Белорусский государственный университет информатики и радиоэлектроники

Кафедра информатики

Лабораторная работа № 7

Криптография с использованием эллиптических кривых

Выполнила студент гр. 653502: Серебренников В. А.

Проверил ассистент КИ: Артемьев В. С.

Минск, 2019

Введение

Протокол Ди́ ффи-Хе́ллмана на эллиптических кривых (англ. Elliptic curve Diffie–Hellman, ECDH) — криптографический протокол, позволяющий двум сторонам, имеющим пары открытый/закрытый ключ на эллиптических кривых, получить общий секретный ключ, используя незащищённый от прослушивания канал связи. Этот секретный ключ может быть использован как для шифрования дальнейшего обмена, так и для формирования нового ключа, который затем может использоваться для последующего обмена информацией с помощью алгоритмов симметричного шифрования. Это вариация протокола Диффи-Хеллмана с использованием эллиптической криптографии.

Безопасность, обеспечиваемая криптографическим подходом на основе эллиптических кривых, зависит от того, насколько трудной для решения оказывается задача определения k по данным kP и P. Эту задачу обычно называют проблемой логарифмирования на эллиптической кривой. В рамках лабораторной работы необходимо реализовать программные средства шифрования и дешифрования для аналога алгоритма Деффи-Хеллмана на основе эллиптических кривых.

Блок-схема алгоритма

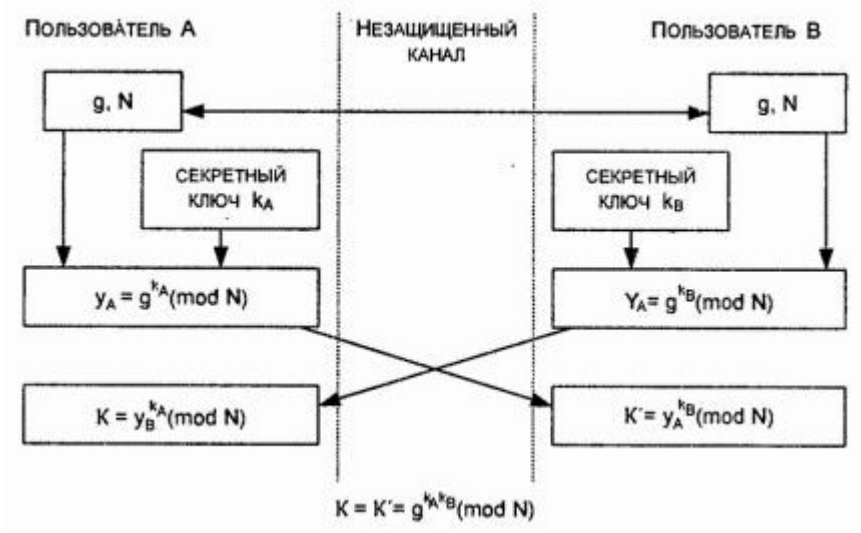

Рис.1. Схема алгоритма

Пример работы программы

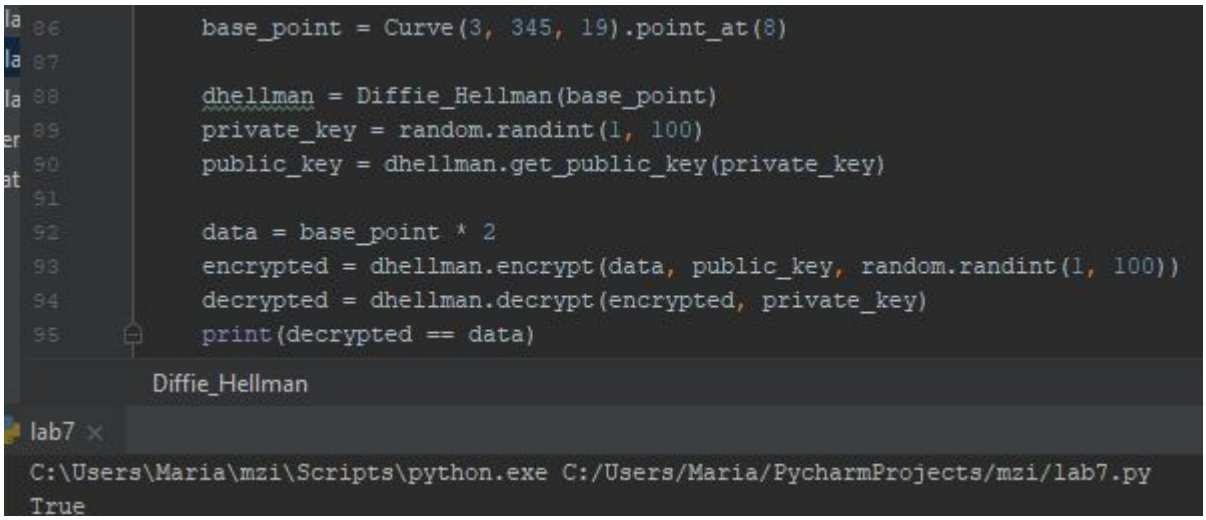

Рис.2. Пример работы

Код программы

```
import random


def inv(n, q):
    for i in range(q):
        if (n * i) % q == 1:
            return i


class Curve:
    def __init__(self, a, b, p):
        self.a = a
        self.b = b
        self.p = p

    def point_at(self, x):
        ysq = (x ** 3 + self.a * x + self.b) % self.p
        for i in range(1, self.p):
            if pow(i, 2, self.p) == ysq:
                return EllipticCurvePoint(self, x, i)


class EllipticCurvePoint:
    def __init__(self, curve, x, y):
        self.curve = curve
        self.x = x
        self.y = y

    def __neg__(self):
        return EllipticCurvePoint(self.curve, self.x, -self.y % self.curve.p)

    def __add__(self, other):
        if (self.x, self.y) == (0, 0):
            return other
        if (other.x, other.y) == (0, 0):
            return EllipticCurvePoint(self.curve, self.x, self.y)
        if self.x == other.x and (self.y != other.y or self.y == 0):
            return EllipticCurvePoint(self.curve, 0, 0)
        if self.x == other.x:
            l = (3 * self.x ** 2 + self.curve.a) * inv(2 * self.y, self.curve.p) %
self.curve.p
        else:
            l = (other.y - self.y) * inv(other.x - self.x, self.curve.p) % self.curve.p

        x = (l * l - self.x - other.x) % self.curve.p
        y = (l * (self.x - x) - self.y) % self.curve.p

        return EllipticCurvePoint(self.curve, x, y)

    def __mul__(self, number):
        result = EllipticCurvePoint(self.curve, 0, 0)
        temp = EllipticCurvePoint(self.curve, self.x, self.y)
```

```
        while 0 < number:
            if number & 1 == 1:
                result += temp
            number, temp = number >> 1, temp + temp
        return result

    def __eq__(self, other):
        return (self.curve, self.x, self.y) == (other.curve, other.x, other.y)

    def __str__(self):
        return '\n x : {}, y : {}'.format(self.x, self.y)


class Diffie_Hellman:
    def __init__(self, point):
        self.point = point
        for i in range(1, self.point.curve.p + 1):
            if self.point.x == 0 and self.point.y == 0:
                self.n = i
                break

    def get_public_key(self, private_key):
        return self.point * private_key

    def encrypt(self, data_point, public_key, random_number):
        return self.point * random_number, data_point + public_key * random_number

    def decrypt(self, data_point_pair, private_key):
        return data_point_pair[1] + -(data_point_pair[0] * private_key)


if __name__ == "__main__":

    base_point = Curve(3, 345, 19).point_at(8)

    dhellman = Diffie_Hellman(base_point)
    private_key = random.randint(1, 100)
    public_key = dhellman.get_public_key(private_key)

    data = base_point * 2
    encrypted = dhellman.encrypt(data, public_key, random.randint(1, 100))
    decrypted = dhellman.decrypt(encrypted, private_key)
    print(decrypted == data)
```

Вывод

Большинство криптосистем современной криптографии естественным образом можно «переложить» на эллиптические кривые. Основная идея заключается в том, что известный алгоритм, используемый для конкретных конечных групп, переписывается для использования групп рациональных точек эллиптических кривых.

Необходимо отметить, что безопасность таких систем цифровой подписи опирается не только на криптостойкость алгоритмов шифрования, но и на криптостойкость используемых криптографических хеш-функций и генераторов случайных чисел.

В ходе написания лабораторной работы были изучены алгоритмы шифрования и дешифрования для аналога алгоритма Диффи-Хеллмана на основе эллиптических кривых, а также написаны их программные реализации.