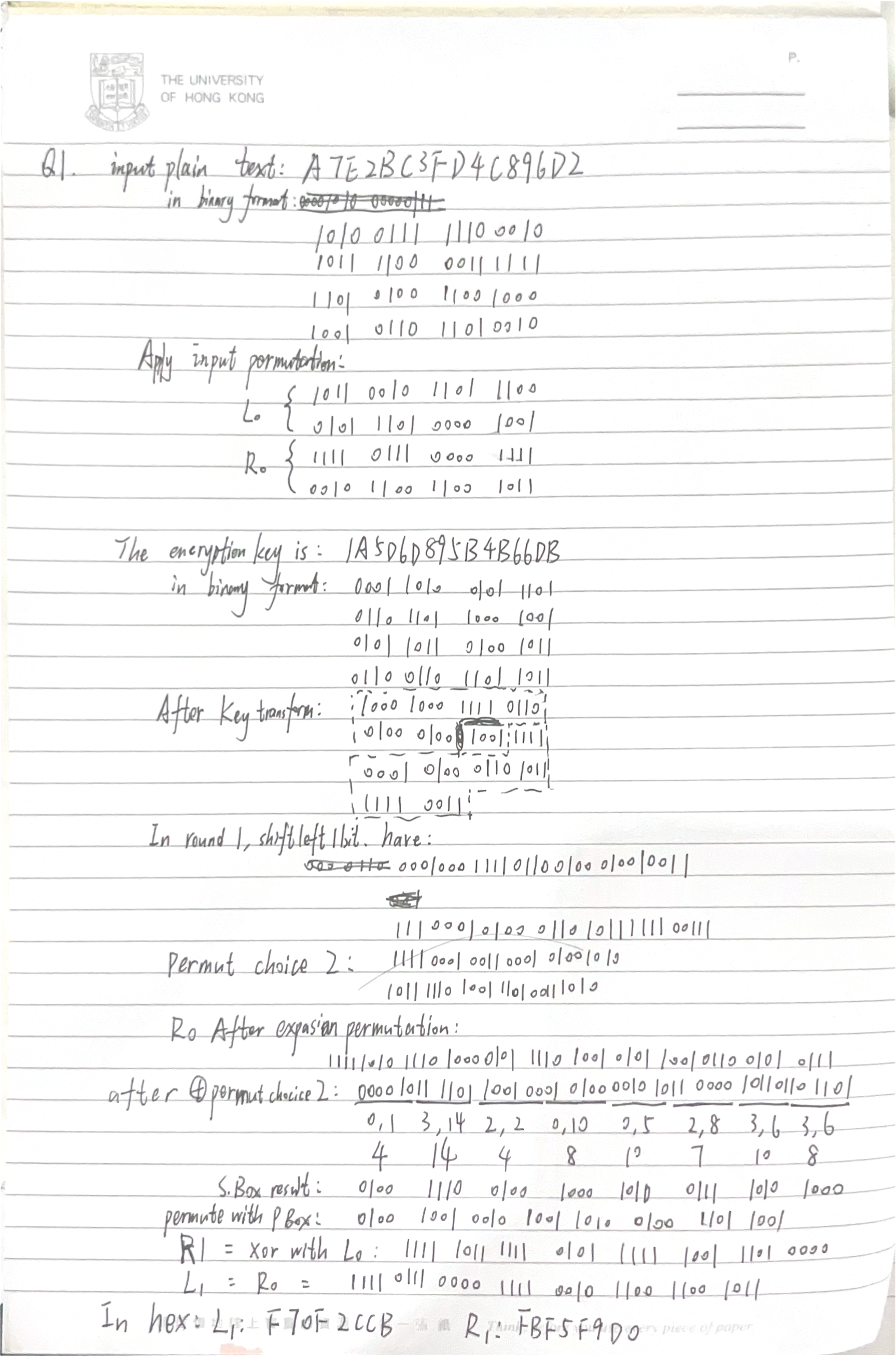
[Q1: 2](#_Toc178122426)

[Q2 8](#_Toc178122427)

[(a) 8](#_Toc178122428)

[(b) 9](#_Toc178122429)

# Q1:



Helper Code:

def getBinMat(s):

mat = ""

for i in range(len(s)):

mat += bin(eval("0x"+s[i]))[2:].zfill(4)

for i in range(len(mat)):

if i % 4 == 0:

print(" ", end="")

if i % 16 == 0:

print()

print(mat[i],end="")

return mat

def permute(mat, p):

mat2 = ""

for i in range(len(p)):

mat2 += mat[p[i]-1]

return mat2

def shift\_left(mat, n):

return mat[n:] + mat[:n]

def xor(mat1, mat2):

return "".join([str(int(mat1[i]) ^ int(mat2[i])) for i in range(len(mat1))])

input\_permutation = [58, 50, 42, 34, 26, 18, 10, 2,

60, 52, 44, 36, 28, 20, 12, 4,

62, 54, 46, 38, 30, 22, 14, 6,

64, 56, 48, 40, 32, 24, 16, 8,

57, 49, 41, 33, 25, 17, 9, 1,

59, 51, 43, 35, 27, 19, 11, 3,

61, 53, 45, 37, 29, 21, 13, 5,

63, 55, 47, 39, 31, 23, 15, 7]

key\_permutation = [57, 49, 41, 33, 25, 17, 9,

1, 58, 50, 42, 34, 26, 18,

10, 2, 59, 51, 43, 35, 27,

19, 11, 3, 60, 52, 44, 36,

63, 55, 47, 39, 31, 23, 15,

7, 62, 54, 46, 38, 30, 22,

14, 6, 61, 53, 45, 37, 29,

21, 13, 5, 28, 20, 12, 4]

key\_permutation\_2 = [14, 17, 11, 24, 1, 5,

3, 28, 15, 6, 21, 10,

23, 19, 12, 4, 26, 8,

16, 7, 27, 20, 13, 2,

41, 52, 31, 37, 47, 55,

30, 40, 51, 45, 33, 48,

44, 49, 39, 56, 34, 53,

46, 42, 50, 36, 29, 32]

expansion\_permutation = [32, 1, 2, 3, 4, 5,

4, 5, 6, 7, 8, 9,

8, 9, 10, 11, 12, 13,

12, 13, 14, 15, 16, 17,

16, 17, 18, 19, 20, 21,

20, 21, 22, 23, 24, 25,

24, 25, 26, 27, 28, 29,

28, 29, 30, 31, 32, 1]

s\_box\_1 = [[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],

[0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],

[4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],

[15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13]]

s\_box\_2 = [[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],

[3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],

[0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],

[13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9]]

s\_box\_3 = [[10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],

[13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],

[13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],

[1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12]]

s\_box\_4 = [[7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],

[13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],

[10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],

[3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14]]

s\_box\_5 = [[2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9],

[14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],

[4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],

[11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3]]

s\_box\_6 = [[12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],

[10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],

[9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],

[4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13]]

s\_box\_7 = [[4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],

[13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],

[1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],

[6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12]]

s\_box\_8 = [[13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],

[1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],

[7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],

[2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11]]

s\_boxes = [s\_box\_1, s\_box\_2, s\_box\_3, s\_box\_4, s\_box\_5, s\_box\_6, s\_box\_7, s\_box\_8]

p\_box\_permutation = [16, 7, 20, 21, 29, 12, 28, 17,

1, 15, 23, 26, 5, 18, 31, 10,

2, 8, 24, 14, 32, 27, 3, 9,

19, 13, 30, 6, 22, 11, 4, 25]

if \_\_name\_\_ == "\_\_main\_\_":

s = "A7E2BC3FD4C896D2"

s\_mat = getBinMat(s)

s\_init\_perm = permute(s\_mat, input\_permutation)

l0 = s\_init\_perm[:32]

r0 = s\_init\_perm[32:]

print("\n\n")

for i in range(len(s\_init\_perm)):

if i % 4 == 0:

print(" ", end="")

if i % 16 == 0:

print()

print(s\_init\_perm[i],end="")

key = "1A5D6D895B4B66DB"

print("\n\n")

print("key: ", key)

key\_mat = getBinMat(key)

key\_perm = permute(key\_mat, key\_permutation)

print("\n\n")

for i in range(len(key\_perm)):

if i % 4 == 0:

print(" ", end="")

if i % 16 == 0:

print()

print(key\_perm[i],end="")

key\_left = key\_perm[:28]

key\_right = key\_perm[28:]

print("\n\n")

key\_left\_shift = shift\_left(key\_left, 1)

key\_right\_shift = shift\_left(key\_right, 1)

print("key\_left\_shift: ", key\_left\_shift)

print("key\_right\_shift: ", key\_right\_shift)

key\_shifted = key\_left\_shift + key\_right\_shift

key\_perm\_2 = permute(key\_shifted, key\_permutation\_2)

print("key\_perm\_2: ")

print(key\_perm\_2[:24])

print(key\_perm\_2[24:])

expanded\_r0 = permute(r0, expansion\_permutation)

print("expanded\_r0: ", expanded\_r0)

xor\_result = xor(expanded\_r0, key\_perm\_2)

print("expanded\_r0 ^ key\_perm\_2: ", xor\_result)

s\_box\_result = ""

for i in range(8):

sb\_input = xor\_result[i\*6:(i+1)\*6]

row\_num = int(sb\_input[0] + sb\_input[5], 2)

col\_num = int(sb\_input[1:5], 2)

target = s\_boxes[i][row\_num][col\_num]

bin\_target = bin(target)[2:].zfill(4)

s\_box\_result += bin\_target

print("sb\_input: ", sb\_input, "row\_num: ", row\_num, "col\_num: ", col\_num, "target: ", target, bin\_target)

print("s\_box\_result: ", s\_box\_result)

p\_box\_result = permute(s\_box\_result, p\_box\_permutation)

print("p\_box\_result: ", p\_box\_result)

xor\_result\_2 = xor(p\_box\_result, l0)

print("r1 = xor with l0: ", xor\_result\_2, "In hex per 4 bit: ", hex(int(xor\_result\_2, 2)).upper()[2:])

print("l1 = r0: ", r0, "In hex per 4 bit: ", hex(int(r0, 2)).upper()[2:])

A screenshot of a computer

Description automatically generated

# Q2

## (a)

Firstly, calculate the index of coincidence using Analyzer.java.

The result shows that when sequence length is 1, the index of coincidence is 0.04214280266060201, which is much smaller than 0.068, so it is polyalphabetic cipher.

(base) nowonder@MacBook-Pro tools % java Analyzer.java Q2\_cipher.txt 1

Analyzer.java:148: warning: [unchecked] unchecked call to add(E) as a member of the raw type Vector

countedStr.add(str);

^

where E is a type-variable:

E extends Object declared in class Vector

1 warning

reading file...

total number of letters = 1134

index of Coincidence = 0.04214280266060201

frequency counting...

s 72 probability=0.06349207

f 69 probability=0.06084656

w 68 probability=0.059964728

v 67 probability=0.05908289

i 61 probability=0.053791888

u 59 probability=0.05202822

h 58 probability=0.051146384

b 50 probability=0.04409171

j 49 probability=0.043209877

m 47 probability=0.04144621

e 42 probability=0.037037037

r 42 probability=0.037037037

z 41 probability=0.0361552

g 40 probability=0.03527337

d 39 probability=0.034391534

k 38 probability=0.0335097

o 37 probability=0.032627866

l 36 probability=0.031746034

x 32 probability=0.028218694

y 32 probability=0.028218694

c 31 probability=0.02733686

t 28 probability=0.024691358

q 26 probability=0.02292769

a 25 probability=0.022045854

p 23 probability=0.020282187

n 22 probability=0.019400353

## (b)

Use Analyzer.java in tools and set the length of sequence to a larger number, say 15.

It can be seen from the result that in 8-gram counting, some strings appear more than two times:

evmwiodm 3 position=18,459,746

vmwiodmb 3 position=19,460,747

and in 9-gram counting, “evmwiodmb” appeared 3 times.

evmwiodmb 3 position=18,459,746

747 - 460 = 746 - 459 = 287, the factors of 287 include 7, 41 and 287

460 - 18 = 459 - 18 = 441, the factors of 441 include 3, 7, 9, 21, 49, 63, 147 and 441

The common factor is 7. According to Kasiski theory, 7 is possibly the length of the key.

According to the table, E and T are the letters with the largest probability to appear in a English paragragh.

A table of numbers and letters

Description automatically generated with medium confidence

Use Kasiski.java and set n-sequence=7.

From the first to the 7th of the key, the counted mostly appeared letters in the cipher and indexes of coincidence are listed in the table.

|  |  |  |
| --- | --- | --- |
| Position | Probs | index of coincidence |
| 1 | v 20 probability=0.12345679  z 19 probability=0.11728395  … | 0.07200368069933287 |
| 2 | s 23 probability=0.14197531  b 18 probability=0.11111111  … | 0.07024001226899777 |
| 3 | s 24 probability=0.14814815  g 16 probability=0.09876543  … | 0.0703166935050993 |
| 4 | i 20 probability=0.12345679  m 20 probability=0.12345679  … | 0.07407407407407407 |
| 5 | b 21 probability=0.12962963  f 14 probability=0.086419754  … | 0.0647189632696879 |
| 6 | u 17 probability=0.10493827  n 16 probability=0.09876543  … | 0.06065485775630703 |
| 7 | h 24 probability=0.14814815  d 21 probability=0.12962963  … | 0.07553101756000306 |

Assume that e can be the top k of the encoded text, write a program to try the possible alters of e.

The alters are: vswifyh

So, the key is: rosebud

The shift is:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 17 | 14 | 18 | 4 | 1 | 20 | 3 |

There is no permutation. The substitution is shown in the table.

## (c)

over recent decade s australia s foreign relations have been driven by a close association with the united states through the an zusp act and by a desire to develop relationships with asia and the pacific particularly through as e an and the pacific islands forum three years ago australia secured an in augural seat at the east asia summit following its access i on to the treaty of amity and cooperation australia is a member of the commonwealth of nations in which the commonwealth heads of governzcaa aaa

# Q3

A black background with white text

Description automatically generated

I love my kitty

A screen shot of a computer

Description automatically generated  
My kitty loves me

A screen shot of a computer screen

Description automatically generated  
Together we're happy as can be

A black background with white letters

Description automatically generated

**Here extract the uppercase letters, have:**  
Though my head has suspicions

A black background with white letters

Description automatically generated  
That I keep under my hat

A screen shot of a computer screen

Description automatically generated  
Of what if I shrank to the size of a rat.

A black background with white text

Description automatically generated  
Yeah, she would probably eat me.

# Q4

Assume that the expiration period is T.

q = GT/R = (10^11)\*T/(95^10)

We need to make sure that q < 1 in 2^10

T<(1/2^10)\*(95^10)/(10^11)

T should be less than 584704.0422249795 seconds.

Which is 162.41778950693876 hours.

# Q5

## What are the key features w.r.t. to our discussion in class?

* No Periodic Password Changes: The guidelines suggest that passwords should not be changed periodically (e.g., every 90 days) unless there is evidence of a security breach. This is a shift from previous policies that mandated regular password updates, which often led to weaker password choices among users due to fatigue and limited password creativity.
* Drop the Complexity Requirements: The new guidelines recommend removing the requirement for character composition rules. Instead, they emphasize password length and discourage easily guessed passwords through the use of blacklists of known compromised passwords.
* Encourage Longer Passwords: The emphasis shifts towards encouraging longer passphrases rather than complex passwords. Longer passphrases, which are easier for users to remember and harder for attackers to crack, are promoted as more secure than shorter, complex passwords.
* Use of Screened Passwords: The guidelines recommend screening new passwords against lists of commonly used or compromised passwords. This helps prevent users from selecting passwords that are easily guessable or known to be compromised.
* Multi-factor Authentication: There is a strong recommendation for the use of Multi-factor Authentication to enhance security, especially for applications that handle sensitive or personal data.
* Rate Limiting and Lockout Mechanisms: It's recommended to implement rate limiting and lockout mechanisms to thwart brute force attacks. However, care must be taken to balance security and usability, as aggressive lockout policies can lead to denial of service for legitimate users.

1. Any similar guidelines in China? Compare the guidelines in China with the NIST SP 800-63B.

Yes, GB/T 35273-2020 is a similar guideline.

NIST SP 800-63B specifically addresses digital identity and authentication, providing detailed guidance on password policies, multi-factor authentication, and identity proofing. GB/T 35273-2020 is broader, focusing on overall personal information security rather than specifically on authentication methods. While it does recommend security measures, it is less prescriptive about specific authentication protocols.

NIST SP 800-63B provides detailed specifications for different levels of authentication assurance (AAL1 through AAL3), tailored to the risks associated with different types of digital transactions. GB/T 35273-2020 does not detail different levels of authentication assurance but requires the adoption of effective protective measures which can implicitly include multi-factor authentication where necessary.

# Q6

Take voice authentication as an example. Voice is a combination of physical and behavioral biometrics.

However, there are several weaknesses.

For example, attacker can record the voice of the user and plays back to gain the access.

Another way is using AI like voice deepfake technologies, to simulate the user’s voice. In this way, the attacker can create any fake sentences.

Also, there are techniques that adjust characteristics of one person's voice to make it sound like another person's voice.

Even if the attacker is not a tech guy, a skilled impersonator mimics the voice characteristics of another person. Some people have the ability to alter their vocal patterns significantly to impersonate others.