# CS698G: Assignment #1

Aaryan Maheshwari
maaryan23@iitk.ac.in

Indian Institute of Technology, Kanpur

## Introduction

This file contains my submission for the part 1 of assignment 1.

## 1 One-Time Pad (OTP)

The one-time pad is an amazingly simple, elegant encryption scheme that has theoretically unbreakable security. However, it is rarely used in practice. In this question, we will explore some of the reasons why it is not widely used.

**Question (a)**

Recall that the one-time pad encryption is defined as $c = p \oplus k$, where $p$ is the plaintext, $k$ is the key, $c$ is the ciphertext, and $k$, $p$, $c \in \{0,1\}^n$. Suppose that the key is random, and the key is only used once. Is the one-time pad encryption scheme perfectly secure?

**Solution:**

Recall, we say that a cryptosystem is perfectly secret if

$$\Pr(P = p \mid C = c) = \boxed{\Pr(P = p)}$$

for all possible plaintexts $p$ and all possible ciphertexts $c$. In other words, by observing the ciphertext, the knowledge you learned about a given plaintext is **ZERO**.

Expressing $\Pr(P = p \mid C = c)$ using Bayes's theorem, we get

$$\Pr(P = p \mid C = c) = \boxed{\frac{\Pr(C = c \mid P = p) \cdot \Pr(P = p)}{\Pr(C = c)}}$$

Recall that for a uniformly distributed key $k$:

$$\Pr(C = c \mid P = p) = \Pr(\mathsf{Enc}(k, p) = c) = \boxed{\Pr(k = c \oplus p)} = \boxed{\frac{1}{2^n}}$$

$$\Pr(C = c) = \sum_{p \in \{0,1\}^n} \boxed{\Pr(C = c \mid P = p) \cdot \Pr(P = p)} = \boxed{\frac{1}{2^n}}$$

Simplifying the Bayes's expression, we get the proof.

Question (b)

Two-Time Pad:

Suppose the encryption algorithm works as Question 1(a), but the key is reused. Now suppose you have intercepted two ciphertexts:

$$c_1 = \texttt{00011000000101110000010100000000100001010}$$

$$c_2 = \texttt{00000111000111010001101100000000100000001}$$

You know that the two ciphertexts are encrypted using the same key and **EITHER** $c_1$ is an encryption of a string "value" and $c_2$ is an encryption of a string "email" **OR** $c_1$ is an encryption of a string "hello" and $c_2$ is an encryption of a string "world". Can you recover the plaintexts? If so, what are the plaintexts? If not, explain why. (All characters are encoded in ASCII code)

**Solution:**

We are given:

$$c_1 = \texttt{00011000000101110000010100000000100001010}$$

$$c_2 = \texttt{00000111000111010001101100000000100000001}$$

The encryption is OTP: $c = p \oplus k$. Since the same key $k$ was used for both plaintexts $p_1$ and $p_2$, we have:

$$c_1 = p_1 \oplus k, \quad c_2 = p_2 \oplus k$$

Then,

$$c_1 \oplus c_2 = (p_1 \oplus k) \oplus (p_2 \oplus k) = p_1 \oplus p_2$$

So the XOR of the ciphertexts gives us the XOR of the plaintexts.

Convert $c_1$ and $c_2$ from binary to hex:

$$c_1 = \texttt{00011000 00010111 00000101 00000001 00001010} = \texttt{18 17 05 01 0A}$$
$$c_2 = \texttt{00000111 00011101 00011011 00000001 00000001} = \texttt{07 1D 1B 01 01}$$

Compute $c_1 \oplus c_2$:

$$\texttt{0x18} \oplus \texttt{0x07} = \texttt{0x1F}$$
$$\texttt{0x17} \oplus \texttt{0x1D} = \texttt{0x0A}$$
$$\texttt{0x05} \oplus \texttt{0x1B} = \texttt{0x1E}$$
$$\texttt{0x01} \oplus \texttt{0x01} = \texttt{0x00}$$
$$\texttt{0x0A} \oplus \texttt{0x01} = \texttt{0x0B}$$

So,

$$c_1 \oplus c_2 = p_1 \oplus p_2 = \texttt{1F 0A 1E 00 0B}$$

Now check both cases by computing candidate pairs:

1. "value" $\oplus$ "email":

   ASCII codes:

   $$\texttt{value} = [\texttt{0x76}, \texttt{0x61}, \texttt{0x6C}, \texttt{0x75}, \texttt{0x65}]$$

   $$\texttt{email} = [\texttt{0x65}, \texttt{0x6D}, \texttt{0x61}, \texttt{0x69}, \texttt{0x6C}]$$

   XOR:

   $$\texttt{value} \oplus \texttt{email} = [\texttt{0x13}, \texttt{0x0C}, \texttt{0x0D}, \texttt{0x1C}, \texttt{0x09}]$$

   This is **not equal** to $c_1 \oplus c_2$.

2. "hello" $\oplus$ "world":

   ASCII codes:

   $$\texttt{hello} = [\texttt{0x68}, \texttt{0x65}, \texttt{0x6C}, \texttt{0x6C}, \texttt{0x6F}]$$

   $$\texttt{world} = [\texttt{0x77}, \texttt{0x6F}, \texttt{0x72}, \texttt{0x6C}, \texttt{0x64}]$$

   XOR:

   $$\texttt{hello} \oplus \texttt{world} = [\texttt{0x1F}, \texttt{0x0A}, \texttt{0x1E}, \texttt{0x00}, \texttt{0x0B}]$$

   This **matches** $c_1 \oplus c_2$.

Thus, the pair "hello" and "world" is correct. Therefore:

$c_1$ is the encryption of "hello", and $c_2$ is the encryption of "world".

Question (c)

There is nothing exclusively special about strings and XOR in the OTP scheme, arithmetic operations can also be used instead of the XOR. Suppose $n$ is a prime, we have a plaintext $p \in \mathbb{Z}_n$ and a key $k \in \mathbb{Z}_n^+$ that are both integers. We can encrypt the plaintext by computing:

$$c = p \cdot k \bmod n$$

1. Show the corresponding decryption scheme.
2. Is this encryption scheme perfectly secure? Justify your answer.

**Solution:**

**1. Decryption Scheme:**

Since $k \in \mathbb{Z}_n^*$ and $n$ is prime, every non-zero element modulo $n$ has a unique multiplicative inverse $k^{-1} \in \mathbb{Z}_n^*$ such that:

$$k \cdot k^{-1} \equiv 1 \mod n$$

Hence, to decrypt $c$ we can compute:

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

2. **No, this scheme is not perfectly secure.**

Perfect secrecy requires that:

$$\Pr(P = p \mid C = c) = \Pr(P = p)$$

for all $p, c$. That is, observing the ciphertext should give no information about the plaintext.
However, in this multiplicative scheme:

- The number of possible ciphertexts is at most $n - 1$ (since $k \in \mathbb{Z}_n^*$).
- If a plaintext $p = 0$, then $c = 0$ regardless of the key $k$.

Hence, the ciphertext leaks information about the plaintext (e.g., if $c = 0$, we know $p = 0$ with certainty), violating perfect secrecy.

Question (d)

The daily-life communications are usually meaningful sentences. As a result, sometimes hackers do not even need a key to decrypt a ciphertext. We inherit the encryption scheme from 1(a), and you know the length of $k$, $|k| = 28$, and a ciphertext

$c =$ 0x221C05471C0E00551B09151D4F171C550B4F164F1301011C1D000E04
6E20646F20666F7220796F752C2061736B207768617420796F
752063616E20646F20666F7220796F757220636F756E7472792E204A464B

Identify the vulnerability of this scheme and try to recover the plaintext and the key. Note that we use ASCII encoding for the plaintext, the ciphertext, and the key.

**Solution:**

We are given the encryption as:

$$c_i = p_i \oplus k_{i \bmod 28}$$

This type of encryption is particularly vulnerable to attacks, since the key is small in size and is used repeatedly (although not in this case, as explained below). This allows the attacker to identify all of the ciphertext if he somehow decrypts just enough to recover the key.

Note that we are also given that the cipher is not a block cipher. This means that only the starting $28$ bytes are encrypted, the rest are not.

In the given ciphertext, this latter portion is:

```
6E 20 64 6F 20 66 6F 72 20 79 6F 75 2C 20 61 73
6B 20 77 68 61 74 20 79 6F 75 20 63 61 6E 20 64
6F 20 66 6F 72 20 79 6F 75 72 20 63 6F 75 6E 74
72 79 2E 20 4A 46 4B
```

Now, this must be plaintext. Thus, it decodes to the ASCII message:

```
n do for you, ask what you can do for your country. JFK
```

These words seem to be the ending of the famous quotation:

```
Ask not what your country can do for you, ask what you can do for your country. JFK
```

Following the hint, we assume this quotation to be the plaintext. Now we know that:

$$k_i = c_i \oplus p_i$$

Computing the XOR for each corresponding byte gives the $28$-byte key. In ASCII, this key is:

```
congratulations you found me
```

# 2 Affine Ciphers

Consider the following version of a classical cipher where plaintext and ciphertext elements are from $\mathbb{Z}_{28}$. The encryption function, which maps any plaintext $p$ to a ciphertext $c$, is given by:

$$c = E_{(a,b)}(p) = a(p + b) \bmod 28,$$

where $a, b \in \mathbb{Z}_{28}$.

Question (a)

Derive the decryption function for the scheme. Show your work.

**Solution:**

We are given the encryption function:

$$c = E_{(a,b)}(p) = a(p + b) \bmod 28,$$

where $a, b \in \mathbb{Z}_{28}$. To decrypt, we aim to recover $p$ from $c$ using a decryption function $D_{(a,b)}(c)$.

**1. Invert the encryption function**
Start with the encryption equation:

$$c \equiv a(p + b) \pmod{28}$$

**2. Multiply both sides by the multiplicative inverse of $a$ modulo 28**
Assume that $a$ is invertible modulo 28, i.e., $\gcd(a, 28) = 1$. Let $a^{-1}$ be the inverse of $a$ in $\mathbb{Z}_{28}$. Then:

$$a^{-1}c \equiv p + b \pmod{28}$$

**3. Subtract $b$ from both sides**

$$p \equiv a^{-1}c - b \pmod{28}$$

Therefore, the decryption function is:

$$\boxed{D_{(a,b)}(c) = a^{-1}c - b \mod 28}$$

**Note:** The decryption function only exists when $a$ has a multiplicative inverse modulo 28, which requires that $\gcd(a, 28) = 1$. Hence $a$ must be a unit, i.e., $a \in \{1, 3, 5, 9, 11, 13, 15, 17, 19, 23, 25, 27\}$.

Question (b)

A key is considered to be trivial if $c = p$ for all input $p$. How many non-trivial keys are possible for this scheme?

**Solution:**

We want:

$$a(p + b) \equiv p \pmod{28} \quad \text{for all } p.$$

This must hold for all $p \in \mathbb{Z}_{28}$. Expand and rearrange:

$$ap + ab \equiv p \pmod{28} \quad \Rightarrow \quad ap - p + ab \equiv 0 \pmod{28}$$

$$p(a - 1) + ab \equiv 0 \pmod{28}$$

This must be true for all $p$, so the coefficient of $p$ must be $0$ modulo 28:

$$a - 1 \equiv 0 \pmod{28} \quad \Rightarrow \quad a = 1$$

Substitute $a = 1$ into the equation:

$$1(p + b) \equiv p \pmod{28} \quad \Rightarrow \quad p + b \equiv p \pmod{28} \Rightarrow b \equiv 0 \pmod{28} \Rightarrow b = 0$$

So the only trivial key is $(a, b) = (1, 0)$.

Now, there are $12$ choices for $a$ (since $a \in Z_{28}^*$) and $28$ choices for $b$, so:

$$\text{Total number of keys} = 12 \times 28 = 336$$

$$\text{Number of trivial keys} = 1 \quad \text{(i.e., } (a, b) = (1, 0)\text{)}$$

$$\text{Number of non-trivial keys} = 336 - 1 = \boxed{335}$$

Question (c)

Assume there is a helper which can output the corresponding ciphertext for arbitrary plaintext you supply. Describe an efficient way to retrieve the key using this helper.

**Solution:**

**1. Query the helper with** $p = 0$.
From the encryption function:

$$c_0 = E_{(a,b)}(0) = a(0 + b) \bmod 28 = ab \bmod 28$$

**2. Query the helper with** $p = 1$.

$$c_1 = E_{(a,b)}(1) = a(1 + b) \bmod 28 = a(b + 1) \bmod 28$$

**3. Subtract the two results.**

$$c_1 - c_0 \equiv a(b + 1) - ab = a \pmod{28}$$

So,

$$a \equiv (c_1 - c_0) \bmod 28$$

Now that we know $a$, we can recover $b$ using the first query result:

$$c_0 = ab \pmod{28} \quad \Rightarrow \quad b \equiv a^{-1}c_0 \pmod{28}$$

provided that $a$ has an inverse modulo 28 (i.e., $\gcd(a, 28) = 1$).

# 3 Cryptanalysis on Monoalphabetic Cipher

In the lectures, we learned that a brute-force attack on a Monoalphabetic Cipher has a searching space of $N!$ where $N$ is the size of the substitution list. However, such a simple substitution cipher is vulnerable to a language statistics analysis. In this question, you are asked to find the **decryption key** (the substitution list) for a given ciphertext encrypted using Monoalphabetic Cipher.

Question (a)

You are given the following:

1. A ciphertext file `ciphertext.txt` which contains approximately 500 words encrypted from a plaintext using a **Monoalphabetic Cipher**. The original plaintext is taken from an English book. All characters in both plaintext and ciphertext are lower-cased.
2. The knowledge that the encryption key contains 26 lower-cased letters in the English alphabet (‘a’ to ‘z’) plus the **blank space** ‘ ’, the **comma** ‘,‘, and the **period** ‘.‘. Other special characters (for example: numbers, accented characters like é, hyphens, etc.) in the plaintext are kept un-substituted.

For example, `light-thinking` may be decrypted as `xgebr-rbgkjgke`.
You are required to write detailed and reasonable analysis steps for inferring each letter of the key.

**Solution:**

We are given a ciphertext encrypted using a monoalphabetic substitution cipher over a custom alphabet of 29 characters:

$$\Sigma = \{\texttt{a, b, c, ..., z, } \sqcup\texttt{, ',', '.'}\}$$

Characters outside $\Sigma$ (e.g., numbers, accented characters) remain unchanged in the ciphertext.

A visual inspection of the ciphertext shows that the character `m` is the most frequent. It consistently appears between groups of letters, which is the defining characteristic of the space character in written English. Hence we can say, `m` $\rightarrow$ space character `( )`.

Following this, the characters `l` and `v` were among the next most frequent and can be hypothesized to correspond to the English letters `e` and `t`, respectively.

The three-letter word `vsl` appears frequently, including at the very beginning of the text. The most common three-letter word in English is "the". Thus, we get `s` $\rightarrow$ `h`.

Now note the first few characters of the ciphertext. It translates to `the qlfx 1866` (since numbers are not encrypted, `1866` remains `1866`). So this word is very likely to be the word `year`, which is in line with our assumption of `l` $\rightarrow$ `e`. Thus, `q` $\rightarrow$ `y`, `f` $\rightarrow$ `a` and `x` $\rightarrow$ `r`.

See that the word `fch` is very common, and it starts with `a`. So it is very likely to be `and`, which is in line with our previous assumptions since `fch` is followed by `vsl` many time in the text, corresponding to 'and the'. So we have `c` $\rightarrow$ `n`, `h` $\rightarrow$ `d`.

Following similar logics and mapping already known letters with frequent english digrams, trigrams etc. we can do the decryption. We also use the fact the plaintext is from an english book; this leads to the complete decryption as:

```
the year 1866 was signalised by a remarkable incident, a mysterious and puzzling
phenomenon, which doubtless no one has yet forgotten.  not to mention rumours which
agitated the maritime population and excited the public mind, even in the interior
of continents, seafaring men were particularly excited.  merchants, common sailors,
captains of vessels, skippers, both of europe and america, naval officers of all
countries, and the governments of several states on the two continents, were deeply
interested in the matter.  for some time past vessels had been met by "an enormous
thing," a long object, spindle-shaped, occasionally phosphorescent, and infinitely
larger and more rapid in its movements than a whale.  the facts relating to this
apparition (entered in various log-books) agreed in most respects as to the shape
of the object or creature in question, the untiring rapidity of its movements,
its surprising power of locomotion, and the peculiar life with which it seemed
endowed.  if it was a whale, it surpassed in size all those hitherto classified
in science.  taking into consideration the mean of observations made at divers
times-rejecting the timid estimate of those who assigned to this object a length
of two hundred feet, equally with the exaggerated opinions which set it down as
a mile in width and three in length-we might fairly conclude that this mysterious
being surpassed greatly all dimensions admitted by the ichthyologists of the day,
if it existed at all.  and that it did exist was an undeniable fact; and, with
that tendency which disposes the human mind in favour of the marvellous, we can
understand the excitement produced in the entire world by this supernatural apparition.
as to classing it in the list of fables, the idea was out of the question.
```

The final decryption key is:

| Cipher | Plaintext | Cipher | Plaintext | Cipher | Plaintext |
|---|---|---|---|---|---|
| m | (space) | l | e | v | t |
| r | w | k | b | f | a |
| y | s | (space) | o | x | r |
| s | h | q | y | o | i |
| b | j | z | c | n | l |
| i | f | c | n | , | g |
| h | d | e | p | p | u |
| j | m | g | k | u | , |
| a | . | t | x | d | v |
| . | q | w | z | | |

# 4 A Practical Content Delivery Encryption System

A game company tends to protect game content delivery on PS/Switch/Xbox through DVDs. Here is one possible approach.

Suppose there are at most a total of $n$ consoles in the world (e.g., $n = 2^{32}$). We view these $n$ consoles as the leaves of a binary tree with height $\log_2 n$. Every node $v_j$ in this binary tree contains a key $k_j \in \mathcal{K}$. These keys are kept secret from consumers and are fixed for all time.

At manufacturing time, every console is assigned a serial number $i \in \{0, \ldots, n-1\}$. Let $S_i$ be the set of $1 + \log_2 n$ nodes along the path from the root of the binary tree to leaf number $i$. The manufacturer embeds in player number $i$ the $1 + \log_2 n$ keys corresponding to the nodes in $S_i$. In this way, each console ships with $1 + \log_2 n$ keys embedded in it, and these keys are supposedly inaccessible to the end user.

Ideally, a game $m$ is encrypted as:

$$\texttt{Console} := E(k_{\texttt{root}}, k)_{\texttt{header}} \parallel E(k, m)_{\texttt{body}}$$

where $E(\texttt{key}, \texttt{message})$ is an encryption scheme, $\parallel$ denotes string concatenation, and $k \leftarrow_R \mathcal{K}$ is a fresh random key called a *content key* (you can think of key $k$ as fully random and unique for each different $m$).

Since all consoles have the key $k_{\texttt{root}}$, all consoles can decrypt the content $m$. We refer to $E(k_{\texttt{root}}, k)$ as the *header* and $E(k, m)$ as the *body*.

In what follows the console header may contain multiple ciphertexts where each ciphertext is the encryption of the content $k$ under some key $k_j$ in the binary tree. That's because if some consoles are hacked, the industry can use keys $k_j$ in the binary tree to encrypt a newly released game to revoke access to this game of the hacked console. Let's see some examples.

Question (a)

Let's say that the $1 + \log_2 n$ keys embedded in console number $i$ are exposed by hackers and disclosed to the public. Show that when a new game $m$ is about to release (e.g., *Baldur's Gate 3*), $m$ can be encrypted by using a header containing $\log_2 n$ short ciphertexts so that all consoles can decrypt the game $m$ except for console number $i$. In effect, the industry disables the game for console number $i$.

**Solution:**

We're given that the $1 + \log_2 n$ keys embedded in console number $i$ are exposed by hackers and disclosed to the public. The goal is to encrypt a new game $m$ (e.g., *Baldur's Gate 3*) in such a way that all consoles *except* console number $i$ can decrypt it.

This can be achieved by constructing a header containing $\log_2 n$ ciphertexts. Each ciphertext is an encryption of the content key $k$ under a carefully selected key from the binary tree. These keys are chosen such

that every console other than $i$ has at least one of them, while console $i$ has none.

**Construction:**
Let $T$ denote the binary tree with $n$ leaves and height $\log_2 n$, where each internal node $v_j$ holds a secret key $k_j$.

Each console is associated with a path from the root to a leaf. Denote by $S_i$ the set of $1 + \log_2 n$ nodes on the path from the root to leaf $i$. The keys in $S_i$ are precisely the ones embedded in console $i$.

To prevent console $i$ from decrypting $k$, we do the following:

1. Remove leaf $i$ from the tree.
2. Let $\mathcal{C}_i$ be the set of $\log_2 n$ nodes that form a **cover** for all remaining $n-1$ leaves (i.e., every remaining console's path to the root intersects at least one node in $\mathcal{C}_i$).
3. Encrypt the content key $k$ under each $k_j$ where $v_j \in \mathcal{C}_i$.

Formally, the new header is:

$$\texttt{Header} := \{E(k_j, k) \mid v_j \in \mathcal{C}_i\}, \quad |\mathcal{C}_i| = \log_2 n$$

**Proof of correctness:**

- Any console $j \neq i$ has at least one key $k_v$ where $v \in \mathcal{C}_i$, because its path to the root intersects $\mathcal{C}_i$.
- Console $i$ has none of the keys in $\mathcal{C}_i$, because $\mathcal{C}_i$ was chosen to avoid $S_i$.

Hence, all consoles except $i$ can decrypt $k$ and thus obtain the game $m$, while console $i$ cannot.
This scheme provides an efficient and scalable method for revoking access from specific compromised consoles without reissuing keys to the rest of the user base.

Question (b)

Now suppose the keys embedded in $s$ consoles, $I = \{i_0, \ldots, i_{s-1}\}$, are exposed by hackers, where $s > 1$. At this time, the industry needs to ban all the consoles in the console set $I$ from decrypting the game. Show a way that the industry can encrypt the contents of a new game using a header containing $\mathcal{O}(s \log_2 n)$ short ciphertexts so that all the consoles can decrypt the game except for the console set $I$.

**Solution:**

We are given that the keys embedded in $s$ consoles, $I = \{i_0, \ldots, i_{s-1}\}$, are exposed by hackers, where $s > 1$. The goal is to prevent all consoles in the set $I$ from decrypting a newly released game, while allowing all other consoles to access it.

To achieve this, the industry can encrypt the content key $k$ using a header that contains $\mathcal{O}(s \log_2 n)$ short ciphertexts.

**Approach:**

- Consider the binary tree $T$ of height $\log_2 n$ whose leaves correspond to the $n$ consoles.
- For each compromised console $i_j \in I$, identify the path $P_{i_j}$ from the root to leaf $i_j$.
- Mark all nodes along each path $P_{i_j}$. Then, compute the set of nodes $C$ (called a *cover set*) such that:
  1. The subtrees rooted at nodes in $C$ collectively cover all leaves not in $I$.
  2. No node in $C$ lies on any path $P_{i_j}$ for $i_j \in I$.

- For each node $v \in C$, encrypt the content key $k$ using the key $k_v$ stored at that node:

$$\texttt{Header} := \{E(k_v, k) \mid v \in C\}$$

Since each path $P_{i_j}$ contains $\log_2 n$ nodes, and there are $s$ such paths, the total number of relevant nodes is at most $s \log_2 n$. The size of the cover set $C$ is also $\mathcal{O}(s \log_2 n)$ in the worst case.

Thus, only the consoles whose embedded keys intersect with the set $\{k_v \mid v \in C\}$ will be able to decrypt $k$ and thus decrypt the game. Since all compromised consoles were excluded from this cover set, they cannot recover $k$. All other consoles can decrypt at least one ciphertext in the header and thus access the content.