# Ochrona Danych - lab 1

Jakub Jaśków
Grupa laboratoryjna: nieparzyste

28 października 2025

## Spis treści

# 1 Zadanie 1 – Tabliczki dodawania i mnożenia w GF(p)

## Cel

Wyznaczyć tabliczki dodawania i mnożenia w ciele $GF(p)$ oraz elementy przeciwne i odwrotne dla trzech liczb pierwszych $p \leqslant 19$.

## Opis

Działania w ciele $GF(p)$ opisują zależności:

$$a + b = (a + b) \bmod p, \quad a \cdot b = (a \cdot b) \bmod p$$

oraz:

$$a^{-1} = a^{p-2} \bmod p, \quad \text{dla } a \neq 0.$$

## Przykładowe wyniki

- Dla $GF(3)$:
  - Elementy przeciwne: $\{0 : 0, 1 : 2, 2 : 1\}$
  - Elementy odwrotne: $\{1 : 1, 2 : 2\}$
- Dla $GF(5)$:
  - Elementy odwrotne: $\{1 : 1, 2 : 3, 3 : 2, 4 : 4\}$

## Wynik działania programu

```
(.venv) mango@T14:~/PycharmProjects/PythonProject$ python3 zad1.py

=== GF(3) ===
Dodawanie:
[0, 1, 2]
[1, 2, 0]
[2, 0, 1]
Mnożenie:
[0, 0, 0]
[0, 1, 2]
[0, 2, 1]
Elementy przeciwne: {0: 0, 1: 2, 2: 1}
Elementy odwrotne: {0: None, 1: 1, 2: 2}

=== GF(5) ===
Dodawanie:
[0, 1, 2, 3, 4]
[1, 2, 3, 4, 0]
[2, 3, 4, 0, 1]
[3, 4, 0, 1, 2]
[4, 0, 1, 2, 3]
Mnożenie:
[0, 0, 0, 0, 0]
[0, 1, 2, 3, 4]
[0, 2, 4, 1, 3]
[0, 3, 1, 4, 2]
[0, 4, 3, 2, 1]
Elementy przeciwne: {0: 0, 1: 4, 2: 3, 3: 2, 4: 1}
Elementy odwrotne: {0: None, 1: 1, 2: 3, 3: 2, 4: 4}

=== GF(7) ===
Dodawanie:
[0, 1, 2, 3, 4, 5, 6]
[1, 2, 3, 4, 5, 6, 0]
[2, 3, 4, 5, 6, 0, 1]
[3, 4, 5, 6, 0, 1, 2]
[4, 5, 6, 0, 1, 2, 3]
[5, 6, 0, 1, 2, 3, 4]
[6, 0, 1, 2, 3, 4, 5]
Mnożenie:
[0, 0, 0, 0, 0, 0, 0]
[0, 1, 2, 3, 4, 5, 6]
[0, 2, 4, 6, 1, 3, 5]
[0, 3, 6, 2, 5, 1, 4]
[0, 4, 1, 5, 2, 6, 3]
[0, 5, 3, 1, 6, 4, 2]
[0, 6, 5, 4, 3, 2, 1]
Elementy przeciwne: {0: 0, 1: 6, 2: 5, 3: 4, 4: 3, 5: 2, 6: 1}
Elementy odwrotne: {0: None, 1: 1, 2: 4, 3: 5, 4: 2, 5: 3, 6: 6}
```

# 2 Zadanie 2 – Rząd multiplikatywny elementów ciała GF(p)

## Cel

Dla każdego elementu $a \in GF(p)$ wyznaczyć jego rząd multiplikatywny:

$$\text{rząd}(a) = \min\{k > 0 : a^k \equiv 1 \pmod{p}\}$$

## Przykładowe wyniki

- Dla $GF(5)$:
$$\text{rząd}(2) = 4, \quad \text{rząd}(3) = 4, \quad \text{rząd}(4) = 2$$
- Dla $GF(7)$: $\text{rząd}(3) = 6$ (element pierwotny)

## Wynik działania programu

```
(.venv) mango@T14:~/PycharmProjects/PythonProject$ python3 zad2.py

=== GF(3) ===
rząd(1) = 1
rząd(2) = 2

=== GF(5) ===
rząd(1) = 1
rząd(2) = 4
rząd(3) = 4
rząd(4) = 2

=== GF(7) ===
rząd(1) = 1
rząd(2) = 3
rząd(3) = 6
rząd(4) = 3
rząd(5) = 6
rząd(6) = 2
```

# 3 Zadanie 3 – Elementy pierwotne w GF(p)

## Cel

Wyznaczyć wszystkie elementy pierwotne w ciele $GF(p)$, czyli takie $g$, dla których:

$$\text{rząd}(g) = p - 1$$

## Przykładowe wyniki

- $GF(3)$: [2]
- $GF(5)$: [2, 3]
- $GF(7)$: [3, 5]

### Wynik działania programu

```
(.venv) mango@T14:~/PycharmProjects/PythonProject$ python3 zad3.py
GF(3) → elementy pierwotne: [2]
GF(5) → elementy pierwotne: [2, 3]
GF(7) → elementy pierwotne: [3, 5]
```

# 4 Zadanie 4 – Sekwencje okresowe z wielomianów nad GF(p)

## Cel

Wygenerować sekwencję okresową dla wielomianu o współczynnikach $(a_0, a_1, \ldots, a_{m-1})$ nad $GF(p)$ zgodnie z zależnością rekurencyjną:

$$s_{n+m} = -(a_0 s_n + a_1 s_{n+1} + \cdots + a_{m-1} s_{n+m-1}) \bmod p$$

## Testowane wielomiany

- $x^4 + x + 1$ nad $GF(2)$
- $x^2 + x + 2$ nad $GF(3)$
- $x^2 + 2x + 1$ nad $GF(3)$

## Wyniki

| Wielomian | Ciało | Okres | Maksymalny | Pierwotny |
|---|---|---|---|---|
| $x^4 + x + 1$ | $GF(2)$ | 15 | 15 | Tak |
| $x^2 + x + 2$ | $GF(3)$ | 8 | 8 | Tak |
| $x^2 + 2x + 1$ | $GF(3)$ | 6 | 8 | Nie |

### Wynik działania programu

```
x^4 + x + 1 nad GF(2): okres = 15, maksymalny = 15, pierwotny = True
x^2 + x + 2 nad GF(3): okres = 8, maksymalny = 8, pierwotny = True
x^2 + 2x + 1 nad GF(3): okres = 6, maksymalny = 8, pierwotny = False
```

# 5 Zadanie 5 – Szyfrowanie strumieniowe z wykorzystaniem LFSR

## Cel

Zaimplementować szyfrowanie i deszyfrowanie strumieniowe z wykorzystaniem generatora pseudolosowego z wielomianem pierwotnym:

$$x^{10} + x^3 + 1 \quad \text{nad } GF(2)$$

### Opis działania

Generator LFSR tworzy ciąg bitów pseudolosowych, które są następnie łączone z bajtami danych wejściowych operacją XOR:

$$c_i = p_i \oplus k_i$$

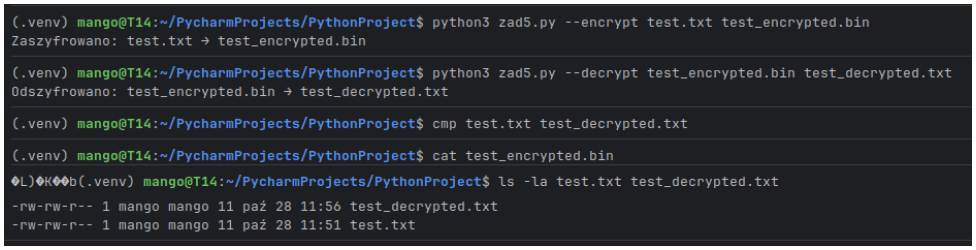
Ten sam klucz binarny umożliwia deszyfrowanie:

$$p_i = c_i \oplus k_i$$

### Przykładowy wynik

```
Tekst: Hello, world! This is LFSR stream cipher test.
Szyfrogram (hex): 5a3d2e...
Odszyfrowany: Hello, world! This is LFSR stream cipher test.
Zgodność: True
```

### Wynik działania programu




# 6 Podsumowanie

| Zadanie | Temat | Wynik |
|---|---|---|
| 1 | Tablice GF(p) | Tablice i odwrotności poprawne |
| 2 | Rzędy elementów | Wyniki zgodne z teorią |
| 3 | Elementy pierwotne | Wyznaczone poprawnie |
| 4 | Sekwencje okresowe | Potwierdzono pierwotność / niepierwotność |
| 5 | Szyfrowanie LFSR | Szyfrowanie i deszyfrowanie poprawne |

### Wnioski

- Wszystkie programy poprawnie realizują operacje w ciałach skończonych.
- Zastosowanie wielomianu pierwotnego zapewnia maksymalny okres sekwencji.
- Operacja XOR stanowi prostą i odwracalną metodę szyfrowania strumieniowego.