Правительство Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Кафедра «Компьютерная безопасность»

Отчёт по курсовой работе
по дисциплине
“Программирование алгоритмов защиты информации”

Программная реализация алгоритма поиска кратной точки на эллиптической кривой в форме Гессе (с использованием библиотеки libcrypto)

Выполнил студент группы СКБ-171
Лисьев Александр Николаевич

Москва, 2022

Оглавление

Задача

Реализовать вычисление кратной точки на эллиптической кривой в форме скрученной Гессе, используя при этом библиотеку libcrypto для работы с большими числами.

Ссылка на код: https://github.com/alexlis13/TwistedHesse_SSL_PAZI

Теория и описание математических алгоритмов

Эллиптическая кривая в искривленной форме Гессе имеет вид:
$aX^3 + Y^3 + Z^3 = dXYZ\ (mod\ p)$, где a, d некоторые коэффициенты,
(X: Y: Z) – точка на данной прямой, заданная в проективных координатах
a, d, X, Y, Z ∈ Fp, где p простое число, p > 3.

Кривая в краткой форме Вейерштрасса имеет следующий вид: в аффинных

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

координатах, где n и b параметры кривой.

$$n = -\frac{d^4+216da}{48}, \qquad b = \frac{d^6-540d^3a-5832a^2}{864}$$

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:

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

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 эллиптических кривых в форме скрученной Гессе нейтральный элемент равен (0 : -1 : 1).

Обратным элементом к точке (X : Y : Z) является (X/Y : 1/Y : Z)

Порядком точки P называется такое минимальное число q, что qP= O
также выполняются:
1. [q+1]P = P
2. [q-1]P = -P

Алгоритм сложения двух точек (X1 : Y1 : Z1) и (X2 : Y2 : Z2) реализуется таким образом:
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

Theorem 3.2. *Let H be the twisted Hessian curve $aX^3 + Y^3 + Z^3 = dXYZ$ over a field k. Let $X_1, Y_1, Z_1, X_2, Y_2, Z_2$ be elements of k such that $(X_1 : Y_1 : Z_1), (X_2 : Y_2 : Z_2) \in H(k)$. Define*

$$X_3 = X_1^2 Y_2 Z_2 - X_2^2 Y_1 Z_1,$$
$$Y_3 = Z_1^2 X_2 Y_2 - Z_2^2 X_1 Y_1,$$
$$Z_3 = Y_1^2 X_2 Z_2 - Y_2^2 X_1 Z_1.$$

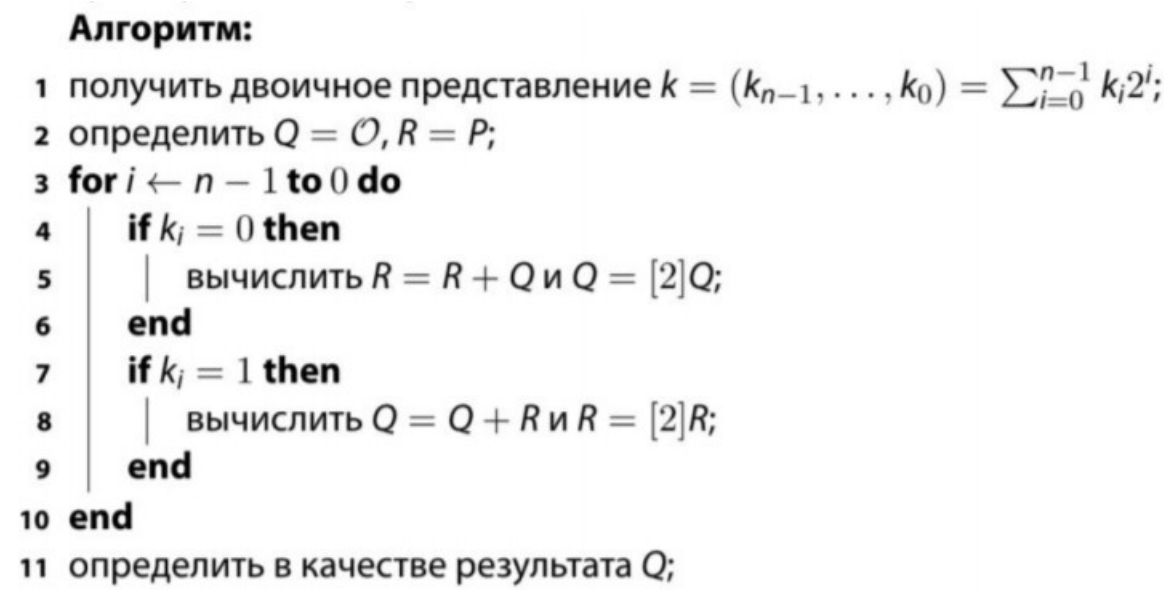
If $(X_3, Y_3, Z_3) \neq (0, 0, 0)$ then $(X_1 : Y_1 : Z_1) + (X_2 : Y_2 : Z_2) = (X_3 : Y_3 : Z_3)$.

Рис. 3: Standard Addition Law

Способ перехода из проективных координат в аффинные:
x = X/Z
y = Y/Z

Пусть P =(x,y) точка на кривой в искривленное форме Гессе, тогда:
[k]P= P + P …..+ P(k раз) – кратная точка , где k∈ Z
где 0 <= k < q

Лесенка Монтгомери:
Данный алгоритм используется для более быстрого вычисления кратной точки.



Значения параметров в Вейерштрасса(взяты из презентации прошлого года pazi-works-2020.pdf):

p =

11579208923731619542357098500868790785326998466564056403945758400

7913111864739

u =

44328971593885937857970623207174810055095945000614270339392047863

929064377300

v =

73987224069968535275377617159869580030126023743076722472100521420

353122284142

q =

11579208923731619542357098500868790785327974047775871481770429372

7807715164245

/*параметры кривой в скрученной форме Гессе */

a = 8

d = 48

Описание программной реализации

```
struct par{        //структура хранения параметров
    BIGNUM* p;
    BIGNUM* u;
    BIGNUM* v;
    BIGNUM* q;
};


struct twisted_hesse{  // структура для хранения параметров кривой
    BIGNUM* X;
    BIGNUM* Y;
    BIGNUM* Z;
    BIGNUM* a;
    BIGNUM* d;
    BIGNUM* p;
};


struct point{   // структура для хранения параметров точки
    BIGNUM* X;
    BIGNUM* Y;
    BIGNUM* Z;
};


void par_init(struct par* par); // функция для инициализации структуры
содержащей параметры в виде больших чисел


void twisted_hesse_init(struct twisted_hesse* curve, struct par* par);
// переводит точки в координаты скрученной кривой Гессе


void print_in_projective(struct point* P); //вывод точки в проективных
координатах


void print_in_affine(struct point* P); // вывод точки в аффинных
координатах


void point_init(struct point* point, char* X, char* Y, char* Z); //
Инициализация экземпляра структуры Point координатами записанными в
типе str


int is_point_equal(struct point* P1, struct point* P2, struct twisted_hesse*
curve); // проверяет равны ли точки P1
и P2 на кривой curve. Возвращает 0 , если равны, иначе возвращает -1.

void std_sum(struct point* P1, struct point* P2, struct point* P3, struct
twisted_hesse* curve); //стандартное правило сложения двух точек в
проективных координатах

void rot_sum(struct point* P1, struct point* P2, struct point* P3, struct
twisted_hesse* curve); //"перевернутое" правило сложения двух точек в
проективных координатах
```

```
void reverse_point (struct point *res, struct point *point, struct par* par);
// получает обратный элемент точки point и складывает результат в res.


int aff_point_check(struct point* Q, struct twisted_hesse* par); // проверяет
лежит ли точка P на кривой


void swap_to_affin(struct point* aff_point, struct point* P, struct
twisted_hesse* curve); // переводит точку P из проективных координат в
аффинные

void cra_find(struct point* kP, struct point* P, struct twisted_hesse* curve,
BIGNUM* degree); // Вычисление кратной точки

void FreePoint(struct point* P); //Очищает память
```

Описание использованных функций из библиотеки *libcrypto*

1. Данная библиотека была установлена на на Ubuntu с помощью команды: *sudo apt-get install openssl*

   В качестве среды разработки был использован Qt Creator.

Для работы с большими числами, в программе использовалась библиотека libcrypto. Для хранения больших чисел использовался тип BIGNUM. Для использования операций с большими числами понадобились временные структуры для хранения промежуточных вычислений, эти структуры имели тип BN_CTX.

Определения , использованных в программе функций, описаны в заголовочном файле <bn.h>:

1. BIGNUM* BN_new(void) – выделяет память в куче и инициализирует структуру BIGNUM.

2. void BN_free(BIGNUM* a) – освобождает память структуры BIGNUM.

3. BN_CTX* BN_CTX_new(void) – выделяет память в куче и инициализирует структуру BN_CTX.

4. void BN_CTX_free(BN_CTX *c) – высвобождает память структуры BN_CTX.

5. int BN_add(BIGNUM* r, const BIGNUM* a, const BIGNUM* b) – прибавляет a к b и помещает результат в r .

6. int BN_mod_add(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGNUM *m, BN_CTX *ctx) - прибавляет a к b по модулю m и помещает неотрицательный результат в r.

7. int BN_mod_sub(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGNUM *m, BN_CTX *ctx) - вычитает b из a по модулю m и помещает неотрицательный результат в r.

8. int BN_mod_mul(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGNUM *m, BN_CTX *ctx) - умножает a на b и помещает неотрицательный результат по модулю m.

9. int BN_mod_sqr(BIGNUM *r, const BIGNUM *a, const BIGNUM *m, BN_CTX*ctx) - возводит a в квадрат и помещает результат по модулю m в r.

10.int BN_exp(BIGNUM *r, const BIGNUM *a, const BIGNUM *p, BN_CTX *ctx) - возводит a в p степень и помещает результат по модулю m в r.

11.int BN_num_bits(const BIGNUM *a) – возвращает число значащих бит в a.

12. BIGNUM *BN_mod_inverse(BIGNUM *ret, const BIGNUM *a, const BIGNUM *n, BN_CTX *ctx) - вычисляет обратный элемент по модулю n и помещает результат в ret.

13.int BN_cmp(const BIGNUM *a, const BIGNUM *b) - сравнивает числа a и b, возвращает -1, если a < b, 0 – если a == b, 1 – если a > b.

14.void BN_set_negative(BIGNUM *b, int n) - устанавливает знак b (n == 0, если b должно быть положительным; n != 0 в обратном случае).

15. int BN_is_bit_set(const BIGNUM *a, int n) - проверяет, есть ли бит n в a

16.int BN_is_zero(const BIGNUM *a) - проверяет, если a == 0. Возвращает 1, если условие верно, и 0 в обратном случае.

17.char *BN_bn2dec(const BIGNUM *a) - возвращают печатаемые строки, содержащие десятичную кодировку a соответственно.

18.int BN_dec2bn(BIGNUM **a, const char *str) - берут как можно больше символов из строки str, включая начальный символ «-», который означает отрицательный элемент, чтобы сформировать десятеричное представление числа соответственно, и преобразуют их в BIGNUM, сохраняя в a.

Результаты тестирования

```
Набор параметров:
p = 1157920892373161954235709850086879078532699846656405640394575840079131118647739
u = 443289715938859378579706232071748100550959450006142703393920478639290643773300
v = 739872240699685352753776171598695800301260237430767224721005214203531222841142

Параметры Скрученного Гессе:
a = 8
d = 48

Точка P:
X_base = 3247901096424149361954530586870958650528631185663311355501815916425031325871 4
Y_base = 1971940666552221725303892477229736986718545589380570217647679061405010847393
Z_base = 1

Точка P находится на кривой
```

```
Нейтральный элемент:
В проективных координатах:
X:0
Y:1157920892373161954235709850086879078532699846656405640394575840079131118647738
Z:1

В афинных:
x:0
y:1157920892373161954235709850086879078532699846656405640394575840079131118647738

Точка O находится на кривой

Тест 1: Лежит ли точка S=(2:3:4) на кривой
Точка S не находится на кривой

Тест 2: Проверка нахождения [k]P на кривой
k = 184252
Точка kP находится на кривой
```

```
Тест 3: Проверка [q]P = O
q = 1157920892373161954235709850086879078532797404777587148177042937278077151644245
[q]P равно O

Тест 4: P+O, O+O
X:3247901096424149361954530586870958650528631185663311355501815916425031325871
4
Y:1971940666552221725303892477229736986718545589380570217647679061405010847393
Z:1

X:0
Y:1157920892373161954235709850086879078532699846656405640394575840079131118647
38
Z:1

Тест 5: Проверка равенства [q+1]P = P и [q-1] = -P

[q+1]P равно P

[q-1]P равно -P
```

```
Тест 6: Проверка равенства [k1]P + [k2]P = [k1 + k2]P
Генерация k1 и k2
k1 = 8695036707983984
k2 = 73653826353010733
k = k1 + k2 = 82348863060994717

1) [k1]P + [k2]P равно [k1 + k2]P

2) Точка [k]P находится на кривой
```

Используемые ссылки:

1. Лекции “Программирование алгоритмов защиты информации” А.Ю. Нестеренко - https://drive.google.com/drive/u/1/folders/17_F1NM91KR-6HOnUG_pHWxmtlRf7auU1
2. - Elliptic Curves, Group Law, and efficient computation by Hüseyin Hişil
3. https://www.hyperelliptic.org/EFD/g1p/auto-twistedhessian.html