Федеральное государственное автономное образовательное учреждение высшего профессионального образования Национальный исследовательский университет «Высшая школа экономики»

Московский институт электроники и математики Факультет прикладной математики и кибернетики

Кафедра «Компьютерная безопасность»

КУРСОВАЯ РАБОТА

по дисциплине «Программирование алгоритмов защиты информации»

Программная реализация алгоритма поиска кратной точки на эллиптической кривой в форме Пересечение Якоби (с использованием библиотеки libtommath)

Выполнил: студент группы СКБ-171 Зайцева А.А

Проверил: Доцент

Нестеренко А. Ю.

МОСКВА 2020

Программа и библиотека:

Ссылка на программу: https://github.com/Nasty09/ISAP

Для установки библиотеки нужно разархивировать файлы и, зайдя в папку, прописать sudo make install.

После этого программа должна запускаться без проблем.

Работа с библиотекой libtommath

В данной библиотеке типом для хранения больших чисел является структура **mp_int**

```
mp_err mp_init(mp_int *a) MP_WUR;
```

mp_init() инициализирует структуру mp_int.

```
void mp_clear(mp_int *a);
```

mp_clear() освобождает память структуры mp_int.

```
mp_err mp_init_copy(mp_int *a, const mp_int *b) MP_WUR;
```

mp_init_copy() инициализирует структуру mp_int через копирование

```
void mp_set_l(mp_int *a, long b);
```

mp_set_l() устанавливает значение структуры равным переданному значению long в переменной **b**.

```
mp_err mp_read_radix(mp_int *a, const char *str, int radix) MP_WUR;
```

mp_read_radix() устанавливает значение структуры равным переданному значению char в переменной **str**.

```
mp_err mp_mulmod(const mp_int *a, const mp_int *b, const mp_int *c, mp_int *d) MP_WUR;
```

mp_mulmod() умножает **a** и **b** и помещает не отрицательный результат по модулю **c** в **d** (d = a*b (mod c))

```
mp_err mp_div(const mp_int *a, const mp_int *b, mp_int *c, mp_int *d) MP_WUR;
```

mp_div() делит **a** на **b** и записывает целую часть в **c**, а остаток в **d** (c = a div b, d = a mod b)

```
mp_err mp_addmod(const mp_int *a, const mp_int *b, const mp_int *c, mp_int *d) MP_WUR;
```

mp_addmod() складывает **a** с **b** по модулю **c** и записывает результат в **d** (d = a + b (mod c))

```
mp_err mp_submod(const mp_int *a, const mp_int *b, const mp_int *c, mp_int *d) MP_WUR;
```

mp_submod() вычитает из **a** **b** по модулю **c** и записывает результат в **d** (d = a – b (mod c))

```
mp_err mp_mod(const mp_int *a, const mp_int *b, mp_int *c) MP_WUR;
```

mp_mod() записывает в **c** положительный результат привидения **a** по модулю **b** (c = a mod b, c>=0)

```
mp_err mp_sqrmod(const mp_int *a, const mp_int *b, mp_int *c) MP_WUR;
```

mp_sqrmod() возводит **a** в квадрат по модулю **b** и записывает результат в **c** (c = a^2 (mod b))

```
mp_err mp_neg(const mp_int *a, mp_int *b) MP_WUR;
```

mp_neg() делает отрицание **a** и записывает результат в **b** (b = – a)

```
mp_ord mp_cmp(const mp_int *a, const mp_int *b) MP_WUR;
```

mp_cmp() сравнивает значение **a** и **b**. Возвращает -1 если a < b, 0 если равны и 1 если a > b.

```
mp_err mp_fwrite(const mp_int *a, int radix, FILE *stream) MP_WUR;
```

mp_fwrite() выводит значение **a** в системе счисления **radix** в **stream**

Теоретическая часть

Для простого p > 3 и поля F_p эллиптической кривой в форме Пересечения Якоби называется множество точек $(s, c, d) \in F_p$, удовлетворяющих системе:

$$E_{ji}(F_p): \begin{cases} s^2 + c^2 = 1 \ (mod\ p) \\ ls^2 + d^2 = 1 \ (mod\ p) \end{cases}$$

Нейтральный элемент – это такой элемент O, который обладает свойствами:

1. $O + O = O$
2. $O + (x, y) = (x, y) + O = (x, y)$

Для нашей кривой (0,1,1,1)

Пусть P – точка на кривой E_{ji}. Если $k \in Z$, то $Q = Q + Q + \cdots + Q \ \ [k \text{ раз}] = kP$ – кратная точка.

В криптографии точка P является открытым ключом, а значение k используется как секретный ключ.

k называется дискретным логарифмом точки Q по снованию P. Эффективное вычисление kP осуществляется повторением операций удвоения и сложения точек (алгоритм «Лесенка Монтгомери»).

В аффинных координатах, требуется частое выполнение операции деления, что неэффективно. Введем проективную систему координат.

Пусть задано нечетное простое число $p > 3$ и конечное поле F_p, тогда

$$\mathbb{P}^n(F_p) = \{(x_1 \dots x_n);\ x_1, \dots, x_n \in F_p\}$$

проективное пространство с условием эквивалентности:

$$(x_1 \dots x_n) \sim (\lambda x_1 \dots \lambda x_n), \lambda \neq 0, \lambda \in F_p$$

В проективных координатах точка P с координатами (x, y) переходит в точку P с координатами $(X: Y: Z) \in \mathbb{P}^n(F_p)$, при этом:

$$x = \frac{X}{Z}, y = \frac{Y}{Z}, Z \neq 0 \ (mod\ p)$$

Пусть ЭК в форме Вейерштрасса представлена в виде:

$$E_w(F_p)\, y^2 = x^3 + ax^2 + bx$$

причем $E_w(F_p) mod\ 4 \equiv 0$. Применив стандартное преобразование Мебиуса, можем переместить три точки порядка 2 в (0, 0), (–1, 0) и (– λ, 0) , мы получим изоморфную кривую

$$E'_w(F_p) y^2 = x(x+1)(x+\lambda)$$

Представим полученную ЭК в проективном виде. Для этого нужно вычислить значения X, Y, Z и λ используя корни θ_1 , θ_2 и θ_3 из E'_w:

$$X = \frac{x - \theta_1 z}{(\theta_1 - \theta_2)^2}; \qquad Y = y(\theta_1 - \theta_2)^{5/2}; \qquad Z = \frac{z}{\theta_1 - \theta_2}; \qquad \lambda = \frac{\theta_1 - \theta_3}{\theta_1 - \theta_2}$$

где z = 1.

Таким образом, проективная форма:

$$E''_w(F_p) Y^2 Z = X(X+Z)(X+\lambda Z)$$

которая эквивалентна пересечению двух квартик.

Пусть $\lambda = 1 - l$.

Переведем в проективную форму с координатами (S, C, D, T) нашу кривую:

$$E'_{ji}(F_p): \begin{cases} S^2 + C^2 = T^2\ (mod\ p) \\ lS^2 + D^2 = T^2\ (mod\ p) \end{cases}$$

Установим связь между E''_w и E'_{ji}:

$$E''_w(F_p) \to E'_{ji}(F_p)$$

$$S = -2(X+Z)Y$$

$$C = -\lambda(X^2 + Z^2 + 2XZ) + Y^2$$

$$D = \lambda(X^2 + Z^2 + 2XZ) + Y^2$$

$$T = \lambda Z^2 + Y^2 + 2XZ + (2-\lambda)X^2$$

$$E'_{ji}(F_p) \to E''_w(F_p)$$

$$X = (D - T)\lambda$$

$$Y = S\lambda l$$

$$Z = Cl - D + T\lambda$$

Формулы сложения точек:

$$S_3 = T_1 C_2 S_1 D_2 + D_1 S_2 C_1 T_2$$

$$C_3 = T_1 C_2 C_1 T_2 - D_1 D_2 S_1 S_2$$

$$D_3 = T_1T_2D_1D_2 - lS_1S_2C_1C_2$$

$$T_3 = (T_1C_2)^2 + (S_2D_1)^2$$

Формулы удвоения точки:

$$S_3 = 2T_1C_1S_1D_1$$

$$C_3 = (T_1C_1)^2 - (T_1D_1)^2 + (D_1C_1)^2$$

$$D_3 = (T_1D_1)^2 - (T_1C_1)^2 + (D_1C_1)^2$$

$$T_3 = (T_1C_1)^2 + (T_1D_1)^2 - (D_1C_1)^2$$

Алгоритм «Лесенка Монтгомери». Пусть $Q = O,\ R = P,\ P = (X_1, Y_1, Z_1, T_1)$. Идем по каждому биту из двоичного представления числа K, при этом точка Q $\rightarrow nP = (X_n, Y_n, Z_n, T_n)$, а R $\rightarrow (n+1)P = (X_n, Y_n, Z_n, T_n)$. На каждом шаге:

$$Q_{i-1} = \begin{cases} 2Q_i, k_i = 0 \\ Q_i + R_i, k_i = 1 \end{cases}$$

$$R_{i-1} = \begin{cases} Q_i + R_i, k_i = 0 \\ 2R_i, k_i = 1 \end{cases}$$

По окончании работы алгоритма, Q -- искомая точка.

Параметры для кривой в форме Вейерштрасса:

p – характеристика простого поля, над которым определяется эллиптическая кривая;

$p = 115792089237316195423570985008687907853269984665640564039457584007913111864739$

q – порядок подгруппы простого порядка группы точек эллиптической кривой;

$\boldsymbol{q} = 48246703848881748093154577086953294938685260625318658283582607084863912737 21$

$\boldsymbol{x} = 53120966775614406564057078211824306529450596665534024993770026409585664726263$

$\boldsymbol{y} = 255072882790823071450429086279552128303052551633132455823889622698181761485 95$

$\boldsymbol{a} = 4$

$\boldsymbol{b} = 3$

После вычислений в вольфраме, для кривой в форме Вейерштрасса было получено 3 корня:

$\theta_1 = 0;$

$\theta_2 = 115792089237316195423570985008687907853269984665640564039457584007913111864736;$

$\theta_3 = 115792089237316195423570985008687907853269984665640564039457584007913111864738$

Используя эти значения, можно вычислить проективные координаты формы вейрштрасса и значение λ:

$X = 531209667756144065640570782118243065294505966665534024993770026409585664726263;$

$Y = 255072882790823071450429086279552128303052551633132455823889622698181761485 95;$

$Z = 1; \quad \lambda = 3;$

Теперь вычислим точку и коэффициент нашей кривой:

$S = 867738676704837460827097004973327815980340162269479980805710743693320584458 93$

$C = 112365783073842736181650671369471258524588261898469739668735228897200995462257$

$D = 349694970362508945766024003009033206155229743250838261281524971076561156838 06$

$T = 532699483149077005628178146826911214168751318513876473979498046689301624649 09$

$l = 115792089237316195423570985008687907853269984665640564039457584007913111864737$

Т.к. сравнивать точки в форме Монтгомери или проективной форме невозможно, то воспользуемся расширенным алгоритмом Эвклида для перевода из проективной формы в афинную.

Структуры и функции

Есть два класса:

1) Класс-обертка **mp** для структуры mp_int для удобной работа с библиотекой на C++.

```
class mp
{
    mp_int a;
```

Были перегружены операторы =, +, *, -, %, /, <<, ==, !=, <, а также операции возведения в квадрат и отрицание числа.

Так же в нем был описан расширенный алгоритм Эвклида:

```
mp AlgEucl() {...}
```

2) Основной класс **point** для реализации работы с точкой.

Был реализован конструктор, который по умолчанию устанавливает нейтральную точку, и деструктор.

```
class point
{
    mp _x, _y, _z, _t;
public:
    point(mp x = 0, mp y = 1, mp z = 1, mp t = 1) : _x(x), _y(y), _z(z), _t(t) {};
    ~point(){};
```

В этом классе были перегружены операторы ==, !=, <<, +

```
bool operator== (const point& A) { return _x == A._x && _y == A._y && _z == A._z && _t == A._t; }
bool operator!= (const point& A) { return !(*this == A); }
friend std::ostream& operator<< (std::ostream& out, point A) {...}

point operator+ (point &A) {...}
```

Также были реализованы функции:

Отрицания точки

```
point otr()
```

Проверки принадлежности точки кривой

```
bool in() {...}
```

Вычисления кратной точки

```
point crat(mp k) {...}
```

Перевод из формы Монтгомери в аффинные координаты

```
point from_pol_to () {...}
```

Коэффициент кривой **k** был задан глобальной переменной типа **mp** koef. Значение **p** было задано глобальной переменно типа **mp_int** p.

Пример работы программы

Пример 1: все точки кроме 4q и нейтральной выведены в аффинной форме

```
k1: 33496901538219218764050851946507726178212400341577406815725022275999013 70
k2: 15632791850589468802966246119709451579970922843359630063534259079745293 71
k3: 49129693388808687567017098066217177758183323184937036879259281355744307 41
tochka P:
(86773867670483746082709700497332781598034016226947998080571074369332058445893,
11236578307384273618165067136947125852458826189846973966873522889720099546225 7,
53269948314907700562817814682691121416875131851387647397949804668930162464909,
34969497036250894576602400300903320615522974325083826128152497107656115683806)


proverka na prinadleznost tocki P: 1

proverka na prinadleznost tochki kratnosti k1: 1

tochka kratnosti 4q:
(0,
63130664632989012033747806039846852700789336959147990810183191963103292286642,
63130664632989012033747806039846852700789336959147990810183191963103292286642,
63130664632989012033747806039846852700789336959147990810183191963103292286642)

neitralnaya:
(0,
1,
1,
1)


proverka na ravenstvo tochki kratnosti 4q-1 i -P: 1
tochka kratnosti 4q-1:
(53120966775614406564057078211824306529450596665534024993770026409585664726263,
90284800958233888278528076380732695022964729502327318457068621738094935716144,
1,
0)

-P:
(53120966775614406564057078211824306529450596665534024993770026409585664726263,
90284800958233888278528076380732695022964729502327318457068621738094935716144,
1,
0)


proverka na ravenstvo tochki kratnosti 4q+1 i P: 1
tochka kratnosti 4q+1:
(53120966775614406564057078211824306529450596665534024993770026409585664726263,
25507288279082307145042908627955212830305255163313245582388962269818176148595,
1,
0)
P:
(53120966775614406564057078211824306529450596665534024993770026409585664726263
25507288279082307145042908627955212830305255163313245582388962269818176148595,
1,
0)


proverka na prinadleznost summi tochek kratnosti k1 i k2: 1
summa tochek kratnosti k1 i k2:
(69581394616828985243330603614989054691573150725622354097337707368487568429 98
91794203150165143655734587608273033640514296650135200040642911895607282667488,
1,
0)

tochka kratnosti k3:
(69581394616828985243330603614989054691573150725622354097337707368487568429 98
91794203150165143655734587608273033640514296650135200040642911895607282667488,
1,
0)

proverka na ravenstvo summi tochek kratnosti k1 i k2 i tochki kratnosti k3: 1
```

Пример 2: все точки выведены в форме Монтгомери.

```
k1: 3349690153821921876405085194650772617821240034157740681572502227599901370
k2: 1563279185058946880296624611970945157997092284335963006353425907974529371
k3: 491296933888086875670170980662171777581833231849370368792592813557443074 1
tochka P:
(867738676704837460827097004973327815980340162269479980805710743693320584458 93,
112365783073842736181650671369471258524588261898469739668735228897200995462257,
532699483149077005628178146826911214168751318513876473979498046689301624649 09,
349694970362508945766024003009033206155229743250838261281524971076561156838 06)


proverka na prinadleznost tocki P: 1

proverka na prinadleznost tochki kratnosti k1: 1

tochka kratnosti 4q:
(0,
631306646329890120337478060398468527007893369591479908101831919631032922866 42,
631306646329890120337478060398468527007893369591479908101831919631032922866 42,
631306646329890120337478060398468527007893369591479908101831919631032922866 42)

neitralnaya:
(0,
1,
1,
1)


proverka na ravenstvo tochki kratnosti 4q-1 i -P: 1
tochka kratnosti 4q-1:
(100672303214762057168236739497051564150654451635945364216072322642295387191395,
398312407814039313654362345825626746990076602121135566287254910923279041310 36,
105262741231124449242745595091287751063635238097508639108085996629220742672941,
435182257289639586395263730913349566128690786268098138526127197437300640285 8)

-P:
(290182215668324493408612845113551262552359684386925659588865096385810534188 46,
112365783073842736181650671369471258524588261898469739668735228897200995462257,
532699483149077005628178146826911214168751318513876473979498046689301624649 09,
349694970362508945766024003009033206155229743250838261281524971076561156838 06)

proverka na ravenstvo tochki kratnosti 4q+1 i P: 1
tochka kratnosti 4q+1:
(245565323334313259182531440052418650441149336207477050810458250230029553556 42,
838178357676679739103592601439587240102373676792275180339854257549200845611 6,
680387096382318468420964313083123027330319249711176399700645445932492811693 09,
805227966502385584215238463584908092402912764322697916533710563271631605726 63)

P:
(867738676704837460827097004973327815980340162269479980805710743693320584458 93,
112365783073842736181650671369471258524588261898469739668735228897200995462257,
532699483149077005628178146826911214168751318513876473979498046689301624649 09,
349694970362508945766024003009033206155229743250838261281524971076561156838 06)


proverka na prinadleznost summi tochek kratnosti k1 i k2: 1
summa tochek kratnosti k1 i k2:
(707061527772350452167759358957868357900157571471791603352146030855876036732 2,
204371780408627548986833454137198351365531316924654000260771291592860944648 78,
987421065341474467294926692464146731205724840182023836043565164131231531884 7,
354850418822360044689287527323849985598020421477412614251014872031419615573 28)

tochka kratnosti k3:
(327605845407543265563329352851913997026856099576828234354245022805155144031 1,
950558714472433221545002088050510738777003346275960999286515967164242893122 91,
928094542855464593548879962914227114106437391694793493679488563587737861732 26,
641694780173731299149733126105247868735605763516144037737518570392880334365 12)

proverka na ravenstvo summi tochek kratnosti k1 i k2 i tochki kratnosti k3: 1
```

Как видно из примеров, несмотря на то, что представление точек в форме Монтгомери различается, в аффинной форме они одинаковы. Следовательно, алгоритм вычисления кратной точки на эллиптической кривой в форме Пересечение Якоби был реализован успешно.

Список литературы

1. “Preventing SPA/DPA in ECC Systems Using the Jacobi Form” – P.Y. Liardet and N.P. Smart
2. “Быстрые алгоритмы вычисления преобразований на основе эллиптических кривых” – Хлебородов Д.С.
3. Курс лекций «Методы программной реализации СКЗИ» – Нестеренко А. Ю.
4. “Sequences of numbers generated by addition in formal groups and new primality and factorization tests” – D.V.Chudnovsky and G.V.Chudnovsky