

# Security & Cryptography Assignment 1

Anna Visman 6351115

01/12/2025

## 1 Historical Ciphers

- a) This ciphertext is made using the Caesar cipher with a shift(key) of 5. The plaintext is:

To be, or not to be, that is the question: Whether 'tis nobler in the mind to suffer The slings and arrows of outrageous fortune, Or to take arms against a sea of troubles And by opposing end them. To die—to sleep, No more; and by a sleep to say we end The heart-ache and the thousand natural shocks That flesh is heir to: 'tis a consummation Devoutly to be wish'd.

- b) This cipher is made using the Vigenère cipher with the key "oppenheimer". The plaintext is:

Reveal Thy Self; what awful Form art Thou? I worship Thee! Have mercy, God supreme! Thine inner Being I am fain to know; This Thy forth-streaming Life bewilders me. Time am I, laying desolate the world, Made manifest on earth to slay mankind! Not one of all these warriors ranged for strife Escapeth death; thou shalt alone survive. Therefore stand up! win for thyself renown, Conquer thy foes, enjoy the wealth filled realm, By Me they are already overcome, Be thou the outward cause, left-handed one. Drona and Bhishma and Jayadratha, Karna, and all the other warriors here, Are slain by Me. Destroy them fearlessly. Fight! thou shalt crush thy rivals in the field. Sanjaya said: Having heard these words of Keshava, he who weareth a diadem, with joined palms, quaking and prostrating himself, spake again to Kri- shna, stammering with fear, casting down his face.

- c) The code for the cipher in part a) can be found in appendix A.0.1. A proof of the resulting key and plaintext can be found in appendix 1.

## 2 Number Theory and Elliptic Curve

- a)  $x \equiv 13^{\frac{1}{7}} \pmod{119}$  is equivalent to  $x^7 \equiv 13 \pmod{119}$  We can undo the exponentiation of  $x$  by raising 13 to the inverse of 7  $\pmod{\phi(119)}$ , or for  $d$  in  $7d \equiv 1 \pmod{\phi(119)}$ , where  $\phi(119)$  is the totient function of 119.

The prime factorization of 119 is  $119 = 7 \cdot 17$ , so  $\phi(119) = (7 - 1) \cdot (17 - 1) = 96$ .

We first find  $\gcd(7, 96)$ :

$$96/7 = 13 \text{ rem } 5 \rightarrow 96 = 13 \cdot 7 + 5$$

$$7/5 = 1 \text{ rem } 2 \rightarrow 7 = 1 \cdot 5 + 2$$

$$5/2 = 2 \text{ rem } 1 \rightarrow 5 = 2 \cdot 2 + 1$$

$$2/1 = 2 \text{ rem } 0 \rightarrow 2 = 2 \cdot 1 + 0$$

$\gcd(7, 96) = 1$ , so the modular inverse exists. We use the extended Euclidean algorithm to solve for  $d$  in  $7d \equiv 1 \pmod{96}$ . We do so by finding  $d$  such that  $7d - 96k = 1$  for some  $k$ .

$$5 - 2 \cdot 2 = 1$$

$$1 = 5 - 2 \cdot (7 - 1 \cdot 5) = 5 - 2 \cdot 7 + 2 \cdot 5 = 3 \cdot 5 - 2 \cdot 7$$

$$1 = 3 \cdot (96 - 7 \cdot 13) - 2 \cdot 7 = 3 \cdot 96 - 7 \cdot 39 - 2 \cdot 7 = 3 \cdot 96 - (39 + 2) \cdot 7 = 3 \cdot 96 - 41 \cdot 7$$

So  $k = 3, d = -41$ . The modular inverse  $7^{-1} \pmod{96}$  is  $-41 \pmod{96} = 55 \pmod{96}$ .

Now back to the problem:  $x \equiv 13^{55} \pmod{119}$ . We use the Chinese Remainder Theorem to solve for  $x$ :

1.  $x \equiv 13^{55} \pmod{7}$

2.  $x \equiv 13^{55} \pmod{17}$

1.  $13 \pmod{7} = 6$ , so  $x \equiv 13^{55} \pmod{7} = 6^{55} \pmod{7}$ . We reduce the exponent using Fermat's Little Theorem:

$6^{7-1} \equiv 1 \pmod{7}$ . So the powers of 6 mod 7 repeat every 6 steps, and we can reduce the exponent 55 using mod 6.  $55 = 9 \cdot 6 + 1$ , so  $55 \pmod{6} = 1$ . Thus  $6^{55} \pmod{7} = 6^1 \pmod{7} = 6 \pmod{7}$

2.  $13 \pmod{17} = 13$ , so no simplification here.  $13^{17-1} \equiv 1 \pmod{17}$ .  $55 \pmod{16} = 7$ . So  $13^{55} \pmod{17} = 13^7 \pmod{17}$ .  $13^7 \pmod{17} = 13^4 \cdot 13^2 \cdot 13 \pmod{17}$

$$13^2 = 169 \equiv 16 \pmod{17}$$

$$13^4 = (13^2)^2 = 16^2 = 256 \equiv 1 \pmod{17}$$

$$\text{So } 13^7 \pmod{17} = 1 \cdot 16 \cdot 13 \pmod{17} = 208 \pmod{17} = 4 \pmod{17}.$$

CRT applies because 7 and 17 are coprime.  $M = 119, M_1 = 17, M_2 = 7, y_1 = 17^{-1} \pmod{7}, y_2 = 7^{-1} \pmod{17}$ . Clearly,  $y_1 = y_2$ . We solve  $17y_1 \equiv 1 \pmod{17} \rightarrow 3y_1 \equiv 1 \pmod{17}$ .

$$3 \cdot 1 \equiv 3 \pmod{7}$$

...

$$3 \cdot 5 = 15 \equiv 1 \pmod{7}, \text{ so } y_1 = y_2 = 5.$$

$$\text{Then: } x = a_1 M_1 y_1 + a_2 M_2 y_2 \pmod{M} = 6 \cdot 17 \cdot 5 + 4 \cdot 7 \cdot 5 \pmod{119} = 650 \pmod{119} = 55 \pmod{119}$$

b)  $F: y^2 = x^3 - 2x + 2 \pmod{71}, P = (1, 1), Q = (4, 49)$ , so  $a = -2, b = 2, p = 71$ .

$$2P = (x_3, y_3):$$

$$\lambda = \frac{3 \cdot x_1^2 + a}{2y_1} = \frac{3 \cdot 1^2 - 2}{2 \cdot 1} = \frac{1}{2} = 1 \cdot 2^{-1} \pmod{71}$$

Find the modular inverse of 2 mod 71:

$$2k \equiv 1 \pmod{71}. \text{ Clearly, } 2 \cdot 36 = 72 \equiv 1 \pmod{71}. \text{ So, } 2^{-1} = 36. \text{ Then } \lambda = 36 \pmod{71}.$$

$$x_3 = \lambda^2 - 2x_1 = 36^2 - 2 \cdot 1 = 1294 \pmod{71} = 16 \pmod{71}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 = 36(1 - 16) - 1 = 36 \cdot -15 - 1 = -541 \pmod{71} = -541 + 8 \cdot 71 \pmod{71} = 27 \pmod{71}$$

$$\text{So, } 2P = (16, 27).$$

$$P + Q = (x_3, y_3):$$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{49 - 1}{4 - 1} = \frac{48}{3} = 16 \pmod{71}$$

$$x_3 = \lambda^2 - x_1 - x_2 = 16^2 - 1 - 4 = 256 - 5 = 251 \pmod{71} = 38 \pmod{71}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 = 16(1 - 38) - 1 = 16 \cdot -609 \pmod{71} = -593 \pmod{71} = -593 + 9 \cdot 71 \pmod{71} = 46 \pmod{71}$$

$$\text{So, } P + Q = (38, 46).$$

### 3 Information Theoretic Security

a) To avoid hand calculations, I coded the formulas required for this exercise. My code can be found in appendix A.0.2.

First we calculate the probability of each cipher text occuring using the formula:

$$p(C = c) = p(P = m) \cdot p(C = c | P = m)$$

$$P(C = 1) = p(K = k_1) \cdot p(P = b) + p(K = k_2) \cdot p(P = c) + p(K = k_3) \cdot p(P = d) + p(K = k_4) \cdot p(P = a) = \frac{1}{5} \cdot \frac{4}{15} + \frac{3}{10} \cdot \frac{1}{5} + \frac{1}{5} \cdot \frac{1}{5} + \frac{3}{10} \cdot \frac{1}{3} = 0.25\bar{3}$$

Repeating this for the other ciphertexts gives:

$$P(C = 2) = 0.25\bar{3}$$

$$P(C = 3) = 0.24\bar{6}$$

$$P(C = 4) = 0.24\bar{6}$$

Then, we calculate the probability of each cipher text occuring given the plaintext and key distributions using the encryption schema:

$$p(C = c) = \sum_{k: c \in \mathbb{C}(k)} p(K = k) \cdot p(P = d_k(c))$$

$$\begin{aligned} P(C = 1 | P = a) &= 0.3, & P(C = 2 | P = a) &= 0.3, & P(C = 3 | P = a) &= 0.2, & P(C = 4 | P = a) &= 0.2, \\ P(C = 1 | P = b) &= 0.2, & P(C = 2 | P = b) &= 0.2, & P(C = 3 | P = b) &= 0.3, & P(C = 4 | P = b) &= 0.3, \\ P(C = 1 | P = c) &= 0.3, & P(C = 2 | P = c) &= 0.3, & P(C = 3 | P = c) &= 0.2, & P(C = 4 | P = c) &= 0.2, \\ P(C = 1 | P = d) &= 0.2, & P(C = 2 | P = d) &= 0.2, & P(C = 3 | P = d) &= 0.3, & P(C = 4 | P = d) &= 0.3. \end{aligned}$$

Finally, we can calculate the probability of each plaintext conditioned on each ciphertext occurrence (see code in appendix):

$$p(P = m | C = c) = \frac{p(P = m) \cdot p(C = c | P = m)}{p(C = c)}$$

$$\begin{aligned} P(P = a | C = 1) &= 0.395, & P(P = a | C = 2) &= 0.395, & P(P = a | C = 3) &= 0.270, & P(P = a | C = 4) &= 0.270, \\ P(P = b | C = 1) &= 0.211, & P(P = b | C = 2) &= 0.211, & P(P = b | C = 3) &= 0.324, & P(P = b | C = 4) &= 0.324, \\ P(P = c | C = 1) &= 0.237, & P(P = c | C = 2) &= 0.237, & P(P = c | C = 3) &= 0.162, & P(P = c | C = 4) &= 0.162, \\ P(P = d | C = 1) &= 0.158, & P(P = d | C = 2) &= 0.158, & P(P = d | C = 3) &= 0.243, & P(P = d | C = 4) &= 0.243. \end{aligned}$$

b) We use the formula for conditional entropy from the slides:

$$H(M | C) = \sum_c p(C = c) \cdot H(M | C = c) = - \sum_m \sum_c p(C = c) \cdot p(M = m | C = c) \cdot \log_2 p(M = m | C = c).$$

to write the code in appendix A.0.3 and calculate the value. We find:

$$H(M | C) \approx 0.458$$

c) A cryptosystem has perfect secrecy if

$$p(P = m | C = c) = p(P = m)$$

for all plain texts  $m$  and ciphertexts  $c$ . It can be easily seen that this condition is already violated for  $P = a$  with  $p(P = a) = \frac{1}{3}$ . So, the system does not have perfect secrecy.

## A Code

This section lists the code written to help solve exercises 1 and 3.

### A.0.1 Exercise 1c:

Caesar cipher:

```
# works only on letters 'a' to 'z'
# shift using unicode values
def cshift(text, shift):
    result = []
    for char in text:
        if char.isalpha():
            if char.islower():
                result.append(chr((ord(char) - 97 + shift) % 26 + 97))
            else:
                result.append(chr((ord(char) - 65 + shift) % 26 + 65))
        else:
            result.append(char)
    return ''.join(result)

def shiftback(text, shift):
    return cshift(text, -shift)

with open('cipher.txt') as f:
    cipher = f.read().strip()

for i in range(26):
    print(i, shiftback(cipher, i))
```

### A.0.2 Exercise 3a:

```
M = [1/3, 4/15, 1/5, 1/5]
K = [1/5, 3/10, 1/5, 3/10]

m = ['a', 'b', 'c', 'd']

scheme = [
    [3, 1, 4, 2],
    [2, 4, 1, 3],
    [4, 2, 3, 1],
    [1, 3, 2, 4]
]

def prob_cipher(cipher, M, K, scheme):
    rows, cols = len(scheme), len(scheme[0])

    total = 0
    for i in range(rows):
        for j in range(cols):
            if scheme[i][j] == cipher:
                total += M[i] * K[j]

    return total

for i in range(4):
```

```

    print(f"P(C={i+1})^=" , prob_ciper(i+1, M, K, scheme))

def prob_cipher_given_plain(cipher , plain , K, scheme):
    rows, cols = len(scheme), len(scheme[0])

    total = 0

    col = ord(plain) - ord('a')

    # get only the column col
    col = [scheme[i][col] for i in range(rows)]

    for c in col:
        if c == cipher:
            total+= K[col.index(c)]

    return total

for char in m:
    for i in range(4):
        print(f"P(C={i+1}|P={char})^=" ,
            prob_cipher_given_plain(i+1, char , K, scheme))

def prob_plain_given_cipher(plain , cipher , M, scheme):
    rows, cols = len(scheme), len(scheme[0])

    total = 0

    p = ord(plain) - ord('a')

    prob = M[p]*prob_cipher_given_plain(cipher , plain , K, scheme)/
        prob_ciper(cipher , M, K, scheme)

    return prob

for char in m:
    for i in range(4):
        print(f"P(P={char}|C={i+1})^=" ,
            prob_plain_given_cipher(char , i+1, M, scheme))

```

### A.0.3 Exercise 3b:

```

import math

def conditional_entropy(M, K, scheme, msg):
    m_total = 1
    for char in msg:
        c_total = 0
        for i in range(4):
            pc = prob_ciper(i+1, M, K, scheme)
            ppc = prob_plain_given_cipher(char , i+1, M, scheme)

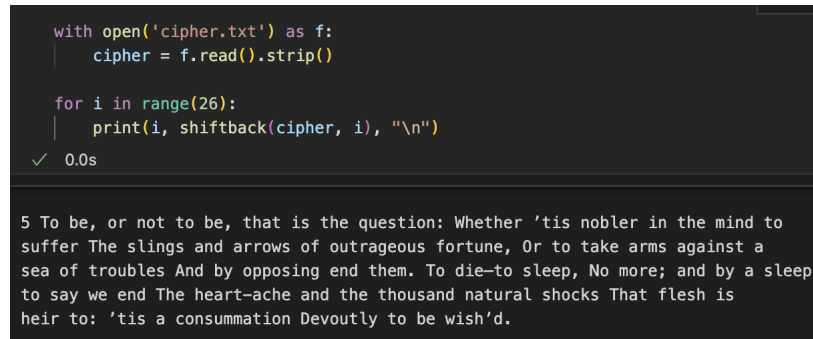
```

```
c_total += (pc * ppc * math.log(ppc, 2))

m_total *= c_total

return -1*m_total
```

## B Proofs



```
with open('cipher.txt') as f:
    cipher = f.read().strip()

for i in range(26):
    print(i, shiftback(cipher, i), "\n")
```

✓ 0.0s

5 To be, or not to be, that is the question: Whether 'tis nobler in the mind to  
suffer The slings and arrows of outrageous fortune, Or to take arms against a  
sea of troubles And by opposing end them. To die—to sleep, No more; and by a sleep  
to say we end The heart-ache and the thousand natural shocks That flesh is  
heir to: 'tis a consummation Devoutly to be wish'd.

Figure 1: Proof of code finding the plaintext using a shift of 5.