ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего профессионального образования Национальный исследовательский университет «Высшая школа экономики»
Московский институт электроники и математики Национального исследовательского университета «Высшая школа экономики»
Кафедра «Компьютерная безопасность»

Курсовая работа
по дисциплине «Программирование алгоритмов защиты информации»

Выполнил:
студент группы СКБ-171
Лисьев А. Н.
Проверил:
Нестеренко А. Ю.

МОСКВА – 2021

Задание

Реализовать алгоритм вычисления кратной точки для заданной кривой (Монтгомери), используя набор функций из библиотеки GMP.

Математическое описание

Кривая Монтгомери – это эллиптическая кривая над полем $\mathrm{F_q}$, заданная в аффинных координатах уравнением:

$$\mathrm{By^2 = x^3 + Ax^2 + x, где\ B \neq 0, A^2 \neq 4}$$

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

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

Операции сложения и удвоения зависят от эллиптической кривой, над которой происходит вычисление кратной точки.

В аффинных координатах формулы сложения двух точек и удвоение требуют выполнения неэффективной операции деления. По этой причине осуществляется переход в проективные координаты:

$$\mathrm{x = \frac{X}{Z}, y = \frac{Y}{Z}, где\ Z = \overline{1, p-1}}$$

Т. е. точка $\mathrm{(x, y)}$ переходит в $\mathrm{(X, Y, Z)}$.

Кривая в форме Монтгомери в проективных координатах выглядит следующим образом:

$$\mathrm{B}Y^2Z \equiv X^3 + AX^2Z + XZ \ (mod\ p)$$

Коэффициенты кривых в формах Монтгомери и скрученного Эдвардса выражаются через друг друга по формулам:

$$A = 2\frac{e+d}{e-d}, B = \frac{4}{e-d}, x = \frac{1+v}{1-v}$$

В проективных координатах требуемые формулы имеют вид:

1. Сложение:
    a. $X_{2n+1} = \left((X_n - Z_n)(X_{n+1} + Z_{n+1}) + (X_n + Z_n)(X_{n+1} - Z_{n+1})\right)^2 Z_1$
    b. $Z_{2n+1} = \left((X_n - Z_n)(X_{n+1} + Z_{n+1}) - (X_n + Z_n)(X_{n+1} - Z_{n+1})\right)^2 X_1$
2. Удвоение:
    a. $X_{2n} = (X_n - Z_n)^2 (X_n + Z_n)^2$
    b. $Z_{2n} = ((X_n + Z_n)^2 - (X_n - Z_n)^2)((X_n + Z_n)^2 + \frac{A-2}{4}((X_n + Z_n)^2 - (X_n - Z_n)^2))$

Алгоритм сложения точек **xADD**:

Algorithm 1: xADD: differential addition on $\mathbb{P}^1$

Input: (X_P, Z_P), (X_Q, Z_Q), and $(X_\ominus, Z_\ominus)$ in $\mathbb{F}_q^2$ such that $(X_P : Z_P) = \mathbf{x}(P)$, $(X_Q : Z_Q) = \mathbf{x}(Q)$, and $(X_\ominus : Z_\ominus) = \mathbf{x}(P \ominus Q)$ for P and Q in $\mathcal{E}(\mathbb{F}_q)$

Output: $(X_\oplus, Z_\oplus)$ in $\mathbb{F}_q^2$ such that $(X_\oplus : Z_\oplus) = \mathbf{x}(P \oplus Q)$ if $P \ominus Q \notin \{O, T\}$, otherwise $X_\oplus = Z_\oplus = 0$

Cost: 4M + 2S + 3a + 3s, or 3M + 2S + 3a + 3s if $Z_\ominus$ is normalized to 1

$V_0 \leftarrow X_P + Z_P$ // 1a
$V_1 \leftarrow X_Q - Z_Q$ // 1s
$V_1 \leftarrow V_1 \cdot V_0$ // 1M
$V_0 \leftarrow X_P - Z_P$ // 1s
$V_2 \leftarrow X_Q + Z_Q$ // 1a
$V_2 \leftarrow V_2 \cdot V_0$ // 1M
$V_3 \leftarrow V_1 + V_2$ // 1a
$V_3 \leftarrow V_3^2$ // 1S
$V_4 \leftarrow V_1 - V_2$ // 1s
$V_4 \leftarrow V_4^2$ // 1S
$X_\oplus \leftarrow Z_\ominus \cdot V_3$ // 1M / 0M if $Z_\ominus = 1$
$Z_\oplus \leftarrow X_\ominus \cdot V_4$ // 1M
return $(X_\oplus : Z_\oplus)$

Алгоритм удвоения точки **xDBL**:

Algorithm 2: xDBL: pseudo-doubling on $\mathbb{P}^1$ from $\mathcal{E}_{(A,B)}$

Input: (X_P, Z_P) in $\mathbb{F}_q^2$ such that $(X_P : Z_P) = \mathbf{x}(P)$ for P in $\mathcal{E}(\mathbb{F}_q)$

Output: $(X_{[2]P}, Z_{[2]P})$ in $\mathbb{F}_q^2$ such that $(X_{[2]P} : Z_{[2]P}) = \mathbf{x}([2]P)$ if $P \notin \{O, T\}$, otherwise $Z_{[2]P} = 0$

Cost: 2M + 2S + 1c + 3a + 1s

$V_1 \leftarrow X_P + Z_P$ // 1a
$V_1 \leftarrow V_1^2$ // 1S
$V_2 \leftarrow X_P - Z_P$ // 1s
$V_2 \leftarrow V_2^2$ // 1S
$X_{[2]P} \leftarrow V_1 \cdot V_2$ // 1M
$V_1 \leftarrow V_1 - V_2$ // 1s
$V_3 \leftarrow ((A+2)/4) \cdot V_1$ // 1c
$V_3 \leftarrow V_3 + V_2$ // 1a
$Z_{[2]P} \leftarrow V_1 \cdot V_3$ // 1M
return $(X_{[2]P} : Z_{[2]P})$

Алгоритм*:*

1. Получаем на вход *k* и *P*
2. k переводится в двоичное представление $k = k_1 \dots k_i \dots k_n$
3. $R = P, Q = O \ (O = (0, 1, 0))$
4. Для каждого $k_i, i = \overline{1, n}$:
    a. Если $k_i == 0$
        i. $R = R + Q; Q = [2]Q$
    b. Если $k_i == 1$

i. $Q = Q + R; R = [2]R$

В конце алгоритма можно перевести точку обратно в аффинные координаты, но, поскольку Z в знаменателе, нужно рассмотреть два случая:

$$P = (X, Z) \to \begin{cases} (x, z), & \text{если } P = (x, 1) \\ (1,0), & \text{если } P = O = (0, 1, 0) \end{cases}$$

Тесты

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

q – порядок подгруппы простого порядка группы точек эллиптической кривой.

Перед вычислением кратной точки можно убедиться, что указанная точка P лежит на заданной кривой, с помощью символа Якоби. Для этого необходимо вычислить $n = y^2 = (x^3 + Ax^2 + x)/B$ и посчитать значение символа Якоби для n и p. Если он равен 1, точка принадлежит эллиптической кривой.

Для тестирования алгоритма можно использовать следующие свойства:

1. $[q \pm 1]P = P$;
2. $[q]P = e -$ единичный элемент.

Библиотека GMP

Библиотека позволяет проводить вычисления над большими числами. Целые числа в GMP представлены типом mpz_t.

Для инициализации и определения переменных типа mpz_t будут использоваться следующие функции:

1. $void\ mpz\_inits(mpz\_t\ a)$ – инициализация одной переменной a нулем;
2. $void\ mpz\_inits(mpz\_t\ a, mpz\_t\ b, \dots, 0)$ – инициализация нескольких переменных $a, b, \dots$ нулем;
3. $void\ mpz\_set(mpz\_t\ a, mpz\_t\ b)$ – установить значение a равным b
4. $void\ mpz\_set\_str(mpz\_t\ a, const\ char\ *\ c, int\ i)$ – установить значение a равным c в системе счисления с основанием i;
5. $void\ mpz\_init\_set\_str(mpz\_t\ a, const\ char\ *\ c, int\ i)$ – проинициализировать переменную a, а затем установить ее значение равным c в системе счисления с основанием i.

Для освобождения ресурсов будут использоваться следующие функции:

1. $void\ clear(mpz\_t\ a)$ – очистить память, в которой хранится a
2. $void\ clear(mpz\_t\ a, mpz\_t\ b, \dots, 0)$ – очистить память, в которой хранится $a, b, \dots$

Вычисление необходимых математических операций будет осуществляться с помощью функций:

1. $void\ mpz\_add(mpz\_t\ r, mpz\_t\ a, mpz\_t\ b)$ – сложить a и b, а результат записать в r;
2. $void\ mpz\_sub(mpz\_t\ r, mpz\_t\ a, mpz\_t\ b)$ – вычесть b из a, а результат записать в r;
3. $void\ mpz\_mul(mpz\_t\ r, mpz\_t\ a, mpz\_t\ b)$ – умножить a на b, а результат записать в r;
4. $void\ mpz\_mod(mpz\_t\ r, mpz\_t\ a, mpz\_t\ p)$ – привести a по модулю p, а результат записать в r;
5. $void\ mpz\_invert(mpz\_t\ r, mpz\_t\ a, mpz\_t\ p)$ – вычислить a^{-1} по модулю p, а результат записать в r;
6. $void\ mpz\_invert(mpz\_t\ r, mpz\_t\ a, mpz\_t\ pow, mpz\_t\ p)$ – вычислить a^{pow} по модулю p, а результат записать в r;
7. $int\ mpz\_jacobi(mpz\_t\ a, mpz\_t\ p)$ – вычислить (a/p)

В реализации алгоритма также используются некоторые вспомогательные функции:

1. $int\ gmp\_printf\ (const\ char\ *, \dots)$ – эквивалентно функции $printf$
2. $void\ gmp\_randinit\_default\ (gmp\_randstate\_t)$ – используется для инициализации рандомного состояния $gmp\_randstate\_t\ state$
3. $void\ mpz\_urandomm\ (mpz\_t\ r, gmp\_randstate\_t\ state, mpz\_t\ max)$ – получение рандомного числа в диапазоне от 0 до 2^{max-1} с помощью $state$ и запись результата в r
4. $void\ gmp\_randclear\ (gmp\_randstate\_t\ state)$ – освобождение памяти, выделенной под $state$
5. $int\ mpz\_sgn\ (const\ mpz\_t\ a)$ – сравнение с нулем
6. $int\ mpz\_tstbit\ (const\ mpz\_t\ a, mp\_bitcnt\_t\ bit\_index)$ – *проверяет значение бита* $index$ в op
7. $size\_t\ mpz\_sizeinbase\ (mpz\_srcptr\ a, int\ base)$ – проверяет длину представления числа a в системе счисления с основанием $base$

Список литературы:

1. **Bernstein Daniel и Lange Tanja** Montgomery curves and the Montgomery ladder [В Интернете]. - Technische Universiteit Eindhoven, The Netherlands; University of Illinois at Chicago, USA. - https://eprint.iacr.org/2017/293.pdf.
2. **ГОСТ РЕКОМЕНДАЦИИ ПО СТАНДАРТИЗАЦИИ** Параметры эллиптических кривых для криптографических алгоритмов и протоколов.
3. **Ю. Нестеренко А.** Курс лекций «Методы программной реализации СКЗИ»