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 выполнил

Студенты 46 группы

Прозоров М.С.

Преподаватель:

Крамаренко А.А.

Краснодар 2024

**Постановка задачи**

Реализовать проверку многочлена на неприводимость и примитивность.

Код содержит реализацию нескольких методов для работы с полиномами над конечным полем GF2 (поле из двух элементов, 0 и 1), а также с числовыми алгоритмами.

Описание каждого метода:

is\_irreducible(poly) - Метод проверяет, является ли полином poly неприводимым над полем. Полином считается неприводимым, если его нельзя разложить на множители меньшей степени, кроме 1 и самого полинома. Алгоритм:

- Полиномы степени 0 и 1 считаются неприводимыми по определению.

- Для полиномов более высоких степеней проверяется деление на все полиномы меньшей степени. Если найден полином меньшей степени, на который исходный полином делится без остатка, то исходный полином приводим.

gf2\_poly\_mul(a, b) - Метод выполняет умножение двух полиномов a и b. Возвращает результат умножения.

gf2\_poly\_mod(poly, mod\_poly) - Метод выполняет операцию взятия остатка от деления полинома poly на полином mod\_poly. Это аналогично операции взятия остатка от деления чисел.

gf2\_poly\_powmod(x, k, mod\_poly) - Метод позволяет возводить полиномы в степень с последующим взятием остатка от деления на модульный полином.

prime\_factors(n) - Метод находит и возвращает все простые делители числа n. Это числовой метод, используемый для факторизации чисел.

is\_primitive(poly) - Метод проверяет, является ли полином poly примитивным. Примитивный полином имеет особенность, что корень этого полинома генерирует мультипликативную группу поля. Для этого проверяются условия:

- Полином должен быть неприводимым.

- x^{2^n - 1} = 1 mod{poly}, где n - степень полинома.

- Условие выше не должно выполняться для любого делителя 2^n - 1 кроме самого числа.

**Вывод**

Реализовано по для проверки многочлена на неприводимость и примитивность.

**Листинг:**

**def** gf2\_poly\_mul(a, b):

result **=** [0] **\*** (len(a) **+** len(b) **-** 1)

**for** i, coeff\_a **in** enumerate(a):

**for** j, coeff\_b **in** enumerate(b):

result[i **+** j] **^=** (coeff\_a **&** coeff\_b)

**return** result

**def** gf2\_poly\_mod(poly, mod\_poly):

**def** degree(p):

**while** p **and** p[**-**1] **==** 0:

p**.**pop() # Удаляем нулевые коэффициенты с конца

**return** len(p) **-** 1

dp **=** degree(poly)

dm **=** degree(mod\_poly)

while dp >= dm:

diff **=** [0]**\***(dp **-** dm) **+** mod\_poly

**for** i **in** range(len(poly)):

poly[i] **^=** diff[i]

dp **=** degree(poly)

return poly

**def** gf2\_poly\_powmod(x, k, mod\_poly):

result **=** [1] # многочлен степени 0

while k > 0:

**if** k **&** 1:

result **=** gf2\_poly\_mod(gf2\_poly\_mul(result, x), mod\_poly)

x **=** gf2\_poly\_mod(gf2\_poly\_mul(x, x), mod\_poly)

k **>>=** 1

**return** result

**def** prime\_factors(n):

i **=** 2

factors **=** []

while i \* i <= n:

**if** n **%** i:

i **+=** 1

**else**:

n **//=** i

factors**.**append(i)

**if** n **>** 1:

factors**.**append(n)

**return** factors

**def** is\_primitive(poly):

**if** **not** is\_irreduciblle(poly):

**return** **False**

n **=** len(poly) **-** 1 # Степень многочлена

order **=** 2**\*\***n **-** 1

# Проверка, что x^(2^n - 1) ≡ 1 (mod poly)

**if** gf2\_poly\_powmod([1, 0], order, poly) **!=** [1]:

**return** **False**

# Проверка, что условие не выполняется для любого делителя 2^n - 1

**for** q **in** prime\_factors(order):

**if** gf2\_poly\_powmod([1, 0], order **//** q, poly) **==** [1]:

**return** **False**

return True

**def** is\_irreducible(poly):

# Полиномы степени 0 и 1 всегда неприводимы

**if** len(poly) **<=** 2:

**return** **True**

# Проверяем деление на все полиномы меньшей степени

**for** div\_degree **in** range(1, len(poly) **//** 2 **+** 1):

divisor **=** [1] **+** [0] **\*** (div\_degree **-** 1) **+** [1] # Простейший полином заданной степени

**while** divisor[0] **==** 1: # Пока не перебрали все полиномы данной степени

\_, remainder **=** divide\_polynomials(poly, divisor)

**if** remainder **==** [0]: # Если делится без остатка

**return** **False** # Полином приводим

# Генерируем следующий полином той же степени

**for** i **in** range(len(divisor) **-** 1, **-**1, **-**1):

divisor[i] **=** 1 **-** divisor[i] *# Переключаем коэффициент*

**if** divisor[i] **==** 1:

**break**

**return** **True** # Если не нашли делителей, полином неприводим