МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КУБГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчет**

**по практическому заданию №3**

**по курсу**

**«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»**

Работу выполнила

Студентка 49 группы

Шестак В. А.

Преподаватель:

Крамаренко А. А.

Краснодар 2025

**Цель работы:** Реализовать программный продукт построения всех примитивов заданного поля GF(2^n). Указать все промежуточные результаты, то есть:

- первичный перебор с указанием проверенных элементов поля и пояснением, почему они не примитивы;

- вывод всех примитивов с указанием степеней образующего элемента.

**Теория:**

Для реализации программного продукта, который строит все примитивные элементы заданного поля GF(2n), необходимо понимать следующие теоретические аспекты:

1. Конечные поля (поля Галуа)

Поле Галуа  GF(2n) — это конечное поле, содержащее 2n2n элементов. Каждый элемент поля может быть представлен как многочлен степени меньше nn с коэффициентами из  GF(2) (т.е. коэффициенты 0 или 1).

Пример: В поле GF(2^4) элементы представляются как многочлены степени меньше 4, например, x^3+x+1.

2. Неприводимые многочлены

Неприводимый многочлен — это многочлен, который нельзя разложить на множители меньшей степени над GF(2).

Неприводимый многочлен используется для построения поля GF(2n). Например, для GF(2^4) можно использовать p(x)=x^4+x+1.

3. Примитивные элементы

Примитивный элемент — это элемент поля GF(2^n), который является генератором мультипликативной группы поля. Это означает, что все ненулевые элементы поля могут быть получены как степени примитивного элемента.

Порядок примитивного элемента равен 2^n−1. Например, в  GF(2^4) порядок примитивного элемента равен 15.

4. Проверка на примитивность

Чтобы проверить, является ли элемент αα примитивным, нужно убедиться, что его порядок равен 2^n−1. Это означает, что:

a^2^n−1=1 (единичный элемент поля).

Α^k≠1 для всех  k<2^n−1.

Если элемент не удовлетворяет этим условиям, он не является примитивным.

5. Арифметика в поле GF(2^n)

Сложение в GF(2^n) выполняется как побитовое XOR.

Умножение выполняется как умножение многочленов с последующим приведением по модулю неприводимого многочлена.

Возведение в степень выполняется через быстрое возведение в степень с использованием умножения в поле.

**Ход работы;**

1. Умножение двух многочленов в поле GF(2^n) по модулю неприводимого многочлена.

**Шаги:**

1. Инициализировать переменную result значением 0.
2. Пока b > 0:

* Если младший бит b равен 1, выполнить result ^= a (побитовое XOR).
* Сдвинуть a влево на 1 бит.
* Если старший бит a выходит за пределы степени n, выполнить a ^n= mod\_poly (приведение по модулю).
* Сдвинуть b вправо на 1 бит.

1. Вернуть результат result.

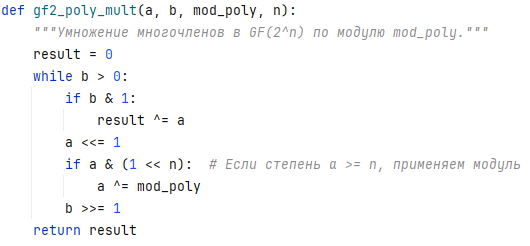


Рисунок 1 – Функция умножения двух элементов

1. Возведение многочлена в степень в поле GF(2^n) по модулю неприводимого многочлена.

**Шаги:**

1. Инициализировать переменную result значением 1.
2. Пока exp > 0:

* Если младший бит exp равен 1, выполнить result = gf2\_poly\_mult(result, base, mod\_poly, n).
* Выполнить base = gf2\_poly\_mult(base, base, mod\_poly, n) (возведение в квадрат).
* Сдвинуть exp вправо на 1 бит.

1. Вернуть результат result.

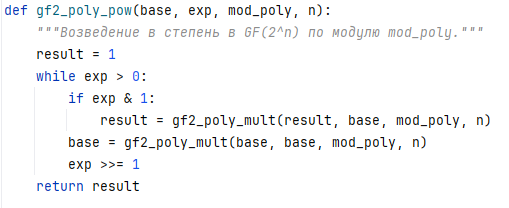


Рисунок 2 – Функция возведение многочлена в степень

1. Проверка на примитивность элемента.

**Шаги:**

1. Вычислить порядок группы: order = (1 << n) - 1.
2. Инициализировать переменную current значением element.
3. Для каждого i от 1 до order - 1:

* Если current == 1, вернуть False (элемент не примитивен).
* Выполнить current = gf2\_poly\_mult(current, element, mod\_poly, n).

1. Проверить, что current == 1 (последняя степень должна быть равна 1).
2. Вернуть True, если элемент примитивен, иначе False.

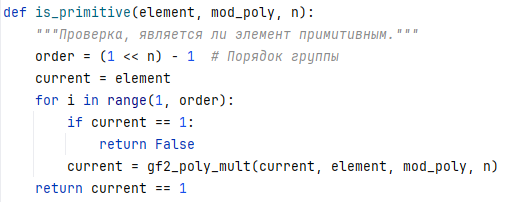


Рисунок 3 – Функция проверки элемента на примитивность

1. Нахождение всех примитивных элементов в поле GF(2^n).

**Шаги:**

1. Инициализировать пустой список primitive\_elements.
2. Для каждого i от 1 до 2n−12n−1:

* Проверить, является ли элемент i примитивным, используя функцию is\_primitive.
* Если элемент примитивен, добавить его в список primitive\_elements.
* Вывести результат проверки (примитивен или нет).

1. Вернуть список primitive\_elements.

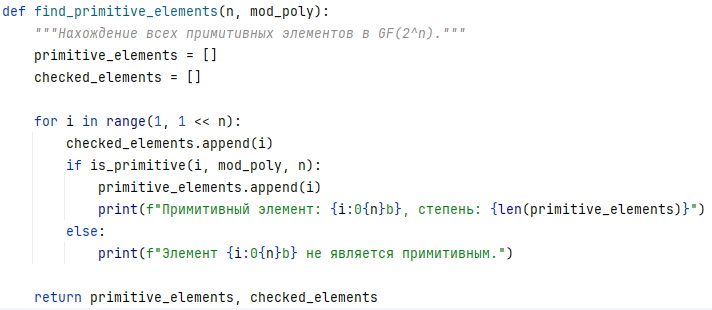


Рисунок 4 – Функция нахождения примитивных элементов

**Листинг программы:**

def gf2\_poly\_mult(a, b, mod\_poly, n):  
 *"""Умножение многочленов в GF(2^n) по модулю mod\_poly."""* result = 0  
 while b > 0:  
 if b & 1:  
 result ^= a  
 a <<= 1  
 if a & (1 << n): *# Если степень a >= n, применяем модуль* a ^= mod\_poly  
 b >>= 1  
 return result  
  
  
def gf2\_poly\_pow(base, exp, mod\_poly, n):  
 *"""Возведение в степень в GF(2^n) по модулю mod\_poly."""* result = 1  
 while exp > 0:  
 if exp & 1:  
 result = gf2\_poly\_mult(result, base, mod\_poly, n)  
 base = gf2\_poly\_mult(base, base, mod\_poly, n)  
 exp >>= 1  
 return result  
  
  
def is\_primitive(element, mod\_poly, n):  
 *"""Проверка, является ли элемент примитивным."""* order = (1 << n) - 1 *# Порядок группы* current = element  
 for i in range(1, order):  
 if current == 1:  
 return False  
 current = gf2\_poly\_mult(current, element, mod\_poly, n)  
 return current == 1  
  
  
def find\_primitive\_elements(n, mod\_poly):  
 *"""Нахождение всех примитивных элементов в GF(2^n)."""* primitive\_elements = []  
 checked\_elements = []  
  
 for i in range(1, 1 << n):  
 checked\_elements.append(i)  
 if is\_primitive(i, mod\_poly, n):  
 primitive\_elements.append(i)  
 print(f"Примитивный элемент: {i:0{n}b}, степень: {len(primitive\_elements)}")  
 else:  
 print(f"Элемент {i:0{n}b} не является примитивным.")  
  
 return primitive\_elements, checked\_elements  
  
  
def main():  
 n = 4 *# Размер поля GF(2^4)* mod\_poly = 0b10011 *# Неприводимый многочлен x^4 + x + 1 для GF(2^4)* primitive\_elements, checked\_elements = find\_primitive\_elements(n, mod\_poly)  
  
 print("\nВсе проверенные элементы поля:")  
 print([f"{x:0{n}b}" for x in checked\_elements])  
 print("\nВсе примитивные элементы:")  
 print([f"{x:0{n}b}" for x in primitive\_elements])  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()