ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

Национальный исследовательский университет «Высшая школа экономики»

Московский институт электроники и математики им. А. Н. Тихонова

Кафедра «Компьютерная безопасность»

Отчёт по курсовой работе по дисциплине

“Программирование алгоритмов защиты информации”

Выполнил студент группы
СКБ-171 Шемякин Д. Н.

Тема работы: “Программная реализация алгоритма возведения в степень точки на эллиптической кривой в скрученной форме Гессе с использованием библиотеки libgmp”

МОСКВА – 2020

Оглавление

Программа и библиотека 3

Введение 3

Теоретическая часть 4

Работа с библиотекой GMP 7

Структуры и функции 8

Пример результата работы программы 10

Список материалов и ссылки: 10

Программа и библиотека

<u>Примечание:</u> в моей программе используется реализация библиотеки для c++

Ссылка на программу: https://github.com/Sh-Dmitry/PAZI

Инструкция:

- Если библиотека не установлена:
    - Возможно потребуется установить m4:

      sudo apt-get install m4

    - Скачать архив с сайта библиотеки https://gmplib.org/
    - Распаковать архив и в полученной папке выполнить команды:

      ./configure --enable-cxx

      make

      make check

      sudo make install

- Если библиотека установлена:

  g++ main.cpp -lgmpxx -lgmp

Введение

В отчете описывается программная реализация вычисления кратной точки на кривой в скрученной форме Гессе.

Параметры были сгенерированы с помощью sage. Были выбраны некоторые параметры a, d для скрученной кривой Гессе. После был выполнен переход к кривой в краткой форме Вейерштрасса, на ней с помощью sage были выбраны случайная точка и порядок группы точек.

Теоретическая часть

Эллиптическая кривая в скрученной форме Гессе имеет следующий вид:

$$a * x^3 + y^3 + 1 \equiv d * x * y (mod\ p)$$

в аффинных координатах, где a, d – параметры кривой, (x, y) – точка кривой в аффинных координатах. a, d, x, y $\in F_p$, p – простое число, p > 3.

Или:

$$a * X^3 + Y^3 + Z^3 \equiv d * X * Y * Z (mod\ p)$$

в проективных координатах, где (X : Y : Z) – точка кривой в проективных координатах. x = X/Z, y = Y/Z.

Кривая в краткой форме Вейерштрасса имеет следующий вид:

$$y^2 \equiv x^3 + n * x + b (mod\ p)$$

в аффинных координатах, где n и b параметры кривой.

$n = -\frac{d^4+216da}{48}$, $b = \frac{d^6-540d^3a-5832a^2}{864}$, где n, b – параметры кривой в сокращенной форме Вейерштрасса. a, d – параметры кривой в скрученной форме Гессе.

Пусть (u, v) – точка на кривой в сокращенной форме Вейерштрасса, тогда можно получить координаты точки (x, y) на кривой в скрученной форме Гессе следующим образом:

$$E_{\mathbf{W}} \to E_{\mathbf{H}},\ (u,v) \mapsto \left(\frac{18d^2+72u}{d^3-12du-108a+24v}, 1-\frac{48v}{d^3-12du-108a+24v}\right).$$

С помощью sage построим точку на кривой в сокращенной форме Вейерштрасса, а также получим значение порядка группы точек. Все значения будут дальше.

Нейтральный элемент – это такая точка O, что выполняются следующие свойства:

1. O + O = O
2. O + P = P + O = P

где P точка на кривой.

Для кривой в скрученной форме Гессе нейтральный элемент равен (0, -1) в аффинных координатах и (0, -1, 1) – в проективных.

Обратный элемент.

Пусть P = (x, y), тогда -P = (x/y, - 1/y) в аффинных координатах.

Пусть P = (X : Y : Z) , тогда -P = (X/Y : 1/Y : Z) в проективных координатах.

Формулы сложения двух точек

В аффинных координатах ((x1,y1) + (x2, y2) = (x3, y3)):

$$x3 = (x1-y1^2*x2*y2)/(a*x1*y1*x2^2-y2)$$
$$y3 = (y1*y2^2-a*x1^2*x2)/(a*x1*y1*x2^2-y2)$$

В проективных координатах:

Rotation addition

A = X1*Z2
B = Z1*Z2
C = Y1*X2
D = Y1*Y2
E = Z1*Y2
F = a*X1*X2
X3 = A*B-C*D
Y3 = D*E-F*A
Z3 = F*C-B*E

Standard addition

$$
\begin{aligned}
X_3 &= X_1^2 Y_2 Z_2 - X_2^2 Y_1 Z_1,\\
Y_3 &= Z_1^2 X_2 Y_2 - Z_2^2 X_1 Y_1,\\
Z_3 &= Y_1^2 X_2 Z_2 - Y_2^2 X_1 Z_1.
\end{aligned}
$$

Лесенка Монтгомери:

Данный алгоритм используется для более быстрого вычисления кратной точки.

получить двоичное представление $k = (k_{n-1}, \ldots, k_0) = \sum_{i=0}^{n-1} k_i 2^i$;
определить $Q = \mathcal{O}$, $R = P$;
for $i \leftarrow n-1$ **to** 0 **do**
    if $k_i = 0$ **then**
        вычислить $R = R + Q$ и $Q = [2]Q$;
    end
    if $k_i = 1$ **then**
        вычислить $Q = Q + R$ и $R = [2]R$;
    end
end
определить в качестве результата Q;

Значения параметров:

p =
11579208923731619542357098500868790785326998466564056403945758400
7913111864739

u =
44328971593885937857970623207174810055095945000614270339392047863
929064377300

v =
73987224069968535275377617159869580030126023743076722472100521420
353122284142

/*порядок группы точек*/m =
115792089237316195423570985008687907853279740477758714817704293727807715164245

/*параметры кривой в сокращенной форме Вейерштрасса */

n =
1157920892373161954235709850086879078532699846656405640394575840079131117524 19

b = 13602384

/*параметры кривой в скрученной форме Гессе */

a = 8

d = 48

Работа с библиотекой GMP

Для работы с большими целыми числами в библиотеке используется тип mpz_t. Для c++ можно использовать тип mpz_class, у него уже перегружены стандартные операторы и функции, что позволяет выполнять арифметические операции. В своей программе я использовал тип mpz_class.

Также у класса mpz_class можно получить тип mpz_t с помощью функции:

- mpz_t mpz_class::get_mpz_t ()

Можно использовать при вызове C функции, если ее реализации нет для класса.

- int mpz_sgn (const mpz_t op) – возвращает 1, если op > 0; возвращает -1, op < 0; 0, если op = 0;

- int mpz_invert (mpz t rop, const mpz t op1, const mpz t op2) – вычисляет обратный элемент op1 по модулю op2 и записывает результат в rop.

- size_t mpz_sizeinbase (const mpz t op, int base) – возвращает размер числа op по основанию base. Для base = 2 это количество значащих бит.

- int mpz_tstbit (const mpz t op, mp_bitcnt_t bit_index) – проверяет бит в op на позиции bit_index и возвращает 1, если бит равен 1, и 0, если равен 0.
- void gmp_randinit_mt (gmp_randstate_t state) инициализирует state алгоритмом Mersenne Twister псевдослучайных чисел.
- void mpz_urandomm (mpz_t rop, gmp_randstate_t state, const mpz_t n) – генерирует число от 0 до n - 1 по алгоритму state и помещает результат в rop

Структуры и функции

1. Структура для хранения координат точки
   ```
   struct Point{ ...};
   ```
   С функциями задания координат и их выводом.

2. Функция
   ```
   int is_equial_aff(Point p1, Point p2){ ...}
   ```
   Проверяет равенство двух точек в аффинных координатах. Возвращает 1, если равны и 0 иначе

3. Структура
   ```
   struct params_of_weierstass{ ...};
   ```
   Для хранения начальных параметров кривой в краткой форме Вейерштрасса.

4. Структура
   ```
   struct twisted_hessian_curve{
   ```
   Для хранения параметров кривой, а также точки point на этой кривой. У структуры есть несколько методов
   1. Функция
      ```
      void point_to_th(struct Point start_point)
      ```

Для перевода точки в координаты скрученной кривой Гессе. Запишет результат в точку, которая хранится в структуре.

2. Функции

```
Point rot_add(Point point_1, Point point_2){ ...}
```

```
Point std_add(Point point_1, Point point_2){ ...}
```

Сложение двух точек в проективных координатах. Возвращает результирующую точку.

3. Функция

```
Point add_aff(Point point1, Point point2)
```

Сложение двух точек в аффинных координатах. Возвращает результирующую точку.

4. Функции

```
Point crat(mpz_class k){ ...}
```

```
Point crat_aff(mpz_class k)
```

Вычисление кратной точки. Изначальная точка берется из структуры.

5. Функции

```
int check_point(Point point_1){ ...}
```

```
int check_point_aff(Point point_1)
```

Проверяют принадлежность точки к кривой.

6. Функция

```
Point invert_point(Point point1){ ...}
```

Вычисляет обратную точку.

Пример результата работы программы

```
TEST_2. small k. [k]P. [k]P is on curve.
k= 20
[k]P is on curve? true

[k]P is on curve (in affine)? true


TEST_3. small k. [k1 + k2]P = [k1]P + [k2]P.
k1=4
k2=5
k3=9
is [k1 + k2]P = [k1]P + [k2]P? (in affine) true
is [k1 + k2]P = [k1]P + [k2]P = (in affine)[k1 + k2]P = (in affine)[k1]P + [k2]P? true


TEST_4. m - is order. [m]P = q.
m=115792089237316195423570985008687907853279740477758714817704293727807715164245
result_of_kp:
is [m]P = q? true


TEST_5. [m + 1]P = P и [m - 1]P = -P.
m=115792089237316195423570985008687907853279740477758714817704293727807715164245
result_of_kp:
is [m+1]P = P? true
is [m-1]P = -P? true


TEST_6. big k. [k1 + k2]P = [k1]P + [k2]P.
k1=8348965167098195611195730644350838900
k2=7315002970883376024520284980298628552
k3=15663968137981571635716015624649467452
is [k1 + k2]P = [k1]P + [k2]P? (in affine) true
is [k1 + k2]P = [k1]P + [k2]P = (in affine)[k1 + k2]P = (in affine)[k1]P + [k2]P? true
```

Список материалов и ссылки:

- Лекции “Программирование алгоритмов защиты информации” А.Ю Нестеренко

http://www.hyperelliptic.org/EFD/g1p/auto-twistedhessian.html

- Elliptic Curves, Group Law, and efficient computation by Hüseyin Hişil