Cryptography Exercises

Pol Pujol Santaella

October 6, 2025

Contents

1 Introduction to Cryptography

1. What is the number of possible keys in the mono-alphabetic substitution cipher?

In a mono-alphabetic substitution cipher, each letter of the plain text is replaced by a unique letter of the ciphertext alphabet. Assuming standard English alphabet of 26 letters (A-Z), each key is a *permutation* of the alphabet. The number of possible permutations of 26 letters is 26!

$$\textbf{Number of possible keys} = \mathbf{26!} \approx \mathbf{4.03} \times \mathbf{10^{26}}$$

2. Decrypt the following ciphertext using the shift cipher (key is 4).

IPPMTXMG GYVZI

First of all we have to map letters to numbers $A = 0, B = 1, ..., Z = 25$

I	P	M	T	X	G	Y	V	Z
8	15	12	19	23	6	24	21	25

To decrypt that message using a shift cipher with a key of 4, we have to know the strategy of cipher decryption:

$$P_i = (C_i - k) \bmod 26$$

Where P_i is the plaintext letter, C_i is the ciphertext letter, and k is the shift key.

I: $P_0 = (8 - 4) \bmod 26 = 4 = E$
P: $P_1 = (15 - 4) \bmod 26 = 11 = L$
...
The decrypted message is: ELLIPTIC CURVE

3. The following ciphertext has been encrypted using shift cipher.

FTUE EOTQYQ UE ZAF EQOGDQ

Figure 1.1. shows the frequency distribution of the letter in the English alphabet. Use this table to guess which is the encryption key, and decrypt the message.

a	b	c	d	e	f	g	h	i	j	k	l	m
8.05	1.62	3.2	3.65	12.31	2.28	1.61	5.14	7.18	0.1	0.52	4.03	2.25

n	o	p	q	r	s	t	u	v	w	x	y	z
7.19	7.94	2.29	0.20	6.03	6.59	9.59	3.1	0.93	2.03	0.2	1.88	0.09

Figure 1.1: Frequency distribution of English letters

To decrypt a shift cipher ciphertext using English letter frequencies to guess the key, we have to know de Shift cipher formula:

$$C = (P + k) \bmod 26, P = (C - k) \bmod 26$$

The strategy is to count letter frequency in the ciphertext, compare the most frequent ciphertext letter to the most frequent letter in English and estimate the shift key:

$$k = (C_{\text{freq}} - P_{\text{freq}}) \bmod 26$$

Then we can decrypt using this key and verify readability of the resulting plaintext; adjust key if necessary.

1. Count letters of ciphertext:

E	Q	F	T	U	O	Z	A	Y	G	D
4	4	2	2	2	2	1	1	1	1	1

2. Try mapping most frequent ciphertext letter Q with plaintext E:

$$Q = 16, E = 4$$
$$k = (16 - 4) \bmod 26 = 12$$

3. Decrypt ciphertext with the key.

$$F = 5, (5 - 12) \bmod 26 = 19 = T$$
$$T = 19, (19 - 12) \bmod 26 = 7 = H$$
$$\ldots$$

4. You will obtain the plaintext:

THIS SCHEME IS NOT SECURE

4. How would you define "perfect secrecy" in a formal way? (Discussed in Unit 3).
Perfect Secrecy is a fundamental concept in information theoretic cryptography. Let:

- M be the random variable representing plaintext message
- C be the random variable respresenting the ciphertext.
- K be the random variable representing the key.

Perfect Screcy is when we observe that ciphertext C = c gives no information about the plaintext M.
So formally a system that achieves perfect secrecy is:

$$P(M|C) = P(M)$$

For example: The One-Time Pad satisfies this condition when the key K is uniformly random, used only once, and as long as the message.

5. In the slide "two notions of security", we talk about two equivalent ways to prove the security of the scheme. Explain why the two wats are equivalent, i.e., what is the logic law we are using.

The logic law we are using here is the *contrapositive* law, which says that the 2 statements *"If A then B" and "If not B then not A"* are equivalent.

6. How would you define "computational security" in a formal way? (Discussed in Unit 3.)

We define a cryptosystem as computationally secure if no probabilistic polynomial-time (PPT) adversary can gain a non-negligible advantatge in breaking it.

7. It is thought that cryptography was present in Islamic culture since, at least, the start of the Abbasid caliphate (VII century). It is known that, at that time, Al-Khalil ibn Ahmad al-Farahidi wrote a book called "Kitab al-Muamma" (Book of Cryptographic Messages), which is considered to be lost. Later, in the IX century, Al-Kindi described a way to break subsitution ciphers, which were popular at that time. He described the frequency attack discussed in Unit 1 as follows. Finde more details in Sin99.

One way to solve an encrypted message, if we know its language, is to find a different plaintext of the same language long enough to fill one sheet or so, and then we count the occurrences of each letter. We call the most frequently occurring letter the "first", the next most occurring letter the "second", the following most occurring letter the "third", and so on, until we account for all the different letters in the plaintext sample. Then we look at the cipher text we want to solve and we also classify its symbols. We finde the most occurring symbol and change it to the form of the "first" letter of the plaintext sample, the next most common symbol is changed to the form of the "second" letter, and the following most common symbol is changed to the form of the "third" letter, and so on, until we account for all symbols of the cryptogram we want to solve.

This method does not always work, because the frequency of letters in the text cannot be exactly the same as the one of the language. Give an algorithm that, following the spirit of this attack, generates the text that are more likely to be the plaintext.

We have to design an algorithm that, given a substitution-cipher text and a language model (frequencies), produces candidate plaintexts ranked by likelihood – following Al-Kindi's idea but using statistical scoring and heuristic search.

We Assume:

- English Alphabet, 26 uppercase letters.
- We have a reference language model (letter frequency)
- Ciphertext is long. Short texts make stats noisy.

Strategy:

1. Create an initial key by rank-matching letter frequencies (Al-Kindi style)
2. Define a scoring function that measures how "English-like" a candidate plaintext is – e.g. sum of log n-gram probabilities plus a word-match bonus.
3. Use a heuristic search (hill-climbing with random restarts or simulated annealing / genetic algorithm) to explore permutations by swapping letter mapping to maximize the score.
4. Output top-K candidate plaintexts and check with dictionary/word matches for verification.

Algorithm 1

Input: C: ciphertext, r: restarts, iters: iterations
Output: Plaintext with max score.
Initialize $bestPlain \leftarrow inf$
Initialize $bestScore \leftarrow inf$
for $i = 1$ to range(r) **do**
  if $r == 0$ **then**
    key = initialKeyFreq(C)
  else
    key = randomKey()
  end if
  $currentPlain$ = Decrypt(C, key)
  $currentScore$ = scoreFunction($currentPlain$)
  $improved$ = True
  $it = 0$
  while $it < iters$ and $improved$ **do**
    $improved$ = False
    $it+ = 1$
    for ? in range(200): **do**
      a, b = randomSample(alphabet, 2)
      $key2$ = key.Copy()
      Tranpose(a, b, key2) // Transpose 2 selected letters of key2
      $plain2$ = Decrypt(C, key2)
      $score2$ = scoreFunction(plain2)
      if $score2$ ¿ $currentScore$ **then**
        $key = key2$
        $currentPlain = plain2$
        $currentScore = score2$
        $improved$ = True break
      end if
    end for
  end while
  if $currentScore$ ¿ $bestScore$ **then**
    $bestScore = currentScore$
    $bestPlain = currentPlain$
  end if
end for
return $bestPlain$, $bestScore$

2 Mathematical Backgorund

1. Computation of probabilities.

A) Coun tossing: Associate Heads to 0, and Tails to 1. Compute the probability that the output is '1' when tossing a coin. Compute the probability that the output is '101' after tossing the coin three times.

B) Dice rolling: Compute the probability that the output is '2', and the probability that the output is 'even'.