

# IDATT2503 - Cryptography Assignment 1

Edvard Berdal Eek

October 2025

## Problem 1

a) Calculate  $-99 \bmod 1001$

$$-99 \bmod 1001 = -99 - 1001 \cdot (-1) = \underline{902}$$

b) Calculate  $232 + 22 \cdot 77 - 18^3 \bmod 8$

I was unsure whether the task asks for  $(232 + 22 \cdot 77 - 18^3) \bmod 8$  or  $232 + 22 \cdot 77 - (18^3 \bmod 8)$  so I calculated both. It is my understanding that this conventionally asks for the remainder of everything before the mod operator.

i

$$\begin{aligned} & (232 + 22 \cdot 77 - 18^3) \bmod 8 \\ & (232 \bmod 8 + 22 \bmod 8 \cdot 77 \bmod 8 - 18^3 \bmod 8) \bmod 8 \\ & (0 + 6 \cdot 5 - 0) \bmod 8 = 22 \bmod 8 \\ & 22 \bmod 8 = \underline{6} \end{aligned}$$

ii

$$\begin{aligned} & 232 + 22 \cdot 77 - (18^3 \bmod 8) \\ & 232 + 22 \cdot 77 - 0 = \underline{1926} \end{aligned}$$

c) Determine if  $55 \equiv 77 \pmod{12}$

$$\begin{aligned} & 77 \pmod{12} = 5 \\ & 55 \pmod{12} = 7 \neq 5 \Rightarrow 55 \not\equiv 77 \pmod{12} \end{aligned}$$

## Problem 2

a) Write the multiplication table of the elements of  $\mathbb{Z}_{12}$ , excluding the 0 element.

I implemented a Python function that returns a matrix containing the modulo multiplication table in the range from a given number.

```
def mod_mul_table(num):  
    A = np.arange(1, num).reshape(-1, 1) @ np.arange(1, num).reshape(1, -1)  
    return A % num
```

[	[	1	2	3	4	5	6	7	8	9	10	11]
	[	2	4	6	8	10	0	2	4	6	8	10]
	[	3	6	9	0	3	6	9	0	3	6	9]
	[	4	8	0	4	8	0	4	8	0	4	8]
	[	5	10	3	8	1	6	11	4	9	2	7]
	[	6	0	6	0	6	0	6	0	6	0	6]
	[	7	2	9	4	11	6	1	8	3	10	5]
	[	8	4	0	8	4	0	8	4	0	8	4]
	[	9	6	3	0	9	6	3	0	9	6	3]
	[	10	8	6	4	2	0	10	8	6	4	2]
	[	11	10	9	8	7	6	5	4	3	2	1]]]

Figure 1: Modulo multiplication table  $n=12$

b) Which integers have a multiplicative inverse modulo 12?

Using **numpy.argwhere** and the previous function I created a new function that returns the indices where the elements is equal to 1. I added 1 so that these indices represent the integers a and b.

```
def mul_inverses(n):
    A = mod_mul_table(n)
    return np.argwhere(A == 1) + 1
```

[	[	1	1]
	[	5	5]
	[	7	7]
	[	11	11]]]

Figure 2: Integers  $[a \ b]$  where  $ab \equiv 1 \pmod{12}$

c) Do the same for  $\mathbb{Z}_{11}$ . Which numbers have multiplicative inverse mod 11?

Images of output where  $n=11$ .

[	1	2	3	4	5	6	7	8	9	10]
[	2	4	6	8	10	1	3	5	7	9]
[	3	6	9	1	4	7	10	2	5	8]
[	4	8	1	5	9	2	6	10	3	7]
[	5	10	4	9	3	8	2	7	1	6]
[	6	1	7	2	8	3	9	4	10	5]
[	7	3	10	6	2	9	5	1	8	4]
[	8	5	2	10	7	4	1	9	6	3]
[	9	7	5	3	1	10	8	6	4	2]
[10	9	8	7	6	5	4	3	2	1]	

Figure 3: Modulo multiplication table  $n=11$

[	1	1]
[	2	6]
[	3	4]
[	4	3]
[	5	9]
[	6	2]
[	7	8]
[	8	7]
[	9	5]
[10	10]	

Figure 4: Integers  $[a \ b]$  where  $ab \equiv 1 \pmod{11}$

- d) Find the multiplicative inverse to 11 modulo 29, by trial and error, ie. just try different values.

I now implemented a function using a for-loop iterating through the numbers from 0 to  $n$  and returning the multiplicative inverse to  $a$  if it finds one.

```
def mul_inverse(a, n):
    for i in range(n):
        if (a*i % n == 1):
            return i
```

I could have used the multiplication table from previous tasks rather than just trying different values, but this solution was more fitting for how the task was worded.

- e) Formulate a condition for  $a$  to have a multiplicative inverse modulo  $n$ . Hint: It involves the factorisations of  $a$  and  $n$ .

$$ab \bmod n = 1 \Rightarrow ab - kn = 1$$

$$ab - kn = ax + ny, x = b, y = -k$$

Here we can use **Bezout's identity** which states that for any integer  $a$  and  $n$  there exists integers  $x$  and  $y$  such that the result is the greatest common denominator of  $a$  and  $n$ .

$$ax + ny = \gcd(a, n)$$

Therefore, the condition for  $a$  to have a multiplicative inverse  $b$  is that its greatest common denominator with  $n$  is 1.

$$ab \equiv 1 \pmod{n} \Leftrightarrow \gcd(a, n) = 1$$

### Problem 3 - Affine Ciphers

- a) Write  $e_k$  as a permutation, i.e. the sequence of letters we get when encrypting a, b, c etc.

I made the following Python function to encrypt letters:

```
def encrypt(x):
    num = (3*x + 11) % 26
    return chr(num + ord('A'))
```

This gave these permutations when iterating through the alphabet:

```
Task a) e_k permutation 'abcdefghijklmnopqrstuvwxyz':
{'a': 'L', 'b': 'O', 'c': 'R', 'd': 'U', 'e': 'X', 'f': 'A', 'g': 'D', 'h': 'G', 'i': 'J', 'j': 'M', 'k': 'P', 'l': 'S', 'm': 'V',
'n': 'Y', 'o': 'B', 'p': 'E', 'q': 'H', 'r': 'K', 's': 'N', 't': 'Q', 'u': 'T', 'v': 'W', 'w': 'Z', 'x': 'C', 'y': 'F', 'z': 'I'}
```

Figure 5: Permutations

- b) Use  $e_k$  to encrypt the message  $m = \text{'alice'}$

```
Task b) Encrypted 'alice':
LSJRX
```

Figure 6: Encrypted 'alice'

- c) Find the inverse of  $e_k$

I chose to first find the formula as I could use this to display the permutations later.

$$y \equiv (3 \cdot x + 11) \pmod{26} \Rightarrow 3x \equiv (y - 11) \pmod{26}$$

Now we have to multiply with the multiplicative inverse of  $3 \pmod{26}$  to find  $x$ , as we can't just divide by 3 in modular arithmetic. I used the formula from the earlier to find the multiplicative inverse  $b = 9$  where  $3 \cdot b \equiv 1 \pmod{26}$ .

$$d_k(y) = 9(y - 11) \pmod{26}$$

```
Task c) inverse permutation 'ABCDEFGHIJKLMNOPQRSTUVWXYZ':
{'A': 'f', 'B': 'o', 'C': 'x', 'D': 'g', 'E': 'p', 'F': 'y', 'G': 'h', 'H': 'q', 'I': 'z', 'J': 'i', 'K': 'r', 'L': 'a', 'M': 'j',
 'N': 's', 'O': 'b', 'P': 'k', 'Q': 't', 'R': 'c', 'S': 'l', 'T': 'u', 'U': 'd', 'V': 'm', 'W': 'v', 'X': 'e', 'Y': 'n', 'Z': 'w'}
```

Figure 7: Inverse permutations

d) Use the inverse d to decrypt  $c = \text{RBKKXRQ}$

```
def decrypt(y):
    num = 9*(y - 11) % 26
    return chr(num + ord('a'))
```

```
Task d) Decrypted 'RBKKXRQ':
correct
```

Figure 8: Decrypted

e) How secure is this cipher, compared to a rotation cipher? Consider both brute force and known plaintext attacks.

i **Brute Force**

Affine has an advantage when it comes to brute force as it requires to attempt all possible values of  $a$  and  $b$ . While in rotation cipher one only has to try 26 different rotation keys.

ii **Known Plaintext Attacks**

If an attacker knows at least two letters, than they can easily solve for  $a$  and  $b$ , and therefor know the key. However, this is still better than rotation cipher where knowing only one letter reveals the key.

f) How many legal keys does are there for this choice of formula and alphabet?

A legal key  $k(a, b)$  has the following requirements for  $a$ :

$$a \in \mathbb{Z}_{26}, \gcd(a, 26) = 1$$

$$a \in \{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}$$

and for  $b$  any number from 0 to 25:

$$b \in \mathbb{Z}_{26}$$

There are 12 values for  $a$  and 26 for  $b$  meaning that the total amount of keys in the key space is 312.

## Problem 4 - Brute Force Affine Cipher Decryption

To find the decrypted message using brute force I outputted the text for all possible keys using the functions below.

```

def brute_force_decrypt(cipher):
    results = []
    for a in range(len(alphabet)):
        a_inv = mul_inverse(a, 26)
        if (a_inv == None):
            continue
        for b in range(len(alphabet)):
            res = decrypt_affine(cipher, a_inv, b)
            results.append({
                "key": (a, b),
                "a_inv": a_inv,
                "decrypted": res
            })
    return results

def decrypt_affine(cipher, a_inv, b):
    res = ""
    for char in cipher:
        if ('A' <= char <= 'Z'):
            y = char_to_pos(char)
            x = (a_inv * (y - b)) % 26
            res += pos_to_char(x)
    return res

```

This led me to the message "noen ganger er det all right" with the key  $k(15, 4)$ .

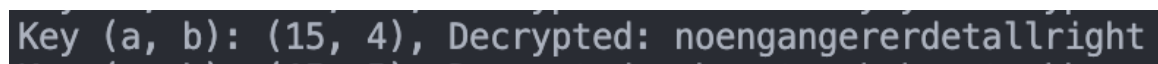


Figure 9: Brute Force result

## Problem 5 - Simple Substitution Cipher

Inserting the text in dCode i could extract the letter frequencies and sort the letters accordingly:  
**[DI BKCVOQWTS AELYPUMFRGJXHNZ]**

Leading me to believe that D is the replacement for space.

I then compared it with the english letter frequencies and unsuccessfully tried to decrypt:

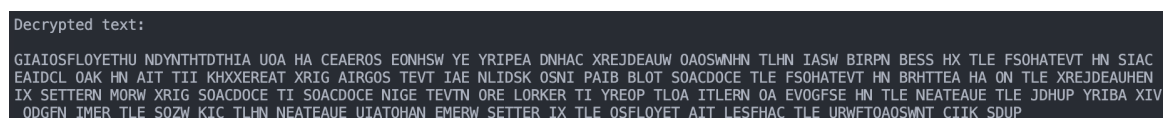


Figure 10: Unsolved decryption

I then looked at the bigrams in the text, revealed that the top three bigrams were either leading or ending with a space (**D**). The fourth did, not and was therefore most likely the bigram **th**. I

now had t and h. After this i attempted to change letters with those near eachother. This allowed me to see som words start to make sense, so i continued doing this until it was solved. With the substitution alphabet being:

[ **ETNAOSLIRHUGCVYPMBKWXDQFJZ**]

```
Decrypted text:
MONOALPHABETIC SUBSTITUTION CAN IN GENERAL EASILY BE BROKEN USING FREQUENCY ANALYSIS THIS ONLY WORKS WELL IF THE PLAINTEXT IS LONG
ENOUGH AND IS NOT TOO DIFFERENT FROM NORMAL TEXT ONE SHOULD ALSO KNOW WHAT LANGUAGE THE PLAINTEXT IS WRITTEN IN AS THE FREQUENCIES
OF LETTERS VARY FROM LANGUAGE TO LANGUAGE SOME TEXTS ARE HARDER TO BREAK THAN OTHERS AN EXAMPLE IS THE SENTENCE THE QUICK BROWN FOX
JUMPS OVER THE LAZY DOG THIS SENTENCE CONTAINS EVERY LETTER OF THE ALPHABET NOT HELPING THE CRYPTANALYST GOOD LUCK
```

Figure 11: Solution