Fundamentals of Cryptography — Laboratory Work No. 3

Student: Andrii Kostiushko
Course: Fundamentals of Cryptography
Date: 10/03/2025

Overview

This laboratory work focuses on three objectives:

1. Consolidation of practical coding skills.
2. Solving of tasks using the XOR operator.
3. Obtaining practical skills in solving problems related to the concept of entropy.

The exercises cover:

- **Task 1:** Encoding messages using the Polybius square (6×6 table over A–Z and 0–9).
- **Task 2:** Evaluating XOR expressions with binary inputs.
- **Task 3:** Calculating the entropy of systems with equiprobable states using Shannon's formula.

Task 1 — Polybius Square Encryption

Algorithm

The Polybius square maps characters `A–Z` and digits `0–9` into coordinate pairs `(row, column)` with values `1..6`.

- Each symbol → 2-digit coordinate.
- Spaces are preserved (unchanged in ciphertext).

Polybius Square

	1	**2**	**3**	**4**	**5**	**6**
1	A	B	C	D	E	F
2	G	H	I	J	K	L
3	M	N	O	P	Q	R
4	S	T	U	V	W	X
5	Y	Z	0	1	2	3
6	4	5	6	7	8	9

Python Program (`polybius.py`)

```
symbols = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
GRID = [list(symbols[r*6:(r+1)*6]) for r in range(6)]

to_xy = {ch: f"{r+1}{c+1}"
         for r, row in enumerate(GRID)
         for c, ch in enumerate(row)}
from_xy = {f"{r+1}{c+1}": ch
           for r, row in enumerate(GRID)
           for c, ch in enumerate(row)}

def polybius_encrypt(text: str) -> str:
    out = []
    for ch in text.upper():
        if ch == " ":
            out.append(" ")
        elif ch in to_xy:
            out.append(to_xy[ch])
        else:
            out.append(ch)
    return "".join(out)

def polybius_decrypt(code: str) -> str:
    out, i = [], 0
    while i < len(code):
        if code[i] == " ":
            out.append(" ")
            i += 1
        elif i+1 < len(code) and code[i:i+2] in from_xy:
            out.append(from_xy[code[i:i+2]])
            i += 2
        else:
            out.append(code[i])
            i += 1
    return "".join(out)

if __name__ == "__main__":
    msg1 = "ENCRYPT ME 2 DAY"
    msg2 = "KOSTIUSHKO"
    print("Message 1:", polybius_encrypt(msg1))
    print("Message 2:", polybius_encrypt(msg2))
```

Results

- **Message a)**: `ENCRYPT ME 2 DAY` → `15321336513442 3115 55 141151`
- **Message b)**: `KOSTIUSHKO` → `25334142234341222533`

Task 2 — XOR Evaluation

We compute:

$$a \oplus b \oplus c \oplus a \oplus b$$

Simplification

XOR is associative and commutative, and satisfies $x \oplus x = 0$.
Thus:

$$a \oplus b \oplus c \oplus a \oplus b = (a \oplus a) \oplus (b \oplus b) \oplus c = 0 \oplus 0 \oplus c = c$$

Results

- Case 1: $a = 1011,\ b = 0110,\ c = 0100 \quad \Rightarrow \quad 0100$
- Case 2: $a = 0101,\ b = 1110,\ c = 1101 \quad \Rightarrow \quad 1101$

So the expression always equals c.

Task 3 — Entropy of a System (equiprobable states)

Shannon entropy:

$$H(X) = -\sum_{i=1}^{n} p(x_i)\ \log_2 p(x_i)$$

For equiprobable states:

$$p(x_i) = \frac{1}{n}$$

Substitute:

$$H(X) = -n \cdot \frac{1}{n} \cdot \log_2 \frac{1}{n} = -\log_2 \frac{1}{n} = \log_2 n$$

Results

- For $n = 8$:

$$H(X) = \log_2 8 = 3 \text{ bits}$$

- For $n = 128$:

$$H(X) = \log_2 128 = 7 \text{ bits}$$

Conclusion

- Implemented Polybius square encryption/decryption and tested on two messages.
- XOR expression simplified to c, verified on given binary inputs.
- Derived entropy values: **3 bits** for 8 states, **7 bits** for 128 states.