THE UNIVERSITY OF MELBOURNE

COMP90043 Cryptography and Security

Assignment 1

Student Name: Lihua Wang

Student ID: 1164051

Tutor: Lianglu Pan

Semester: 2020/S2

School of Computing and Information Systems

CONTENT

Q1. Classical Ciphers

a) $p = (c - b)a^{-1}\, mod\, 36$

b) b can take any value from integers modulo 36. However, a should have an inverse, a should belong to $\{1,5,7,11,13,17,19,23,25,29,31,35\}$ $(\varphi(36) = 12)$. So totally different keys are $12 \times 36 = 432$ keys. For non-trivial keys should be $12 \times 36 - 1 = 431$ keys

c) This cipher is considered as mono-alphabetic cipher. Because its mechanism is each character in an alphabet (# between 0~35) is mapped to its numeric equivalent, usually a mathematical function is applied to encrypt them and then converted to a fixed character. It is a simple substitution cipher and satisfies the standard of Monoalphabetic cipher:

- ➢ Each plain text alphabetic character is mapped to a unique ciphertext alphabetic character.
- ➢ There is a one-to-one relationship between plain text characters and cipher text characters.
- ➢ Each plaintext symbol is mapped to a ciphertext fixed symbol.

d) **Method 1- Cryptanalysis**

This scheme is essentially a one-to-one encryption method. So it does not hide the statistical law of the appearance of letters. In English, the appearance frequency of characters is relatively fixed for a sufficiently long English text. The frequency of some combinations of letters is also relatively fixed so that the ciphertext encrypted by the above encryption method can be cracked through statistical analysis. The key point is finding the two letters with the highest frequency, once make sure the transformation of the two letters, then we will get key $E_{[a,b]}$.

First, once determined the relative frequency of the letters in the ciphertext, then compared to a standard frequency distribution for English.

$$Ciphertext: RDTURDDGRDUIRDWERDRDRDRDRDRDRDRD$$

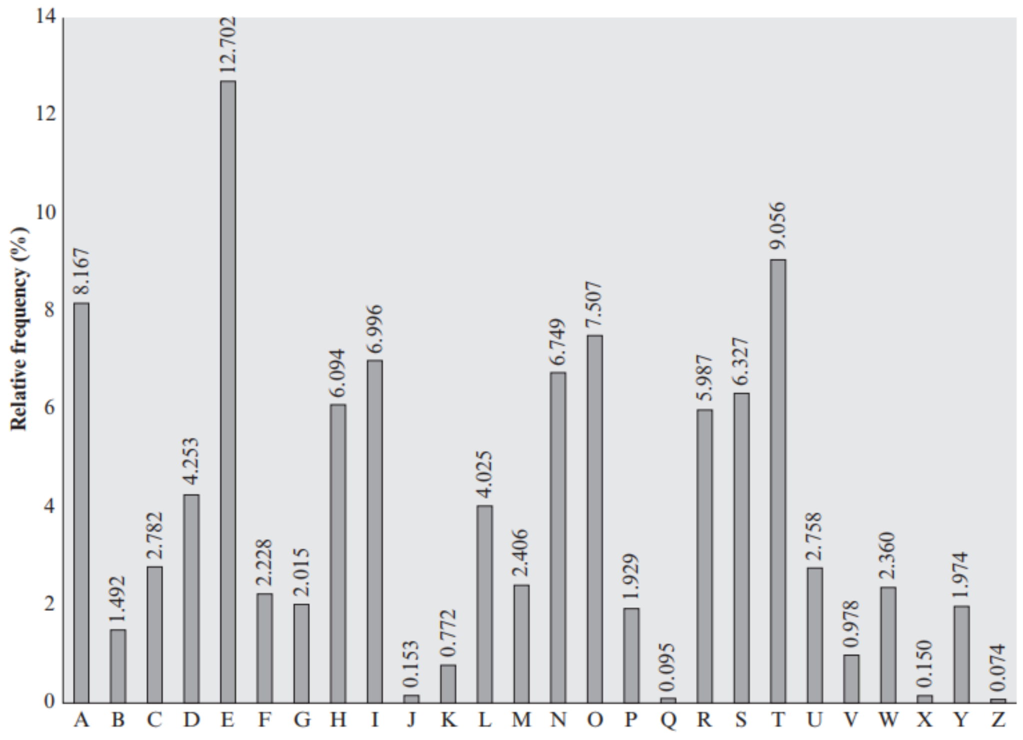

Figure1. Relative Frequency of Letters in English Text[1]

Secondly, comparing and matching the equivalents frequencies of letters in ciphertext and plaintext. In this case, letters "R" and "K" in cipher with the highest frequencies are likely included in the set $\{E, T\}$.

We assume "R" → "E", "D" → "T", then we can make the following equations: $(\mathrm{R} = 17, \mathrm{D} = 3, \mathrm{E} = 4, \mathrm{T} = 19)$

$$17 = (4a + b) mod 26$$
$$3 = (19a + b) mod 26$$

Then we get,

$$a = 6, b = 19$$

Because the $\mathrm{GCD}(a, 26) = 2$, so the assumption does not right.

We continue to assume that "R" is the encryption of "E", "E" is the encryption of "T", and get $a = 8$, which is still not satisfied. Continue to assume that "R" is the encryption of "E", "K" is the encryption of "T", finally get $a = 3, b = 5$, which is a legal key.

Try to fill in the plaintext to see if it looks like a reasonable meaning of a message. Once the meaning makes sense and readable, the keys are finally be found.

Method 2 - Brute-force attack:

Because for standard letters alphabet, for this scheme we have a total of $12 \times 26 - 1 = 311$ possible keys. It is easy to simply search for all possible keys by computers. In order to find the best key, we must try to decrypt the ciphertext with each key, and then determine the "suitability" of each decrypted text. In this way, we can rank different decryption keys. The decryption key we want is the key that generates the decrypted text with the least rare sequence.

e) In this case, it is a kind of choose-plaintext attack. Based on the $\mathrm{Q1(d)}$ method, we can make sure and easily get the 2 highest frequency of letters occurred in the ciphertext and plaintext respectively, and then apply the 4 digits on encryption equation to calculate E_a and E_b.

For general discussion, we establish following equations. Assuming existed top 2 frequencies letters $\mathrm{p_1, p_2}$ in the plaintext, corresponding with c_1, c_2 in ciphertext with the same frequency. Then we get,

$$c_1 = (ap_1 + b)mod26$$
$$c_2 = (ap_2 + b)mod26$$

There are the values p, q, r and s, in order to find a and b, we need to gain number $D = p_1 - p_2$, and D^{-1}, D^{-1} are numbers which makes $D \hat{} * x = 1\ (mod\ 26)$ valid, where x in set between 1 and 25. Then we can gain the value of a and b.

$$a = D^{-1}(c_1 - c_2)(mod26)$$
$$b = D^{-1}(p_1c_2 - p_2c_1)(mod26)$$

Then we try the key to decrypt ciphertext, if the result plaintext makes sense, then the key is finally get it!

Q2. General Security

For my perspective, confidentiality might be the most concern factor by the public using COVIDSafe app.

Any tracking application will cause privacy issues. In the COVIDSafe application, user information is stored on the AWS central server, so the data is public for Amazon, the Australian government, and anyone who can access to the server. They can freely access and read data without users' authorization, which may cause serious privacy leakage problems. At the same time, the access of these institutions will not be restricted by the updated law, which seriously violates the public's expectations for third parties to use their personal data.

Second, the working mechanism of COVIDSafe can also cause privacy issues. Due to the use of Bluetooth for identification and collection of personal data, it may increase the opportunities for third-party tracking based on Bluetooth, which also acts a great risk to user data. When attackers frequently test and use the app, it's easy to learn the contacts.

Finally, due to the long retention time of the information, the longer the retention time, the more vulnerable it is to be attacked by unauthorized access, which poses a great threat to user privacy.

At the same time, data integrity and app availability are also 2 factors that users will consider. For example, if this app works efficient and can effectively track close contacts, it will attract more users to download apps and then benefit the control of the epidemic while health officials specifying epidemic prevention strategies. However, the fact is, according to the survey shows that the download rate of apps is less than 40%, and the prerequisite for the app to be effective should be used by at least 60% of people to achieve the desired effect. The reason for not enough users is that they are worried about data leakage. Therefore, only by ensuring the confidentiality of data can the app's availability become stronger. At the same time, whether the collected data is reliable remains to be unsure. There is no way to judge without sufficient data as evidence.

Overall, the fundamental factor that users concerns about COVIDSafe lie in its data confidentiality. I believe that users' personal privacy and security should be the first to be guaranteed for both users and institutions.

Q3. Euclid's algorithm

I use Python to achieve the implement of algorithms of extend GCD and multiplicative inverse.

a) The screenshot of extended GCD algorithm method code as following:

b)
```
def extended_gcd(a, b):
    prev_x = 0
    prev_y = 1
    x = 1
    y = 0

    while a != 0:
        quotient = b // a
        b, a = a, b % a
        x, prev_x = prev_x - quotient * x, x
        y, prev_y = prev_y - quotient * y, y

    gcd = b
    return gcd, prev_x, prev_y
```

c) The screenshot of multiplicative inverse code as following:

```
def inverse(a, n):
    # a and n are 2 positive integers.
    if a <= 0 or n <= 0:
        print("Please input 2 positive integers!")
        return

    # Recall the extended_gcd API.
    gcd, x, y = extended_gcd(a, n)

    # Check if the inverse exists.
    if gcd != 1:
        print(a, 'mod', n, 'does not have an inverse!')
        return

    # Double check the result to be positive.
    if x < 0:
        x += n

    return x
```

d) $X = 1164051$

The screenshot of inverse of (X mod 16777259) code as following:

```
def main():
    a = 1164051
    b = 16777259
    gcd, x, y = extended_gcd(a, b)
    print('GCD =', gcd,
          ',\n x =', x,
          ',\n y =', y)
```

```
print('The inverse is:', inverse(a, b))
```

The result as following:

The inverse is: 1979601

Q4. Poly-alphabetic Cipher

a) The number of valid keys is same as the number of invertible matrices over of size m over z_{37}^{*}. According to Euler's totient function, we can get $\varphi(37) = 36$. As discussing different m, the following shows some numeric example answers:

when $m = 1, \#\ keys = 36$
when $m = 2, \#\ keys = (36^2 - 1) \times (36^2 - 36) = 1631700$
when $m = 3, \#\ keys = (36^3 - 1) \times (36^3 - 36) \times (36^3 - 36^2) =$
$98660544696{,}000$

b) To decrypt the ciphertext, we need to gain the key to decryption, which is the inverse of the key to the encryption. By solving this question, I made a program in Python to helps me calculate the result. The following is the explaining of my solution, the whole program code was attached as an appendix at last.

Based on the known plaintext and ciphertext block, we can calculate the encryption key. First, the ciphertext and plaintext are divided into two $5 * 5$ matrixes, and the characters are converted into digits, then we get plaintext matrix $P =$
$\begin{bmatrix} 23 & 35 & 1 & 32 & 19 \\ 32 & 9 & 0 & 22 & 29 \\ 20 & 4 & 24 & 33 & 5 \\ 7 & 8 & 22 & 32 & 30 \\ 26 & 31 & 27 & 31 & 25 \end{bmatrix}$, ciphertext matrix $C = \begin{bmatrix} 28 & 16 & 31 & 35 & 25 \\ 25 & 27 & 25 & 32 & 30 \\ 26 & 31 & 27 & 20 & 12 \\ 3 & 13 & 24 & 28 & 9 \\ 7 & 8 & 13 & 19 & 18 \end{bmatrix}$

Next, we need to gain the key in encryption, $K = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} & k_{15} \\ k_{21} & k_{22} & k_{23} & k_{24} & k_{25} \\ k_{31} & k_{32} & k_{33} & k_{34} & k_{35} \\ k_{41} & k_{42} & k_{43} & k_{44} & k_{45} \\ k_{51} & k_{52} & k_{53} & k_{54} & k_{55} \end{bmatrix}$.

Based on the Hill encryption mechanism, $C = P * K$, so $K = P^{-1} * C$. To calculate the P^{-1}, using Numpy.linalg package imported in python to calculate the determinant of matrix P (det _P), then recall the inverse method which written in extend_gcd package to convert to the inverse det_P and apply to each item in matrix P. Finally, we can get $P^{-1} = \begin{bmatrix} 28 & 29 & 15 & 19 & 14 \\ 14 & 16 & 23 & 30 & 19 \\ 32 & 2 & 35 & 30 & 31 \\ 6 & 6 & 0 & 33 & 14 \\ 24 & 7 & 35 & 34 & 28 \end{bmatrix}$, and $K = P^{-1} *$

$$C \, mod \, 37 \; = \begin{bmatrix} 19 & 20 & 9 & 92 & 27 \\ 22 & 24 & 17 & 36 & 16 \\ 17 & 32 & 2 & 20 & 36 \\ 34 & 22 & 15 & 1 & 28 \\ 20 & 30 & 10 & 29 & 5 \end{bmatrix}$$

To decrypt the ciphertext, we need to gain the key of decryption, which is the inverse of K. To calculate K^{-1}, we can easily repeat the above steps by calculating the inverse of determination of matrix K and multiply with each item in matrix K to get $K^{-1} = \begin{bmatrix} 14 & 13 & 33 & 21 & 16 \\ 33 & 21 & 23 & 17 & 34 \\ 0 & 5 & 34 & 15 & 3 \\ 14 & 32 & 0 & 0 & 7 \\ 20 & 26 & 32 & 6 & 26 \end{bmatrix}$. Then multiply matrix C with matrix K^{-1}, after moulding 37 with the result, we can easily affine these the numbers in alphabet table and convert to plain text.

For my result is: (ID:1164051)

4DT519Q9NPMG8FK01E9427S4WMRAWUS4ZWNDYXDMOGGCSQWV6E MLIKG

Q5. Probability

a) $My \; SID = 1164051 \rightarrow x = 4, y = 5$

$N = 5x + 6y + 15 = 5 \times 4 + 6 \times 5 + 15 = 65$

b) Because the pairs of digits have unique solution, so we only consider the probability of digits in place $4th$ and $6th$ is 4 and 5.

I applying on the binomial distribution and birthday attack theory. Based on situation, the 10 possible numbers of each digit are in the set $\{0,1,2,3,4,5,6,7,8,9\}$.

To find the p that at least two students have the same N, we can first calculate the $\bar{p}$ that non-students have the same N, and then $p = 1 - \bar{p}$. The first student have

N (x, y); the probability of second student with a different N from first student is $\frac{9}{10} * \frac{9}{10}$; the third student with a different N from the first student is also $\frac{9}{10} * \frac{9}{10}$, and so on. Therefore, the probability p that n student($s \leq 230$) have different N from the first student can be expressed by the following formula.

$$\overline{p}(s) = \left(\frac{9}{10} * \frac{9}{10}\right)^{230-1}$$

Then, the probability p that at least two students have the same N is 1 minus the above formula.

$$p(s) = 1 - \overline{p}(s) = 1 - \left(\frac{9}{10} * \frac{9}{10}\right)^{229} \approx 0.999$$

c) For each ball, it can be placed in any box from #1 to #5. Because the balls are different, the balls are independent of each other. Because there are 5 situations for each ball, then we get,

$ans = 5^{65} = 27105054312137610635795810087897756547353149 44$[3]

d) Nature is to divide the 65 balls into 5 groups (not empty), and then it turns into a problem of the number of combinations. Using the board method, it is equivalent to just inserting $(5-1)$ boards in the $(64-1)$ gaps of the 65 balls, that is, they are divided into 5 groups that are not empty.

$$ans = C_{65-1}^{5-1} = C_{64}^{4} = 635376$$

e) For each box, we place a ball, so it can be regarded as there are $(65+5)$ balls, and then delete a ball in each group after the arrangement is completed, so that we can enumerate the empty boxes.

$$ans = C_{65+5-1}^{5-1} = C_{69}^{4} = 864501$$

f) Because the condition is at most 2 bins are non-empty, so it can be considered two situations: one for there is only one non-empty bin; the other is there are 2 bins non-empty.

For the first situation, all balls in one bin and there are five choices: $C_5^1 = 5$;

For the second situation, all balls placed in two bins, so they have $C_5^2 = 10$ choices of bins. And when a ball placed into one bin, the rest balls were put in the other, so there are $C_{65}^1 = 65$ ways assigning balls. So, in this case, we have

$C_{65}^{1}C_{5}^{2} = 10 \times 65 = 650$ ways place balls.

Overall, the all ways should be add both results of two situations:

$$ans = C_{5}^{1} + C_{65}^{1}C_{5}^{2} = 5 + 650 = 655$$

Reference

1. William Stallings , Cryptography and Network Security: Principles and Practice, 7/E by William Stallings. (2017)

2. COVIDSafe app, (2020), https://www.health.gov.au/resources/apps-and-tools/covidsafe-app [Accessed 30 Aug. 2020].

3. OSGeo China, Online science calculator, https://www.osgeo.cn/app/sa901, [Accessed 31 Aug. 2020].

Appendix

hill_cipher.py

```
import numpy as np
import textwrap

from extended_euclid import inverse

m = 5
p_dic = {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6,
         'H': 7, 'I': 8, 'J': 9, 'K': 10, 'L': 11, 'M': 12,
         'N': 13, 'O': 14, 'P': 15, 'Q': 16, 'R': 17, 'S': 18,
         'T': 19, 'U': 20, 'V': 21, 'W': 22, 'X': 23, 'Y': 24,
         'Z': 25, '0': 26, '1': 27, '2': 28, '3': 29, '4': 30,
         '5': 31, '6': 32, '7': 33, '8': 34, '9': 35, ' ': 36}

c_dic = {c: p for p, c in p_dic.items()}


def convert_to_matrix(texts):
    matrix = []
    for text in texts:
        row = np.array([p_dic[p] for p in text])
        matrix.append(row)
    return np.array(matrix)


def sub_determinant(matrix, i, j, mod=37):
    tmp = np.delete(matrix, j, 0)
    sm = np.delete(tmp, i, 1)
    return int(round(np.linalg.det(sm))) % mod


def decrypt(ciphertext, key):
    c = np.array([p_dic[c] for c in ciphertext])
    p = np.dot(c, key) % 37
    return ''.join([c_dic[n] for n in p])


def main():
    plaintexts = ['X9B6T', '6JAW3', 'UEY7F', 'HIW64', '0515Z']
    ciphertexts = ['2Q59Z', 'Z1Z64', '051UM', 'DNY2J', 'HINTS']
```

```
matrix_P = convert_to_matrix(plaintexts)
matrix_C = convert_to_matrix(ciphertexts)

print("Matrix P =\n", matrix_P)
print("Matrix C =\n", matrix_C)

# Calculate the determinant of matrix P
det_P = int(round(np.linalg.det(matrix_P))) % 37
print("Determinant of P =", det_P)

# Calculate the inverse of the determinant of matrix P
inv_m = inverse(det_P, 37) % 37
print("(det P)^-1 mod 37 =", inv_m)

# calculate inverse of each item in matrix P
inv_P = [[] for _ in range(m)]
for i in range(m):
    for j in range(m):
        inv_P[i].append(inv_m * (-1) ** (i + j) * sub_determinant(matrix_P, i, j))

inv_P = np.array(inv_P) % 37
print("P^-1 = \n", inv_P)

# Key is equal to (inverse matrix P * matrix_C)% 37
K = np.dot(inv_P, matrix_C) % 37
print("K = \n", K)

"""
To decrypt the ciphertext,
we need to get decryption of key, that is
calculate the inverse matrix of K.
"""

# Calculate the determinant of matrix K
det_K = int(round(np.linalg.det(K))) % 37
print("Determinant of K =", det_K)

# Calculate the inverse of the determinant of matrix K
inv_m = inverse(det_K, 37) % 37
print("(det K)^-1 mod 37 =", inv_m)

# calculate the inverse value of each item in matrix K
inv_K = [[] for _ in range(m)]
for i in range(m):
    for j in range(m):
        inv_K[i].append(inv_m * (-1) ** (i + j)
                              * sub_determinant(K, i, j))

inv_K = np.array(inv_K) % 37
print("K^-1 = \n", inv_K)

"""
Decrypt cipher text
"""
test_ciphertext = \
    "A8VS3XRDEON6JEVXGJID13C07L4C1R4Q965XWRA5DQGYWTNHYO4ND8Z"
test_plaintext = ""
for c in textwrap.wrap(test_ciphertext, m):
    test_plaintext += decrypt(c, inv_K)

print("The Plaintext is:\n", test_plaintext)


main()
```

Result:

```
Matrix P =
[[23 35  1 32 19]
 [32  9  0 22 29]
 [20  4 24 33  5]
 [ 7  8 22 32 30]
 [26 31 27 31 25]]
Matrix C =
[[28 16 31 35 25]
 [25 27 25 32 30]
 [26 31 27 20 12]
 [ 3 13 24 28  9]
 [ 7  8 13 19 18]]
Determinant of P = 22
(det P)^-1 mod 37 = 32
P^-1 =
 [[28 29 15 19 14]
 [14 16 23 30 19]
 [32  2 35 30 31]
 [ 6  6  0 33 14]
 [24  7 35 34 28]]
K =
 [[19 20  9  9 27]
 [22 24 17 36 16]
 [17 32  2 20 36]
 [34 22 15  1 28]
 [20 30 10 29  5]]
Determinant of K = 2
(det K)^-1 mod 37 = 19
K^-1 =
 [[14 13 33 21 16]
 [33 21 23 17 34]
 [ 0  5 34 15  3]
 [14 32  0  0  7]
 [20 26 32  6 26]]
TEST PLAINTEXT:
 4DT519Q9NPMG8FK01E9427S4WMRAWUS4ZWNDYXDMOGGCSQWV6EMLIKG
```