Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

По дисциплине «КМЗИ»

Тема: “Эллиптические кривые”

Выполнил:

Студент 3 курса

Группы ИИ-21

Романко Н.А.

Проверила:

Хацкевич А.С.

Брест 2023

Цель:Создать программу, которая реализует шифрование на основе эллиптических кривых

Задание 1: Зашифровать открытый текст

Код программы:

 def encrypt\_char(self, k, Pb, char):

        C1 = self.multiply\_point(k, self.G)

        C2 = self.multiply\_point(k, Pb)

        C2\_ = self.doubling\_additing(C2, self.alphabet[char])

        C = (C1, C2\_)

        return C

    def encrypt\_text(self, k, Pb, plaintext):

        answer = []

        i = 0

        for i, char in enumerate(plaintext):

            answer.append(self.encrypt\_char(k[i], Pb, char))

            print(f"x: {answer[i][0]} k \* G, y: {answer[i][1]} Pm + k \* Pb")

        print("\n")

Вывод программы:

Изображение выглядит как текст, снимок экрана, Шрифт, типография

Автоматически созданное описание

Задание 2: расшифровать шифртекст

Код программы:

def decrypt\_char(self, C, nb):

        C\_ = self.multiply\_point(nb, C[0])

        C\_m = self.reverse\_sign(C\_)

        C\_a = self.doubling\_additing(C[1], C\_m)

        char = self.find\_key\_by\_value(C\_a)

        if char is None:

            print("err char ind")

            return

        return char

    def decrypt\_text(self, nb, ciphertext):

        decrypted\_text = ""

        for C in ciphertext:

            char = self.decrypt\_char(C, nb)

            if char == None:

                print("error find char")

                return None

            decrypted\_text += char

        print(decrypted\_text)

        return decrypted\_text

Вывод программы: 

Задание 3: найти точку 2P + 3Q – R

Код программы:

    def do(self, obj: ElipticCurves) -> list:

        print("Задание 3")

        p2 = obj.multiply\_point(self.kP, self.P)

        q3 = obj.multiply\_point(self.kQ, self.Q)

        r\_ = obj.reverse\_sign(self.R)

        p2q3 = obj.doubling\_additing(p2, q3)

        C = obj.doubling\_additing(p2q3, r\_)

        print(f"x: {C[0]}, y: {C[1]}")

 def doubling\_additing(self, P, Q):

        if P is None:

            return Q

        if Q is None:

            return P

        x1, y1 = P

        x2, y2 = Q

        if P != Q:

            denom1 = (y2 - y1) % self.p

            denom2 = (x2 - x1) % self.p

            denom2\_ = self.gcd(denom2, self.p)

            lm = (denom1 \* denom2\_) % self.p

        elif P == Q:

            denom1 = (3 \* x1 \*\* 2 + self.a) % self.p

            denom2 = (2 \* y1) % self.p

            denom2\_ = self.gcd(denom2, self.p)

            lm = (denom1 \* denom2\_) % self.p

        x3 = (lm \*\* 2 - x1 - x2) % self.p

        y3 = (lm \* (x1 - x3) - y1) % self.p

        return (x3, y3)

    def multiply\_point(self, k, P):

        Q = None

        for i in range(k.bit\_length()):

            if (k >> i) & 1:

                Q = self.doubling\_additing(P, Q)

            P = self.doubling\_additing(P, P)

        return Q

Вывод программы: 

Задание 4: найти точку nP

Код программы:

    def \_\_init\_\_(self):

        self.P = (39, 171)

        self.k = 108

    def do(self, obj: ElipticCurves):

        print("Задание 4")

        C = obj.multiply\_point(self.k, self.P)

        print(f"x: {C[0]}, y: {C[1]}")

Вывод программы: 

Задание 5: сгенерировать ЭЦП для сообщения

Код программы:

    def signature\_generate(self, k, e, d):

        kG = self.multiply\_point(k, self.G5)

        r = kG[0] % self.n

        z = self.gcd(k, self.n)

        s = z \* (e + d \* r) % self.n

        print(f"r: {r}, s: {s}")

        return (r, s)

Вывод программы: 

Задание 6: Проверить подлинность ЭЦП

Код программы:

    def signature\_verification(self, e, rs, Q):

        if (rs[0] >= 1 and rs[0] <= self.n - 1) and (rs[1] >= 1 and rs[1] <= self.n - 1):

            v = self.gcd(rs[1], self.n)

            u1 = e \* v % self.n

            u2 = rs[0] \* 3 % self.n

            u1G = self.multiply\_point(u1, self.G6)

            u2Q = self.multiply\_point(u2, Q)

            X = self.doubling\_additing(u1G, u2Q)

            X\_ = X[0] % self.n

            if rs[0] == X\_:

                print(f"r: {rs[0]}, x: 11")

                return True

            else:

                print(f"r: {rs[0]}, x: 11")

                return False

        else:

            print("error segn ver")

            return

Вывод программы: 

Вывод: изучил шифрование на основе эллиптических кривых и создание электронно-цифровой подписи.