

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Казанский национальный исследовательский технологический университет»

(ФГБОУ ВО КНИТУ)

Институт нефти, химии и нанотехнологий,

факультет наноматериалов и нанотехнологий

Кафедра: Интеллектуальных систем и управления информационными ресурсами

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

На тему:

**«Диффи-Хеллман на эллиптических кривых»**

Выполнил студент группы 4301-22: Андрианов А.В

Преподаватель: Курбангалеев А.А.

Казань 2023 г.

# Диффи-Хеллман

Два пользователя А и В выбирают общие параметры:

• эллиптическую кривую над конечным полем;

• точку P на этой кривой, имеющую большой порядок n (она

не обязательно должна быть порождающим элементом

группы точек кривой, но порожденная ею подгруппа должна

быть большой, предпочтительно того же порядка, что и

сама группа). Точка P называется базовой.

Общие параметры передаются открытым каналом связи.

1 Пользователь А случайно выбирает число c – свой секретный

ключ, а пользователь В случайно выбирает для своего секретного ключа

число d (числа близки по порядку к общему числу NE точек кривой).

Далее пользователи находят свои точки Q = cP и R = dP соответственно.

2 Пользователи обмениваются точками Q и R по открытому

каналу.

3 Пользователь А, получив точку R , вычисляет точку S = cR .

4 Пользователь В, получив точку Q , вычисляет точку S = dQ .

Так как cR = c(dP) = d (cP) = dQ , то значение S – общий ключ

пользователей.

Код

internal class EllipCurves

{

public int A { get; private set; }

public int B { get; private set; }

public int p { get; private set; }

public EllipCurves(int a, int b, int p)

{

if (FuncCondition(a, b))

{

if (p <= 3)

throw new Exception("p имеет недопустимое значение");

this.A = a;

this.B = b;

this.p = p;

}

else

throw new System.Exception("Не выполняется условие 4a^3 + 27b^2 != 0");

}

public uint Right(long x)

{

return Mod((long)Math.Pow(x, 3) + A \* x + B,p);

}

public uint Left(long y)

{

return Mod((long)Math.Pow(y, 2),p);

}

private bool FuncCondition(BigInteger a,BigInteger b)

{

return 4 \* BigInteger.Pow(a, 3) + 27 \* BigInteger.Pow(b, 2) != 0;

}

private (BigInteger, BigInteger, BigInteger) GCD(BigInteger a, BigInteger b)

{

if (a == 0)

return (b, 0, 1);

(BigInteger gcd, BigInteger x, BigInteger y) = GCD(b % a, a);

return (gcd, y - (b / a) \* x, x);

}

private BigInteger Invmod(BigInteger a, BigInteger m)

{

if (a < 0)

a \*= -1;

if (m < 0)

m \*= -1;

(BigInteger g, BigInteger x, BigInteger y) = GCD(a, m);

return g > 1 ? 0 : Mod(x,m);

}

public BigInteger Mod(BigInteger a,BigInteger m)

{

if (m < 0)

m \*= -1;

return (a % m + m) % m;

}

public uint Mod(long a,long m)

{

if (m < 0)

m \*= -1;

return (uint)((a % m + m) % m);

}

public (BigInteger,BigInteger) Multiplication((BigInteger,BigInteger) P,uint n)

{

(BigInteger,BigInteger) res = Doubling(P);

n -= 2;

while (n > 0)

{

if (res.Item1 == P.Item1 && res.Item2 == P.Item2)

res = Doubling(res);

else

res = Add(res,P);

n--;

}

return res;

}

public (BigInteger,BigInteger) Multi((BigInteger,BigInteger) P,uint n)

{

List<BigInteger> binary = Perevod(n);

(BigInteger,BigInteger) tempRes = P;

(BigInteger, BigInteger) total\_res = (0,0);

for (int i = 0; i < binary.Count; i++)

{

if (i != 0)

tempRes = Doubling(tempRes);

if (binary[i] == 1)

if (i == 0)

total\_res = P;

else

total\_res = Add(total\_res, tempRes);

}

return total\_res;

}

private List<BigInteger> Perevod(BigInteger temp)

{

BigInteger temp1 = 0;

List<BigInteger> s = new List<BigInteger>();

while (temp > 0)

{

temp1 = temp % 2;

temp = temp / 2;

s.Add(temp1);

}

return s;

}

public (BigInteger, BigInteger) Add((BigInteger, BigInteger) P, (BigInteger, BigInteger) Q)

{

if (IsDotInfinity(P) ^ IsDotInfinity(Q))

{

if (IsDotInfinity(P))

return Q;

if (IsDotInfinity(Q))

return P;

}

else if (IsDotInfinity(P) & IsDotInfinity(Q))

return (0, 0);

BigInteger l = new BigInteger(0);

if (Q.Item1 - P.Item1 > 0)

l = Mod((Q.Item2 - P.Item2) \* Invmod(Q.Item1 - P.Item1, p), p);

else if (Q.Item1 - P.Item1 < 0)

l = Mod(-(Q.Item2 - P.Item2) \* Invmod(Q.Item1 - P.Item1, p), p);

else

return (0, 0);

BigInteger x3 = Mod(BigInteger.Pow(l, 2) - P.Item1 - Q.Item1,p);

BigInteger y3 = Mod(l \* (P.Item1 - x3) - P.Item2,p);

return (x3, y3);

}

public (BigInteger, BigInteger) Doubling((BigInteger, BigInteger) P)

{

BigInteger l;

if (2 \* P.Item2 > 0)

l = Mod((3 \* BigInteger.Pow(P.Item1, 2) + A) \* Invmod(2 \* P.Item2, p), p);

else if (2 \* P.Item2 < 0)

l = Mod(-(3 \* BigInteger.Pow(P.Item1, 2) + A) \* Invmod(2 \* P.Item2, p), p);

else

return (0, 0);

BigInteger x3 = Mod(BigInteger.Pow(l, 2) - 2 \* P.Item1,p);

BigInteger y3 = Mod(l \* (P.Item1 - x3) - P.Item2,p);

return (x3, y3);

}

private bool IsDotInfinity((BigInteger,BigInteger) P)

{

return P == (0,0) ? true : false;

}

}

internal class DiffiHellman

{

EllipCurves curves;

uint privateKey;

(BigInteger, BigInteger) P;

public DiffiHellman(EllipCurves curves, uint privateKey, (BigInteger, BigInteger) P)

{

this.curves = curves;

this.privateKey = privateKey;

this.P = P;

}

public (BigInteger,BigInteger) GetPartKey()

{

return curves.Multi(P, privateKey);

}

public (BigInteger,BigInteger) GetFullKey((BigInteger,BigInteger) Q)

{

return curves.Multi(Q,privateKey);

}

}

Результат

