Υπολογιστική Κρυπτογραφία
Πρώτη σειρά ασκήσεων

1 Άσκηση 1

Έστω το affine cipher: $c = \text{Enc}((a, b), m) = (a \cdot x + b) \bmod 26$. Υποθέτουμε ότι ο αντίπαλος μπορεί να επιλέξει δύο μηνύματα m_1, m_2 και να αποκτήσει τις κρυπτογραφήσεις τους c_1, c_2 (CPA – Chosen Plaintext Attack).

1. Πώς μπορεί να χρησιμοποιήσει αυτή τη δυνατότητα ώστε να σπάσει το κρυπτοσύστημα; Πώς θα επιλέξει τα m_1, m_2;

2. Έστω ότι για μεγαλύτερη ασφάλεια αποφασίζουμε να χρησιμοποιήσουμε διπλή κρυπτογράφηση με διαφορετικά κλειδιά, δηλαδή:

   $$\text{Enc}(((a_1, b_1), (a_2, b_2)), m) = \text{Enc}((a_2, b_2), \text{Enc}((a_1, b_1), m))$$

   Πώς επηρεάζεται ο χώρος των κλειδιών σε μια εξαντλητική αναζήτηση; Είναι το νέο κρυπτοσύστημα πιο ασφαλές;

1. Για να σπάσει το affine cipher, ο αντίπαλος μπορεί να εκμεταλλευτεί τη δυνατότητα επιλέγοντας δύο συγκεκριμένα μηνύματα m_1 και m_2 ώστε να παράγει εξισώσεις για να λύσει τα a και b. Ορίζοντας $m_1 = 0$ και $m_2 = 1$, θα έχει:

   $$c_1 = (a \cdot 0 + b) \bmod 26 = b \quad \text{και} \quad c_2 = (a \cdot 1 + b) \bmod 26 = a + b \bmod 26.$$

   Επομένως, μπορεί να υπολογίσει $a = (c_2 - c_1) \bmod 26$ και μετά να αντικαταστήσει σε μία από τις εξισώσεις για να βρει το b.

2. Στην περίπτωση διπλής κρυπτογράφησης με διαφορετικά κλειδιά $((a_1, b_1), (a_2, b_2))$, ο χώρος των κλειδιών αυξάνεται σε $26 \cdot 12 \cdot 26 \cdot 12 = 97344$ πιθανά ζεύγη, ωστόσο η χρήση διπλής κρυπτογράφησης με affine cipher δεν προσφέρει ιδιαίτερη ασφάλεια. Το αποτέλεσμα μιας διπλής affine είναι πάλι affine, άρα ένας επιτιθέμενος μπορεί να βρει το ισοδύναμο ενός a, b που αποκρυπτογραφεί το μήνυμα.

Άσκηση 2

Παρακάτω δίνεται η υλοποίηση σε Python ενός προγράμματος για αποκρυπτογράφηση κειμένων κρυπτογραφημένων με τον κώδικα Vigenère. Το πρόγραμμα δοκιμάζει διάφορα πιθανά κλειδιά και υπολογίζει τον δείκτη σύμπτωσης (IC) για να εντοπίσει τα 5 πιο πιθανά plaintexts και τα αντίστοιχα κλειδιά. oΤο πρόγραμμά σας θα πρέπει να εξάγει και τον δείκτη σύμπτωσης καθενός plaintext. Να εξηγήσετε τις βασικές ιδέες που χρησιμοποιήσατε στον κώδικά σας.

Κώδικας

```
import math
import sys

letter_mappings = { 'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5, 'f': 6, 'g': 7, 'h': 8,
                    'i': 9, 'j': 10, 'k': 11, 'l': 12, 'm': 13, 'n': 14, 'o': 15, 'p': 16,
                    'q': 17, 'r': 18, 's': 19, 't': 20, 'u': 21, 'v': 22, 'w': 23, 'x': 24,
                    'y': 25, 'z': 26 }
english_freqs =    {'a': 8.5, 'b': 2.07, 'c': 4.54, 'd': 3.38, 'e': 11.16, 'f': 1.81, 'g': 2.47, 'h': 3,
                        'i': 7.54, 'j': 0.2, 'k': 1.1, 'l': 5.49, 'm': 3.01, 'n': 6.65, 'o': 7.16, 'p': 3.17,
                        'q': 0.2, 'r': 7.58, 's': 5.74, 't': 6.95, 'u': 3.63, 'v': 1.01, 'w': 1.29, 'x': 0.29,
                        'y': 1.78, 'z': 0.27}


def split(key, input_text):
    n = len(input_text)
    groups = [[] for _ in range(key)]
    for i in range(0, n):
        groups[i % key].append(input_text[i])
    return groups

def index_of_coincidence(text_list, n):
    letter_freqs = {}

    for letter in text_list:
        if letter in letter_freqs:
            letter_freqs[letter] += 1
        else:
            letter_freqs[letter] = 1

    ic = 0
    for letter in letter_freqs:
        ic += (letter_freqs[letter] / n) * ((letter_freqs[letter] - 1) / (n - 1))

    return ic

def caesar_decrypt(group):
```

```
    letter_freqs = {}

    group = [char.lower() for char in group]

    for letter in group:
        if letter in letter_freqs:
            letter_freqs[letter] += 1
        else:
            letter_freqs[letter] = 1

    def shift(letter_freqs, n):
        shifted_freqs = {}
        for letter, frequency in letter_freqs.items():
            shifted_letter = chr(((ord(letter) - ord('a') + n) % 26) +
                ord('a'))
            shifted_freqs[shifted_letter] = frequency

        return shifted_freqs

    min_entropy = 2**10
    n = 0
    for i in range(1, 26):
        shifted_by_n_freqs = shift(letter_freqs, i)
        current_entropy = 0
        for letter in shifted_by_n_freqs:
            current_entropy += (shifted_by_n_freqs[letter] /
                len(group)) * math.log10(english_freqs[letter])
        current_entropy *= -1
        if current_entropy < min_entropy:
            min_entropy = current_entropy
            n = i

    c = group[0]
    p = chr(((ord(group[0]) - ord('a') + n) % 26) + ord('a'))

    key_char_map = (letter_mappings[c] - letter_mappings[p]) % 26
    key_char = chr(ord('a') + key_char_map)

    return key_char


def vigenere_decrypt():
    # Check if the input was provided through the command line
    if len(sys.argv) < 2:
        print("Please provide the encrypted text as a command-line
            argument.")
        sys.exit(1)

    # Retrieve the encrypted text from the command line argument
    input_text = sys.argv[1]

    # Filter the text to only keep alphabetic characters and convert
        to lowercase
    input_text_letters = [char.lower() for char in input_text if
        char.isalpha()]

    key = 2
```

```
    n = len(input_text_letters)
    groups = []
    ic_counter = 0
    while ic_counter < 5:
        decrypted_text = []
        mean_ic = 0

        # Split the input text into groups based on the key
        groups = split(key, input_text_letters)
        for group in groups:
            if len(group) > 1:
                mean_ic += index_of_coincidence(group, len(group))

        # Calculate the mean index of coincidence
        mean_ic /= key

        if mean_ic >= 0.06 and mean_ic <= 0.07:
            vigenere_key = ""
            # Decrypt each group with Caesar cipher to get part of the
                Vigenère key
            for group in groups:
                vigenere_key += caesar_decrypt(group)

            key_counter = 0
            # Decrypt the entire text using the Vigenère key
            for i in range(0, len(input_text)):
                if key_counter == len(vigenere_key):
                    key_counter = 0
                if input_text[i].isalpha():
                    dec_text.append(chr((letter_mappings[input_text[i].lower()]
                        - letter_mappings[vigenere_key[key_counter]])
                        % 26 + ord('a')))
                    key_counter += 1
                else:
                    dec_text.append(input_text[i])


            print()
            decr_text_str = ''.join(dec_text)
            print(vigenere_key + '\n\n' + decr_text_str + '\n' + '\n'
                + '\n')
            print(mean_ic)

            ic_counter += 1

        key += 1
        if key == n:
            break

vigenere_decrypt()
```

2 Δομή του Κώδικα

Ο κώδικας περιλαμβάνει αρκετές συναρτήσεις για να αναλύσει το κείμενο και να βρει το σωστό κλειδί για την αποκρυπτογράφηση. Ακολουθεί αναλυτική εξήγηση

για κάθε συνάρτηση.

2.1 Λεξικό `letter_mappings`

Αυτό το λεξικό αντιστοιχεί κάθε γράμμα του αλφαβήτου σε έναν αριθμό από το 1 μέχρι το 26, το οποίο επιτρέπει την εύκολη μετατροπή των χαρακτήρων σε αριθμούς για περαιτέρω υπολογισμούς.

2.2 Συνάρτηση `split`

Η συνάρτηση `split` χωρίζει το κείμενο σε ομάδες με βάση το μήκος του κλειδιού. Για παράδειγμα, αν το κλειδί είναι 3, το κείμενο θα χωριστεί σε 3 ομάδες, με τα γράμματα του κειμένου να κατανέμονται κυκλικά στις ομάδες.

2.3 Συνάρτηση `index_of_coincidence`

Η συνάρτηση `index_of_coincidence` υπολογίζει τον δείκτη συμφωνίας (IC) για μια ομάδα χαρακτήρων. Ο δείκτης συμφωνίας μετράει πόσο συχνά επαναλαμβάνονται τα ίδια γράμματα σε ένα κείμενο. Είναι χρήσιμος για την ανάλυση του κλειδιού στην κρυπτογράφηση Vigenère.

2.4 Συνάρτηση `caesar_decrypt`

Η συνάρτηση `caesar_decrypt` αναλύει το Caesar cipher που εφαρμόζεται σε κάθε ομάδα του κειμένου. Χρησιμοποιεί τη συχνότητα των γραμμάτων για να βρει την καλύτερη δυνατή μετατόπιση και υπολογίζει το κλειδί του Vigenère.

2.5

section*Μέτρηση Συχνότητας Κάθε Γράμματος

```
for letter in group:
    if letter in letter_freqs:
        letter_freqs[letter] += 1
    else:
        letter_freqs[letter] = 1
```

Αυτή η επανάληψη υπολογίζει τον αριθμό εμφανίσεων κάθε γράμματος στο `group` (το κρυπτογραφημένο κείμενο) και αποθηκεύει το αποτέλεσμα στο `letter_freqs`. Για παράδειγμα, αν `group = "ABCBA"`, μετά την επανάληψη αυτή, το `letter_freqs` θα είναι: `{'A': 2, 'B': 2, 'C': 1}`.

Ορισμός της Συνάρτησης `shift`

```
def shift(letter_freqs, n):
    shifted_freqs = {}
    for letter, frequency in letter_freqs.items():
```

```
        shifted_letter = chr(((ord(letter) - ord('A') + n) % 26) + ord('A'))
        shifted_freqs[shifted_letter] = frequency
    return shifted_freqs
```

Η συνάρτηση `shift` λαμβάνει τις συχνότητες των γραμμάτων και εφαρμόζει μια μετατόπιση του Καίσαρα κατά n θέσεις. Για κάθε γράμμα, υπολογίζει το μετατοπισμένο γράμμα προσθέτοντας n θέσεις στο αλφάβητο και "περιστρέφοντας" αν χρειάζεται. Για παράδειγμα, αν `letter = "A"` και $n = 3$, το μετατοπισμένο γράμμα θα είναι `"D"`.

Εύρεση της Μετατόπισης με Ελάχιστη Εντροπία

```
min_entropy = 2**10
n = 0
for i in range(1, 26):
    shifted_by_n_freqs = shift(letter_freqs, i)
    current_entropy = 0
    for letter in shifted_by_n_freqs:
        current_entropy+=(shifted_by_n_freqs[letter/len(group))*
        *math.log10(english_freqs[letter])
    current_entropy *= -1
    if current_entropy < min_entropy:
        min_entropy = current_entropy
        n = i
```

- Η `min_entropy` αρχικοποιείται σε μια μεγάλη τιμή (στην προκειμένη 2^{10}) και αποθηκεύει την ελάχιστη εντροπία που έχει βρεθεί μέχρι στιγμής.
- Το βρόχο από 1 έως 25 εξετάζει όλες τις πιθανές μετατοπίσεις του Καίσαρα (η μετατόπιση 0 θα έδινε το αρχικό κείμενο, το οποίο πιθανότατα θα έχει υψηλή εντροπία για ένα κρυπτογραφημένο μήνυμα).

Υπολογισμός Εντροπίας: Για κάθε μετατόπιση i, ο κώδικας δημιουργεί το λεξικό `shifted_by_n_freqs` με τις μετατοπισμένες συχνότητες γραμμάτων. Στη συνέχεια, υπολογίζει την εντροπία `current_entropy` για το μετατοπισμένο κείμενο, η οποία μετρά το πόσο "τυχαία" ή "ακανόνιστη" είναι η κατανομή συχνοτήτων σε σχέση με την `english_freqs`. Ο τύπος της εντροπίας είναι:

$$\text{current\_entropy} = -\sum \left(\frac{\text{shifted\_by\_n\_freqs[letter]}}{\text{len(group)}} \right) \cdot \log_{10}(\text{english\_freqs[letter]})$$

Αυτός ο τύπος ουσιαστικά "τιμωρεί" τις μετατοπίσεις στις οποίες οι συχνότητες του μετατοπισμένου κειμένου αποκλίνουν σημαντικά από τις τυπικές αγγλικές συχνότητες.

Προσδιορισμός του Χαρακτήρα Κλειδί Βάσει της Μετατόπισης

```
c = group[0]
p = chr(((ord(group[0]) - ord('A') + n) % 26) + ord('A'))
key_char_map = (letter_mappings[c] - letter_mappings[p]) % 26
key_char = chr(ord('A') + key_char_map)
return key_char
```

Ο χαρακτήρας `group[0]`, δηλαδή το πρώτο γράμμα του κρυπτογραφημένου κειμένου, ονομάζεται `c`. Με την υπολογισμένη μετατόπιση n, ο κώδικας υπολογίζει ποιο γράμμα θα αντιστοιχεί στον αποκρυπτογραφημένο χαρακτήρα `p`. Η μεταβλητή `key_char_map` υπολογίζει τη διαφορά στις αντιστοιχίσεις τους, ώστε να βρει την πραγματική μετατόπιση του Καίσαρα. Η συνάρτηση τελικά επιστρέφει το `key_char`, ο οποίος αντιπροσωπεύει τον χαρακτήρα κλειδί για τη συγκεκριμένη μετατόπιση.

2.6 Συνάρτηση `vigenere_decrypt`

Η συνάρτηση `vigenere_decrypt` είναι υπεύθυνη για την αποκρυπτογράφηση του κειμένου. Διαβάζει το κείμενο, το φιλτράρει για να κρατήσει μόνο τα γράμματα, και στη συνέχεια προσπαθεί διαφορετικά μήκη κλειδιού για να βρει το σωστό κλειδί Vigenère χρησιμοποιώντας τη συνάρτηση `split`, την ανάλυση συχνοτήτων και τον υπολογισμό του δείκτη συμφωνίας (IC). Αποκρυπτογραφεί το κείμενο και εμφανίζει τα αποτελέσματα.

3 Άσκηση 3

1. Σε ένα κρυπτοσύστημα που διαθέτει τέλεια μυστικότητα, είναι αναγκαίο κάθε κλειδί να επιλέγεται με την ίδια πιθανότητα; Είναι αναγκαίο οι χώροι να είναι ισοπληθικοί; Αποδείξτε τους ισχυρισμούς σας. 2. Να αποδείξετε ότι οι παρακάτω προτάσεις είναι ισοδύναμες με τη συνθήκη τέλειας μυστικότητας του Shannon: (i) $\forall x \in M, y \in C : \Pr[C = y] = \Pr[C = y | M = x]$ (ii) $\forall x_1, x_2 \in M, y \in C : \Pr[C = y | M = x_1] = \Pr[C = y | M = x_2]$

1. **Αναγκαιότητα της Ισοπίθανης Επιλογής Κλειδιού και Ισοπληθικότητας Χώρων:**

Απάντηση:

Η τέλεια μυστικότητα σε ένα κρυπτοσύστημα σημαίνει ότι το κρυπτοκείμενο δεν παρέχει καμία πληροφορία για το μήνυμα. Αυτό επιτυγχάνεται όταν για κάθε μήνυμα, η πιθανότητα εμφάνισης οποιουδήποτε κρυπτοκειμένου είναι η ίδια για κάθε μήνυμα.

Κλειδιά με Ίσες Πιθανότητες:

Όταν εξετάζουμε αν είναι αναγκαίο τα κλειδιά να επιλέγονται με την ίδια πιθανότητα, η απάντηση είναι **όχι**. Ένα κρυπτοσύστημα μπορεί να έχει τέλεια μυστικότητα ακόμα κι αν τα κλειδιά δεν επιλέγονται με την ίδια πιθανότητα. Αυτό μπορεί να συμβαίνει, εφόσον η πιθανότητα εμφάνισης του πρώτου bit του κλειδιού είναι η ίδια, ανεξαρτήτως του κλειδιού.

Για παράδειγμα: Ας θεωρήσουμε τα παρακάτω κλειδιά και πιθανότητες:

- $\Pr[K = 00] = \frac{1}{8}$
- $\Pr[K = 01] = \frac{3}{8}$
- $\Pr[K = 10] = \frac{2}{8}$
- $\Pr[K = 11] = \frac{2}{8}$

Αν και τα κλειδιά δεν έχουν ίσες πιθανότητες, το πρώτο bit των κλειδιών έχει κατανομή $\Pr[K_1 = 0] = \frac{1}{2}$ και $\Pr[K_1 = 1] = \frac{1}{2}$, οπότε το κρυπτοσύστημα μπορεί να παραμείνει ανώνυμο και να ικανοποιεί τη συνθήκη της τέλειας μυστικότητας.

Ισοπληθικοί Χώροι:

Όσον αφορά την ισοπληθικότητα των χώρων, **είναι αναγκαίο** οι χώροι $|M|$, $|C|$ και $|K|$ να πληρούν την ανισότητα:

$$|M| \leq |C| \leq |K|$$

Αν το μέγεθος του χώρου του κρυπτοκείμενου $|C|$ είναι μεγαλύτερο από το μέγεθος του χώρου των κλειδιών $|K|$, τότε η συνθήκη της τέλειας μυστικότητας παραβιάζεται, καθώς δεν μπορεί να υπάρξει μια 1-1 αντιστοιχία μεταξύ μηνυμάτων και κρυπτοκειμένων.

2. **Ισοδυναμία Προτάσεων με τη Συνθήκη Τέλειας Μυστικότητας του Shannon:**

i) Απόδειξη ότι η συνθήκη i ισοδυναμεί με την τέλεια μυστικότητα:

Η πρόταση i δηλώνει ότι η πιθανότητα του κρυπτοκειμένου C να είναι ίση με το y είναι η ίδια, ανεξάρτητα από το μήνυμα M. Δηλαδή:

$$\forall x \in M, y \in C : \Pr[C = y] = \Pr[C = y | M = x]$$

Αυτό σημαίνει ότι η κατανομή των κρυπτοκειμένων δεν εξαρτάται από το μήνυμα, άρα το κρυπτοκείμενο C δεν παρέχει καμία πληροφορία για το μήνυμα M.

Αυτή ακριβώς είναι η προϋπόθεση για τη τέλεια μυστικότητα του Shannon. Συνεπώς, αν ισχύει η πρόταση i, τότε το κρυπτοσύστημα έχει τέλεια μυστικότητα.

ii) Απόδειξη ότι η συνθήκη ii ισοδυναμεί με την τέλεια μυστικότητα:

Η πρόταση ii δηλώνει ότι για οποιαδήποτε δυο μηνύματα x_1 και x_2 από τον χώρο των μηνυμάτων M και για οποιοδήποτε κρυπτοκείμενο y, η πιθανότητα του κρυπτοκειμένου C να είναι ίση με το y, δεδομένου το μήνυμα x_1, είναι ίση με την πιθανότητα του κρυπτοκειμένου C να είναι ίση με το y, δεδομένου το μήνυμα x_2:

$$\forall x_1, x_2 \in M, y \in C : \Pr[C = y | M = x_1] = \Pr[C = y | M = x_2]$$

Αυτό σημαίνει ότι για οποιαδήποτε δυο μηνύματα x_1 και x_2, η κατανομή των κρυπτοκειμένων είναι ίδια, άρα το κρυπτοκείμενο C δεν παρέχει καμία πληροφορία για το ποιο από τα δύο μηνύματα έχει κρυπτογραφηθεί.

Αυτή είναι ακριβώς η συνθήκη της τέλειας μυστικότητας του Shannon, που απαιτεί ότι το κρυπτοκείμενο δεν πρέπει να παρέχει καμία πληροφορία για το μήνυμα, ανεξάρτητα από το ποιο μήνυμα έχει επιλεγεί. Συνεπώς, αν ισχύει η πρόταση ii, τότε το κρυπτοσύστημα έχει τέλεια μυστικότητα.

Ισοδυναμία:

Από τις παραπάνω αποδείξεις, βλέπουμε ότι και οι δύο προτάσεις i και ii οδηγούν στην ίδια συνθήκη: Το κρυπτοκείμενο C δεν πρέπει να παρέχει καμία πληροφορία για το μήνυμα M. Επομένως, οι προτάσεις i και ii είναι ισοδύναμες με τη συνθήκη τέλειας μυστικότητας του Shannon.

Άσκηση 4

Η Αλίκη χρησιμοποιεί το one-time pad και συνειδητοποιεί ότι όταν το κλειδί της είναι το k = 0λ (όλο μηδενικά) τότε Enc(k, m) = m. Δηλαδή το μήνυμα στέλνεται χωρίς καμία κρυπτογράφηση! Για να αντιμετωπίσει το παραπάνω πρόβλημα, τροποποιεί τον αλγόριθμο παραγωγής κλειδιών του one-time pad ώστε το κλειδί να επιλέγεται ομοιόμορφα από το $\{0,1\}^\lambda \setminus \{0^\lambda\}$. Δηλαδή το κλειδί μπορεί να είναι οποιαδήποτε συμβολοσειρά λ ψηφίων χωρίς όμως να λαμβάνεται υπόψιν η συμβολοσειρά που αποτελείται από λ μηδενικά. Παραμένει το τροποποιημένο αυτό one-time pad τέλεια ασφαλές; Να αιτιολογήσετε την απάντησή σας.

Απόδειξη για την Τέλεια Μυστικότητα στο Τροποποιημένο One-Time Pad

Για να υπάρχει τέλεια μυστικότητα, πρέπει να ισχύει η συνθήκη:

$$|M| \leq |C| \leq |K|$$

όπου:

- M είναι ο χώρος των μηνυμάτων,

- C είναι ο χώρος των κρυπτοκειμένων,
- K είναι ο χώρος των κλειδιών.

Στο τροποποιημένο one-time pad ισχύουν τα εξής:

- $|M| = 2^\lambda$, καθώς το μήνυμα είναι συμβολοσειρά λ bit.
- $|\mathcal{K}| = 2^\lambda - 1$, αφού εξαιρείται η συμβολοσειρά που αποτελείται από λ μηδενικά από τον χώρο των κλειδιών.

Από την παραπάνω σχέση βλέπουμε ότι:

$$|\mathcal{K}| \leq |M|$$

Επομένως, το τροποποιημένο one-time pad δεν πληροί την προϋπόθεση για τέλεια μυστικότητα, καθώς το μέγεθος του χώρου των κλειδιών είναι μικρότερο από το μέγεθος του χώρου των μηνυμάτων.

Άσκηση 5

Έστω ότι c είναι το κρυπτοκείμενο, δηλαδή:

$$c = (k \cdot m) \mod p$$

Αυτό ισοδυναμεί με:

$$k \cdot m \equiv c \pmod{p}$$

Δεδομένου ότι $\gcd(k, p) = 1$ (καθώς p είναι πρώτος και $k < p$), η εξίσωση έχει λύση ως προς το m. Συγκεκριμένα:

$$m = k^{-1} \cdot c \mod p$$

Έτσι, η διαδικασία αποκρυπτογράφησης δίνεται από:

$$\mathrm{Dec}(k, c) = k^{-1} \cdot c \mod p$$

2. Απόδειξη ότι η Αποκρυπτογράφηση Δίνει Σωστό Αποτέλεσμα

Θα αποδείξουμε ότι η αποκρυπτογράφηση επιστρέφει το σωστό μήνυμα, δηλαδή ότι:

$$m = \mathrm{Dec}(k, c)$$

όπου $c = (k \cdot m) \mod p$. Από την προηγούμενη εξίσωση, έχουμε:

$$\mathrm{Dec}(k, c) = k^{-1} \cdot c \mod p = k^{-1} \cdot (k \cdot m) \mod p$$

Αφού k^{-1} είναι το αντίστροφο του k στο $\mathbb{Z}_p^*$, και $k^{-1} < p$, τότε ισχύει:

$$\mathrm{Dec}(k, c) = (k^{-1} \cdot k \cdot m) \mod p = m \mod p = m$$

Έτσι, αποδεικνύεται ότι η διαδικασία αποκρυπτογράφησης επιστρέφει το σωστό μήνυμα m.

3. Απόδειξη Τέλειας Μυστικότητας

Για να δείξουμε ότι το κρυπτοσύστημα είναι τέλεια ασφαλές, θα χρησιμοποιήσουμε την αναγκαία και ικανή συνθήκη για τέλεια μυστικότητα που είδαμε στο μάθημα. Έστω ότι το κρυπτοσύστημα έχει:

$$|M| = |C| = |K|$$

Το σύστημα έχει τέλεια μυστικότητα αν και μόνο αν ισχύουν οι εξής δύο συνθήκες:

1. Για κάθε $x \in M$, $y \in C$, υπάρχει μοναδικό $k \in K$ τέτοιο ώστε $\text{Enc}(k, x) = y$.
2. Κάθε κλειδί επιλέγεται με την ίδια πιθανότητα, δηλαδή $\frac{1}{|K|}$.

Στο δικό μας κρυπτοσύστημα, έχουμε $M = K = \mathbb{Z}_p^*$, και η κρυπτογράφηση ορίζεται ως:

$$\text{Enc}(k, x) = (k \cdot x) \mod p$$

Επομένως, το κρυπτοκείμενο y ανήκει στο $\mathbb{Z}_p^*$, δηλαδή:

$$C = \mathbb{Z}_p^*$$

Επιπλέον, η ιδιότητα (2) ισχύει προφανώς, καθώς κάθε κλειδί επιλέγεται με ίση πιθανότητα.

Για να δείξουμε ότι ισχύει και η ιδιότητα (1), έστω $x \in M$, $y \in C$. Θα αποδείξουμε ότι υπάρχει μοναδικό $k_1 \in K$ τέτοιο ώστε $y = \text{Enc}(k_1, x)$.

Έστω ότι υπάρχει και $k_2 \in K$, διαφορετικό από k_1, με $y = \text{Enc}(k_2, x)$. Δηλαδή:

$$\text{Enc}(k_1, x) = \text{Enc}(k_2, x)$$

Αυτό συνεπάγεται:

$$(k_1 \cdot x) \mod p = (k_2 \cdot x) \mod p$$

Άρα:

$$k_1 \cdot x \equiv k_2 \cdot x \pmod{p}$$

Αυτό συνεπάγεται:

$$p \mid x \cdot (k_1 - k_2)$$

Δεδομένου ότι το p είναι πρώτος, θα πρέπει είτε $p \mid x$ είτε $p \mid |k_1 - k_2|$. Ωστόσο, αφού $x, |k_1 - k_2| \in \mathbb{Z}_p^*$, και είναι μικρότερα του p, δεν μπορεί το p να τα διαιρεί.

Άρα, η υπόθεση ότι υπάρχει άλλο κλειδί $k_2 \neq k_1$ οδηγεί σε αντίφαση, και συνεπώς υπάρχει μοναδικό $k_1 \in K$ τέτοιο ώστε $y = \text{Enc}(k_1, x)$.

Έτσι, το κρυπτοσύστημα πληροί και τις δύο απαιτήσεις και είναι τέλεια ασφαλές.

Άσκηση 6

1. Απόδειξη ότι n είναι πρώτος

Έστω ότι το n είναι σύνθετος αριθμός. Τότε, το n μπορεί να γραφεί ως γινόμενο δύο θετικών ακεραίων k και x, δηλαδή:

$$n = k \cdot x \quad \text{για} \quad k, x \in \mathbb{N}, \quad k \neq n,\ x \neq 1.$$

Στη συνέχεια, εξετάζουμε την έκφραση $2n - 1$. Επειδή το n είναι σύνθετος, υπάρχει τουλάχιστον ένα ζεύγος k και x που να ικανοποιεί την εξίσωση. Συγκεκριμένα,

$$2n - 1 = 2kx - 1 = (2x - 1) \cdot (2x(k-1) + 2x(k-1) + \cdots + 2x + 1)\,,$$

προκύπτει ότι το $2n - 1$ είναι σύνθετος. Αυτή η παραδοχή οδηγεί στο άτοπο, διότι το n πρέπει να είναι πρώτος.

2. Απόδειξη με το Θεώρημα του Fermat

(i) Εξετάζουμε το αποτέλεσμα $(2p - 1) \mod p$. Με βάση τη θεωρία, έχουμε:

$$(2p - 1) \mod p = (2p \mod p - 1 \mod p) \mod p.$$

Επειδή p είναι πρώτος και περιττός, γνωρίζουμε από το Μικρό Θεώρημα του Fermat ότι:

$$2^{p-1} \equiv 1 \mod p.$$

Έτσι, έχουμε:

$$2p - 1 \equiv 1 \mod p.$$

(ii) Αν $M_p = 2p - 1 \equiv 1 \mod p$, τότε p διαιρεί τη διαφορά $M_p - 1$, δηλαδή:

$$p \mid (2p - 2).$$

Αυτό σημαίνει ότι $M_p = k \cdot p + 1$, όπου το k είναι ένας φυσικός αριθμός.

- Αν το M_p είναι πρώτος, τότε $\varphi(M_p) = k \cdot p$, άρα $p \mid \varphi(M_p)$.
- Αν το M_p είναι σύνθετος, έστω q πρώτος που διαιρεί το M_p. Τότε, το q θα πρέπει να ικανοποιεί την εξίσωση $2p \equiv 1 \mod q$. Αυτό μας οδηγεί στην απόδειξη ότι p διαιρεί τη συνάρτηση $\varphi(M_p)$.

3. Απόδειξη από το Θεώρημα του Fermat για δύο πρώτους αριθμούς p και q

Έστω ότι οι αριθμοί p και q είναι διαφορετικοί πρώτοι. Από το Θεώρημα του Fermat, γνωρίζουμε ότι:

$$p^{q-1} \equiv 1 \mod q \quad \text{και} \quad q^{p-1} \equiv 1 \mod p.$$

Επομένως, προκύπτει ότι $q \mid (p^{q-1} - 1)$ και $p \mid (q^{p-1} - 1)$. Από αυτή την παραδοχή, μπορούμε να καταλήξουμε στο συμπέρασμα ότι $p \mid x$, όπου x είναι κάποιο φυσικό αριθμός, και έτσι καταλήγουμε σε:

$$p \cdot q \mid (q^{q-1} - 1 + p^{p-1}).$$

Αυτό επιβεβαιώνει τη σχέση μεταξύ των πρώτων αριθμών p και q.

4. Απόδειξη για την άθροιση των στοιχείων της ομάδας Z_p^*

Η ομάδα Z_p^* αποτελείται από τα στοιχεία $\{1, 2, \ldots, p-1\}$, όπου p είναι πρώτος. Η άθροιση αυτών των στοιχείων είναι:

$$\sum_{\beta \in Z_p^*} \beta = 1 + 2 + \cdots + (p-1) = \sum_{k=1}^{p-1} k = \frac{(p-1)p}{2}.$$

Αφού p είναι πρώτος και περιττός, το $p-1$ είναι άρτιος, και έχουμε ότι:

$$\sum_{\beta \in Z_p^*} \beta \equiv 0 \mod p.$$

Για κάθε στοιχείο $\beta \in Z_p^*$, υπάρχει το αντίστροφό του $\beta^{-1} \in Z_p^*$. Επομένως, τα στοιχεία της ομάδας $\{1, 2, \ldots, p-1\}$ απεικονίζονται με μοναδικό τρόπο σε μια μετάθεση των ίδιων στοιχείων. Αυτό οδηγεί στο συμπέρασμα:

$$\sum_{\beta^{-1} \in Z_p^*} \beta^{-1} = \frac{(p-1)p}{2}.$$

Από εδώ καταλήγουμε στο ότι:

$$\sum_{\beta^{-1} \in Z_p^*} \beta^{-1} \equiv 0 \mod p.$$

Άσκηση 5

Έστω $n > 2$ ακέραιος. Θα δείξουμε ότι το n είναι πρώτος αν και μόνο αν:

$$(n-1)! \equiv -1 \pmod{n}$$

Αυτό είναι το θεώρημα του Wilson, το οποίο δηλώνει ότι για έναν πρώτο n:

$$(n-1)! \equiv -1 \pmod{n}$$

και ότι αν αυτό ισχύει, τότε το n πρέπει να είναι πρώτος.

Άσκηση 9

Ο αλγόριθμος Miller-Rabin είναι ένας πιθανοτικός αλγόριθμος που χρησιμοποιείται για τον έλεγχο της πρώτότητας ενός αριθμού. Εδώ υλοποιείται με τη χρήση της γλώσσας Python, η οποία υποστηρίζει πράξεις μεγάλων αριθμών.

Κώδικας

Ο παρακάτω κώδικας υλοποιεί τον αλγόριθμο Miller-Rabin:

```python
import random

def is_prime(n, k=5):
    # Special cases
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0:
        return False

    # Write n as 2^r * d + 1 with d odd
    r, d = 0, n - 1
    while d % 2 == 0:
        d //= 2
        r += 1

    # Perform the test k times
    for _ in range(k):
        a = random.randint(2, n - 2)
        x = pow(a, d, n)
        if x == 1 or x == n - 1:
            continue
        for _ in range(r - 1):
            x = pow(x, 2, n)
            if x == n - 1:
                break
        else:
            return False
    return True

# Test numbers
```

```python
numbers = [
    67280421310721,
    170141183460469231731687303715884105 7,
    2**1001 - 1,
    2**2281 - 1,
    2**9941 - 1
]

for number in numbers:
    print(f"{number} is prime: {is_prime(number)}")
```

Ανάλυση του Κώδικα: Μέθοδος Miller-Rabin για Έλεγχο Πρώτων Αριθμών

Ο κώδικας υλοποιεί την μέθοδο Miller-Rabin για τον έλεγχο πρώτων αριθμών. Αυτή η μέθοδος είναι πιθανολογική, που σημαίνει ότι αν η συνάρτηση επιστρέψει ότι ένας αριθμός είναι πρώτος, τότε με πολύ μεγάλη πιθανότητα είναι πρώτος, ενώ αν επιστρέψει ότι είναι σύνθετος, τότε ο αριθμός είναι σίγουρα σύνθετος.

Ειδικές Περιπτώσεις

- Αν ο αριθμός $n \leq 1$, επιστρέφεται `False`, γιατί οι αριθμοί μικρότεροι ή ίσοι του 1 δεν είναι πρώτοι.
- Αν ο αριθμός $n \leq 3$, επιστρέφεται `True` γιατί οι αριθμοί 2 και 3 είναι πρώτοι.
- Αν ο αριθμός n είναι άρτιος και μεγαλύτερος του 2 (δηλαδή $n \mod 2 = 0$), επιστρέφεται `False`, επειδή όλοι οι άρτιοι αριθμοί εκτός του 2 είναι σύνθετοι.

Αναπαράσταση του $n - 1$ ως $2^r \cdot d$

Η μέθοδος Miller-Rabin απαιτεί να γράψουμε τον αριθμό $n - 1$ ως το γινόμενο ενός δύναμης του 2 και ενός περιττού αριθμού d. Αυτό επιτυγχάνεται με τον εξής τρόπο:

```
r, d = 0, n - 1
while d % 2 == 0:
    d //= 2
    r += 1
```

Εδώ:

- Αρχικά θέτουμε $d = n - 1$.
- Επαναλαμβάνουμε τη διαδικασία $d//= 2$, δηλαδή διαιρούμε το d με το 2 όσο το d είναι άρτιος, και κάθε φορά αυξάνουμε το r κατά 1.
- Στο τέλος, το d είναι περιττός και έχουμε την αναπαράσταση $n - 1 = 2^r \cdot d$.

Δοκιμή Miller-Rabin

Η κύρια διαδικασία του αλγορίθμου είναι η δοκιμή του Miller-Rabin. Για k επαναλήψεις, επιλέγουμε τυχαίους αριθμούς a και υπολογίζουμε αν ικανοποιείται η εξής συνθήκη:

```
for _ in range(k):
    a = random.randint(2, n - 2)
    x = pow(a, d, n)
    if x == 1 or x == n - 1:
        continue
```

Αν $x = 1$ ή $x = n - 1$, η δοκιμή συνεχίζεται με την επόμενη επανάληψη. Αν όχι, προχωράμε στον έλεγχο των τετραγώνων του x:

Επαναληπτικός Έλεγχος

Αν το x δεν είναι ίσο με 1 ή $n-1$, επαναλαμβάνουμε την εξής διαδικασία για $r-1$ επαναλήψεις:

```
for _ in range(r - 1):
    x = pow(x, 2, n)
    if x == n - 1:
        break
else:
    return False
```

Αν σε οποιοδήποτε βήμα το x γίνει ίσο με $n - 1$, τότε ο αριθμός n περνά τη δοκιμή και συνεχίζουμε την επόμενη επανάληψη. Αν μετά από όλες τις επαναλήψεις το x δεν γίνει ποτέ $n - 1$, τότε ο αριθμός n είναι σύνθετος και η συνάρτηση επιστρέφει `False`.

Τελική Επιστροφή

Εάν ο αριθμός n περάσει όλες τις επαναλήψεις χωρίς να επιστρέψει `False`, τότε θεωρείται πρώτος με μεγάλη πιθανότητα και επιστρέφεται `True`:

```
return True
```

Σχόλια

Η μέθοδος Miller-Rabin είναι πιθανολογική και επομένως αν η συνάρτηση επιστρέψει `True`, τότε ο αριθμός n είναι με πολύ μεγάλη πιθανότητα πρώτος. Αν επιστρέψει `False`, τότε ο αριθμός είναι σίγουρα σύνθετος.

Ο αλγόριθμος είναι κατάλληλος για μεγάλους αριθμούς, όπως αυτοί που χρησιμοποιούνται σε συστήματα κρυπτογράφησης όπως το RSA.

Άσκηση 7

Ερώτημα 1

Έστω $a \in U(\mathbb{Z}_n)$ τάξης k και $b \in U(\mathbb{Z}_n)$ τάξης m. Θα δείξουμε ότι το στοιχείο $ab \in U(\mathbb{Z}_n)$ έχει τάξη km αν και μόνο αν $\gcd(k, m) = 1$.

Απόδειξη: Αρχικά, από την ορισμό της τάξης, γνωρίζουμε ότι για ένα στοιχείο x τάξης d, ισχύει ότι $x^d = 1$ και κανένα θετικό $d' < d$ δεν ικανοποιεί αυτήν την ιδιότητα. Για να υπολογίσουμε την τάξη του ab, θεωρούμε:

$$(ab)^{km} = a^{km}b^{km} = (a^k)^m(b^m)^k = 1$$

Αυτό δείχνει ότι η τάξη του ab διαιρεί το km.

Αν $\gcd(k, m) = 1$, τότε οι ελάχιστες θετικές δυνάμεις του a και του b που οδηγούν στην ταυτότητα είναι αντιστοίχως k και m. Επομένως, η ελάχιστη θετική δύναμη του ab που οδηγεί στην ταυτότητα είναι το γινόμενο km, και έτσι η τάξη του ab είναι km.

Γενίκευση σε αβελιανές ομάδες: Η ιδιότητα ισχύει για οποιαδήποτε πεπερασμένη αβελιανή ομάδα. Σε μια τέτοια ομάδα, η τάξη του γινομένου δύο στοιχείων ισούται με το ΕΚΠ των ταξεών τους αν και μόνο αν οι τάξεις είναι πρώτες μεταξύ τους.

Ερώτημα 2

Θα δείξουμε ότι σε μια πεπερασμένη αβελιανή ομάδα η τάξη κάθε στοιχείου διαιρεί την μέγιστη τάξη μεταξύ όλων των στοιχείων της ομάδας.

Απόδειξη: Έστω G μια πεπερασμένη αβελιανή ομάδα και d η μέγιστη τάξη στοιχείου της ομάδας G. Αν $g \in G$ είναι οποιοδήποτε στοιχείο της ομάδας με τάξη t, τότε λόγω της αβελιανής ιδιότητας της ομάδας, G είναι ισόμορφη με το άμεσο γινόμενο των κυκλικών υποομάδων της. Σύμφωνα με τη Θεωρία Sylow και την κατασκευή του άμεσου γινομένου, η τάξη κάθε στοιχείου πρέπει να διαιρεί την τάξη του γινομένου, που είναι d. Επομένως, η τάξη t του g διαιρεί τη μέγιστη τάξη d.

Υπενθύμιση από την Άσκηση 6.1

Ερώτημα 1

Έστω ότι $2^n - 1$ είναι πρώτος. Θα δείξουμε ότι n είναι επίσης πρώτος.

Αν υποθέσουμε ότι n είναι σύνθετος, τότε $n = a \cdot b$ για κάποιους ακέραιους $a, b > 1$. Τότε:

$$2^n - 1 = 2^{a \cdot b} - 1 = (2^a - 1)(1 + 2^a + 2^{2a} + \cdots + 2^{(b-1)a})$$

που σημαίνει ότι $2^n - 1$ είναι σύνθετος, άτοπο. Άρα, το n πρέπει να είναι πρώτος.

Άσκηση 8

Απόδειξη για την κυκλική ομάδα $\mathbb{Z}_p^*$ και τις υποομάδες της

1. Απόδειξη ότι το $g^{p-1} \equiv 1 \pmod{p}$

Έστω ότι g είναι γεννήτορας της κυκλικής ομάδας $\mathbb{Z}_p^*$. Από το θεώρημα του Λαγκράνζ για τις κυκλικές ομάδες, γνωρίζουμε ότι η τάξη του γεννήτορα g είναι $p-1$, οπότε ισχύει:

$$g^{p-1} \equiv 1 \pmod{p}.$$

Επίσης, έστω ότι $d \mid (p-1)$, δηλαδή το d διαιρεί το $p-1$. Άρα, μπορούμε να γράψουμε το $p-1$ ως:

$$p-1 = k \cdot d \quad \text{για κάποιον} \quad k \in \{2, \ldots, p-2\}.$$

Από την παραπάνω σχέση, έχουμε:

$$g^{k \cdot d} = 1 \pmod{p}.$$

Αυτό ισοδυναμεί με:

$$(g^k)^d = 1 \pmod{p}.$$

Πρέπει τώρα να δείξουμε ότι το d είναι η τάξη του g^k, δηλαδή ότι δεν υπάρχει μικρότερη τιμή l τέτοια ώστε:

$$(g^k)^l = 1 \pmod{p}.$$

Ας υποθέσουμε ότι υπάρχει τέτοιο $l < d$ ώστε $(g^k)^l = 1 \pmod{p}$. Τότε έχουμε:

$$g^{(p-1)\cdot \frac{l}{d}} = 1 \pmod{p}.$$

Αν $l < d$, τότε $\frac{l}{d} < 1$, άρα $(p-1) \cdot \frac{l}{d} < p-1$, γεγονός που οδηγεί στο συμπέρασμα ότι το g δεν είναι γεννήτορας, κάτι που είναι άτοπο. Συνεπώς, το d είναι η τάξη του g^k.

Άρα το ζητούμενο b είναι το $g^k \pmod{p}$, με g, k, και p γνωστά.

2. Ανάλυση της κυκλικής ομάδας $\mathbb{Z}_p^*$ και των γεννητόρων της

Η ομάδα $\mathbb{Z}_p^*$ περιέχει μία κυκλική υποομάδα τάξης d, σύμφωνα με το θεμελιώδες θεώρημα των κυκλικών υποομάδων. Ας ονομάσουμε αυτή την υποομάδα H_d. Προφανώς, ο αριθμός των στοιχείων τάξης d στην $\mathbb{Z}_p^*$ είναι ίσος με τον αριθμό των γεννητόρων της H_d. Συνεπώς, πρέπει να βρούμε πόσους γεννήτορες έχει η H_d.

Αρχικά, χρησιμοποιούμε την εξής ιδιότητα, την οποία και θα αποδείξουμε:

Ιδιότητα: Έστω G μια κυκλική ομάδα με γεννήτορα g και τάξη $|G| = n$, και έστω $k \in \mathbb{N}$. Τότε η τάξη του g^k είναι:

$$|g^k| = \frac{n}{\gcd(n,k)}.$$

Απόδειξη: Έστω ότι $g^{\gcd(n,k)}$ είναι το στοιχείο που παράγει την υποομάδα τάξης $\frac{n}{\gcd(n,k)}$, δηλαδή:

$$g^{\gcd(n,k)} = e, \quad \text{όπου} \quad e \text{ είναι το ουδέτερο στοιχείο.}$$

Είναι λοιπόν:

$$|g^k| = \frac{n}{\gcd(n,k)}.$$

Άρα, και για την υποομάδα H_d, οι γεννήτορες είναι:

$$g^k \quad \text{με} \quad |g^k| = d \quad \text{αν και μόνο αν} \quad \gcd(k,d) = 1.$$

Άρα οι γεννήτορες της υποομάδας H_d είναι $\varphi(d)$, όπου φ είναι η συνάρτηση του Ευκλείδη.

3. Το στοιχείο $b = g^d$ παράγει την υποομάδα H_d

Το στοιχείο $b = g^d$ παράγει μια κυκλική υποομάδα τάξης d, την υποομάδα H_d. Όλα τα στοιχεία της H_d έχουν τάξη d και παράγουν την υποομάδα H_d. Συνεπώς, υπάρχουν $\varphi(d)$ γεννήτορες για την υποομάδα αυτή, αφού υπάρχουν $\varphi(d)$ στοιχεία τάξης d.

4. Υποομάδα μοναδικής τάξης d

Από το θεμελιώδες θεώρημα των κυκλικών υποομάδων, υπάρχει μόνο μία κυκλική υποομάδα τάξης d στο $\mathbb{Z}_p^*$, και αυτή είναι η υποομάδα $\langle g^{p-1/d} \rangle$. Ας το αποδείξουμε:

Για $d \mid (p-1)$, η τάξη της υποομάδας $\langle g^{p-1/d} \rangle$ είναι:

$$|\langle g^{p-1/d} \rangle| = |g^{p-1/d}| = \frac{p-1}{\gcd(p-1, p-1/d)} = d.$$

Αυτή η υποομάδα είναι κυκλική υποομάδα τάξης d. Πρέπει να δείξουμε ότι είναι μοναδική. Έστω ότι υπάρχει άλλη υποομάδα H της $\mathbb{Z}_p^*$ με τάξη d. Αντίθετα, αν $H \neq \langle g^{p-1/d} \rangle$, τότε H θα έπρεπε να είναι υποομάδα της $\langle g^{p-1} \rangle$, και από την τάξη της υποομάδας καταλήγουμε στο άτοπο.

Άρα, η υποομάδα $\langle g^{p-1/d} \rangle$ είναι μοναδική.

5. Επαναληπτικός έλεγχος για το στοιχείο h στην H_d

Το στοιχείο h παράγει μια κυκλική υποομάδα τάξης d, την υποομάδα H_d. Αν το στοιχείο a ανήκει στην H_d, τότε μπορούμε να το εκφράσουμε ως:

$$a = h^{t_i} \pmod{p}.$$

Θα πρέπει να ελέγξουμε αν $a^d = 1 \pmod{p}$. Αν a δεν ανήκει στην H_d, τότε θα ισχύει $a^d \neq 1 \pmod{p}$, και αν $a^d = 1$, αυτό θα πρέπει να επιβεβαιώνεται με τον έλεγχο που έχουμε για το στοιχείο a.

Αν a έχει τάξη d, τότε $a \in H_d$, άτοπο. Αν a έχει τάξη k με $k \mid d$, τότε υπάρχει μοναδική υποομάδα τάξης k στην H_d, την οποία ονομάζουμε H_{kd}, και συνεπώς $a \in H_d$.

Άρα, αρκεί να ελέγξουμε αν $a^d = 1 \pmod{p}$, το οποίο μπορεί να γίνει σε πολυωνυμικό χρόνο.