МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №5

“Основы шифрования данных”

по дисциплине

“Информационная безопасность”

Вариант №20

Студент:

Миху Вадим Дмитриевич

Группа P34301

Преподаватель:

Рыбаков Степан Дмитриевич

г. Санкт-Петербург

2024

Цель работы:

Сгенерировать ЭЦП для сообщения с известным значением хэш-свертки e, зная секретный ключ подписи d при данном значении выбираемого случайным образом числа k. Используется эллиптическая кривая E751(−1,1) и генерирующая точка G = (416, 55) порядка n =13.

Вариант 5:

№ Варианта	e	d	k
20	6	10	7

Листинг разработанной программы:

```
def addPoints(P: (BigInt, BigInt), Q: (BigInt, BigInt), a: BigInt, p:
BigInt): (BigInt, BigInt) = {
  val (x1, y1) = P
  val (x2, y2) = Q

  if (P == (0, 0)) return Q
  if (Q == (0, 0)) return P

  val l = if (P != Q) {
    (y2 - y1) * (x2 - x1).modInverse(p)
  } else {
    (3 * x1.pow(2) + a) * (2 * y1).modInverse(p)
  }

  val x3 = (l.pow(2) - x1 - x2) % p
  val y3 = (l * (x1 - x3) - y1) % p

  (x3, y3)
}

def modInverse(k: BigInt, n: BigInt): Option[BigInt] = {
  (1 until n.intValue).find(i => (k * i) % n == 1).map(BigInt(_))
}
```

```
@main
def main(): Unit = {

  val a = -1  // Curve parameter
  val p = BigInt(751) // Field order
  val G = (BigInt(416), BigInt(55)) // Generator point
  val n = BigInt(13)  // Order of generator point
  val e = BigInt(6) // Hash value
  val d = BigInt(10) // Secret key
  val k = BigInt(7) // Randomly chosen number k

  var R = G
  for (_ <- 1 until k.intValue) {
    R = addPoints(R, G, a, p)
  }

  val r = R._1 % n
  val z = modInverse(k, n).getOrElse {
    println("Could not find modular inverse for k")
```

```
    return
  }
  val s = (z * (e + d * r)) % n
  println(s"Generated ECDSA Signature: (r = $r, s = $s)")

  if (r < 1 || r > n - 1 || s < 1 || s > n - 1) {
    println("Invalid signature")
    return
  }
  val v = modInverse(s, n).getOrElse {
    println("Could not find modular inverse for s")
    return
  }

  var Q = G
  for (_ <- 1 until d.intValue) {
    Q = addPoints(Q, G, a, p)
  }

  val u1 = (e * v) % n
  val u2 = (r * v) % n
  var u1_G = G
  for (_ <- 1 until u1.intValue) {
    u1_G = addPoints(u1_G, G, a, p)
  }

  var u2_Q = Q
  for (_ <- 1 until u2.intValue) {
    u2_Q = addPoints(u2_Q, Q, a, p)
  }

  val X = addPoints(u1_G, u2_Q, a, p)

  if (r == X._1 % n) {
    println("Signature is valid")
  } else {
    println("Signature is invalid")
  }
}
```

Результат работы программы:

```
Generated ECDSA Signature: (r = 11, s = 11)
Signature is valid
```

Вывод:

В результате выполнения лабораторной работы я ознакомился с тем, как на эллиптических кривых выполняется сложения точек и как с их помощью сгенерировать электронную цифровую подпись и проверить ее на подлинность.