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»

**ЗАДАЧА ПЕРВОГО МОДУЛЯ**

по дисциплине «Программирование алгоритмов защиты информации»

Вариант:

* Форма эллиптической кривой - **искривленная форма Эдвардса**
* Библиотека - **LibTomMath**
* Язык - C

Выполнил: студент группы СКБ-172 Сербаев В.Р.

МОСКВА – 2020

**Работа с библиотекой**

**Начало работы**

Для работы с библиотекой LibTomMath я использовал Oracle VM VirtualBox.

Виртуальная машина, которую я применял – Gentoo (64-bit). Ссылка на скачивание:

<http://dev.michiru.ru/tsugumi-2018-10-05.7z>

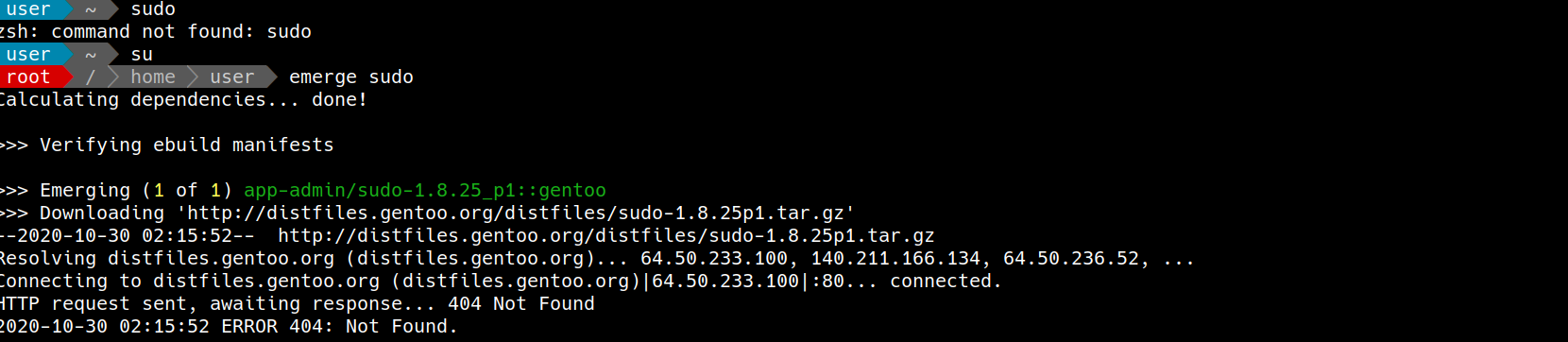
Логин при входе: user

Пароль при входе: user

(также для запуска мне потребовалось включить виртуализацию в BIOS)

После запуска виртуальной машины необходимо выполнить ряд действий, прежде чем начать установку библиотеки.  
В командной строке:

1. su
2. emerge sudo



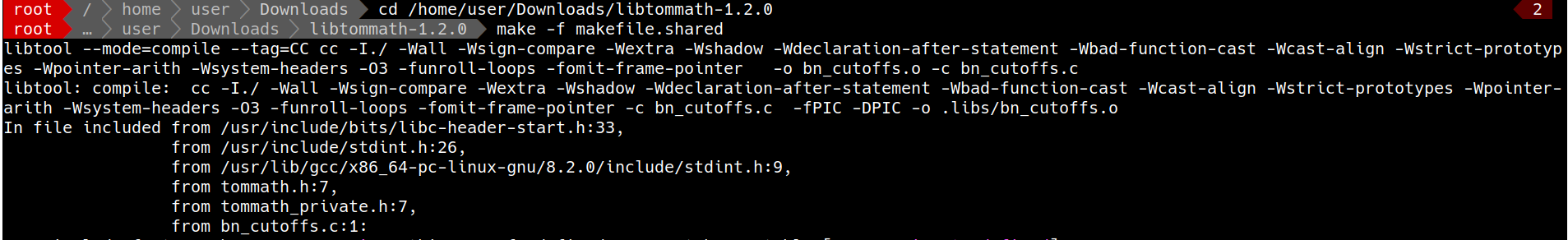
Следующим шагом требуется скачать библиотеку и распаковать (например, в Downloads):

<https://www.libtom.net/>

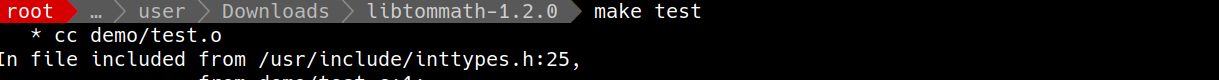
Возвращаемся в командную строку:

переходим в директорию библиотеки

вводим make -f makefile.shared



Вводим make test

****

Переходим в уже встроенный в машину Qt Creator

Создаем новый проект на языке C (Non-Qt Project -> C)

В качестве Build system я использовал qmake

Нажимаем Add Library -> System library -> library file расположен в папке с библиотекой

Теперь в main.c можно добавить заголовочный файл tommath.h и начать работу.

**Функции и типы данных библиотеки LibTomMath**

**mp\_int** – основной тип, применяемый в библиотеке для хранения больших чисел.

typedef struct {

int used, alloc, sign;

mp\_digit \*dp;

} mp\_int;

Библиотечные функции, которые применяются в программе:

**int mp\_init (mp\_int \* a); -** служит для инициализации переменных типа mp\_int:

Пример:

mp\_int number;

int result;

if ((result = mp\_init(&number)) != MP\_OKAY) {

printf("Error initializing the number. %s",

mp\_error\_to\_string(result));

return EXIT\_FAILURE;

}

**void mp\_clear\_multi (mp\_int \* a, mp int \*b, …); -** служит для очистки памяти

**int mp\_init\_multi(mp\_int \*mp, ...); -** функция для инициализации нескольких переменных

**int mp\_toradix (mp\_int \* a, char \*str, int radix); -** функция записывает числовое значение в переменную символьного типа char; radix – в какой системе счисления будет записано большое число.

Пример:



Результатом выполнения будет вывод большого числа.

(arg – mp\_int)

**int mp\_read\_radix (mp\_int \* a, char \*str, int radix); -** функция “заносит” числовое значение, хранящееся в строке в виде отдельных числовых символов в mp\_int так, что с mp\_int, имеющим значение, можно производить различные операции.

Пример:



Теперь с mp\_int two, zero, temp можно производить математические операции.

**mp\_div(&a, &b, &c, &d); -** функция делит a на b и записывает частное в c, остаток в d.

**mp\_mulmod(const mp\_int \*a, const mp\_int \*b, const mp\_int \*c, mp\_int \*d); -** функция выполняет следующую операцию:

d = a \* b (mod c)

**mp\_addmod(const mp\_int \*a, const mp\_int \*b, const mp\_int \*c, mp\_int \*d); -** функция выполняет следующую операцию:

d = a + b (mod c)

**mp\_sqrmod(const mp\_int \*a, const mp\_int \*b, mp\_int \*c);** - функция выполняет следующую операцию:

c = a \* a (mod b)

**mp\_submod(const mp\_int \*a, const mp\_int \*b, const mp\_int \*c, mp\_int \*d) – функция выполняет следующую операцию:**

d = a - b (mod c)

**int mp\_cmp\_mag(mp\_int \* a, mp\_int \* b); -** функция, выполняющая сравнение. a сравнивается с b.

Принимает значение MP\_EQ (a=b), MP\_GT (a>b), MP\_LT (a<b).

Пример использования:

****

mp\_int temp, X: если равны -> вывести “YES”,

иначе вывести “NO”.

**Теоретическая часть**

Эллиптическая криптография — раздел криптографии, в котором изучаются асимметричные криптосистемы, основанные на эллиптических кривых над конечными полями, где образующая точка P используется в качестве открытого ключа, а значение степени k, в которую возводится точка, используется в качестве секретного ключа.

Искривленные кривые Эдвардса лежат в основе схемы электронной подписи под названием EdDSA.

Эллиптическая кривая представлена в форме искривленной формы Эдвардса, определенной над конечным простым полем Fp , где р - простое число и р > 3, если она описывается множеством пар чисел (u, v), u, v принадлежат Fp, удовлетворяющих следующему уравнению:

(eu^2 + v^2) **≡** 1 + du^2 v^2(mod р),

где е, d Fp, ed(e - d) не сравнимо с нулем по модулю р.

Эллиптическая кривая, представленная в форме скрученной кривой Эдвардса имеет также эквивалентное представление в канонической форме, которая задается соответствующими параметрами.

**Параметры и алгоритм возведения в степень**

Параметры для эллиптической кривой в искривленной/скрученной форме Эдвардса: (см. ГОСТ Р 34.10-2012: <https://docplayer.ru/46408167-Tehnicheskiy-komitet-026-zadanie-parametrov-skruchennyh-ellipticheskih-krivyh-edvardsa-v-sootvetstvii-s-gost-r.html>)

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

a, d – коэффициенты эквивалентной скрученной эллиптической кривой в форме Эдвардса.

q – порядок подгруппы простого порядка группы точек эллиптической кривой.

(u, v) – координаты точки P при эквивалентном представлении в форме скрученной кривой Эдвардса.

При выполнении работы был взят следующий набор параметров в десятичной системе счисления id-tc26-gost-3410-2012-256-paramSet A.

p = 11579208923731619542357098500868790785326998466564056403945758400791312963931910 (= 2^256 -617)

a = 110; (иначе, “e”)

d = 272441411047460593183426850116475764599872687847307680943260422341435167538710;

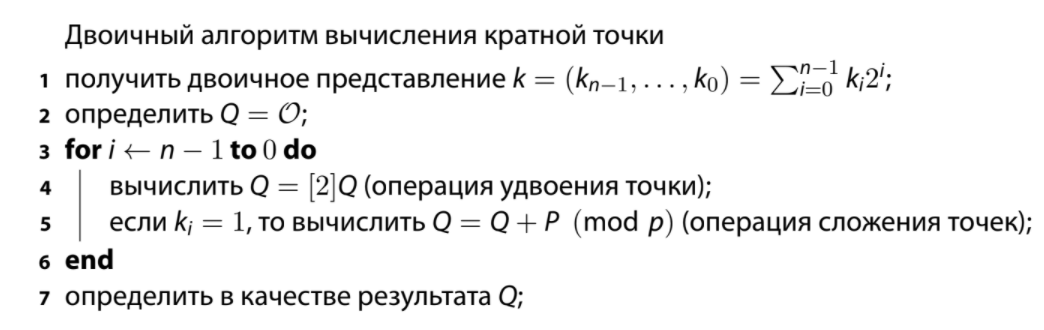
q = 2894802230932904885589274625217197696333856029809225344251215340878553035888710;

u = 1310;

v=

4377914498939898784342877916609043640693419582191518357445422440318617695050310;

Для вычисления кратной точки использовался двоичный алгоритм (лекции, слайд №53):



Точка O – особая выделенная точка с решением на проектной кривой (0; 1; 0).

Операции удвоения точки и сложения точек были взяты отсюда:

<http://hyperelliptic.org/EFD/g1p/auto-twisted-projective.html>

Удвоение - mmadd-2008-bbjlp:

<http://hyperelliptic.org/EFD/g1p/data/twisted/projective/addition/mmadd-2008-bbjlp>

Сложение - mdbl-2008-bbjlp:

<http://hyperelliptic.org/EFD/g1p/data/twisted/projective/doubling/mdbl-2008-bbjlp>

**Структуры и функции программы**

struct Point{ //READY

mp\_int X, Y, Z;

};

Хранит координаты точки;

struct Parameters{ //READY

mp\_int a, d, p, q, u, v;

};

Хранит исходные параметры эллиптической кривой в форме искривленной формы Эдвардса;

void ToText(mp\_int arg); //READY

Небольшая вспомогательная функция для вывода на экран числовых значений, хранящихся в mp\_int;

int BinaryCount(mp\_int arg);

Небольшая вспомогательная функция для получения количества бит в числе в десятичной системе счисления;

struct Point AffinTransform (struct Point P, struct Parameters PR);

Функция переводит координаты точки в афинные координаты;

Возвращает точку;

struct Point DoublePoint(struct Point P, struct Parameters PR) //READY

Функция удваивает точку P; возвращает точку

struct Point AddPoints(struct Point P, struct Point Q, struct Parameters PR) //READY

Функция сложения точек P и Q; возвращает точку

struct Point BinaryMethod(struct Point P, struct Parameters PR, mp\_int seed) //READY

Функция вычисляет кратную точку; возвращает точку;

void TestIfOnCurve(struct Point P, struct Parameters PR) //READY

Функция выполняет проверку, лежит ли результирующая точка на кривой; выводит “YES”, если точка лежит на кривой; иначе – “NO”;

void TestCheckPointCoordinates(struct Point P) //READY

Функция проверяет условие: [q]P = (0,1,0);

void TestCheckNeighbors(struct Point P, struct Parameters PR) //READY

Функция проверяет условия:

[q + 1]P = P и [q − 1]P = −P ;

При выполнении первого выводит “YES”, иначе – “NO”;

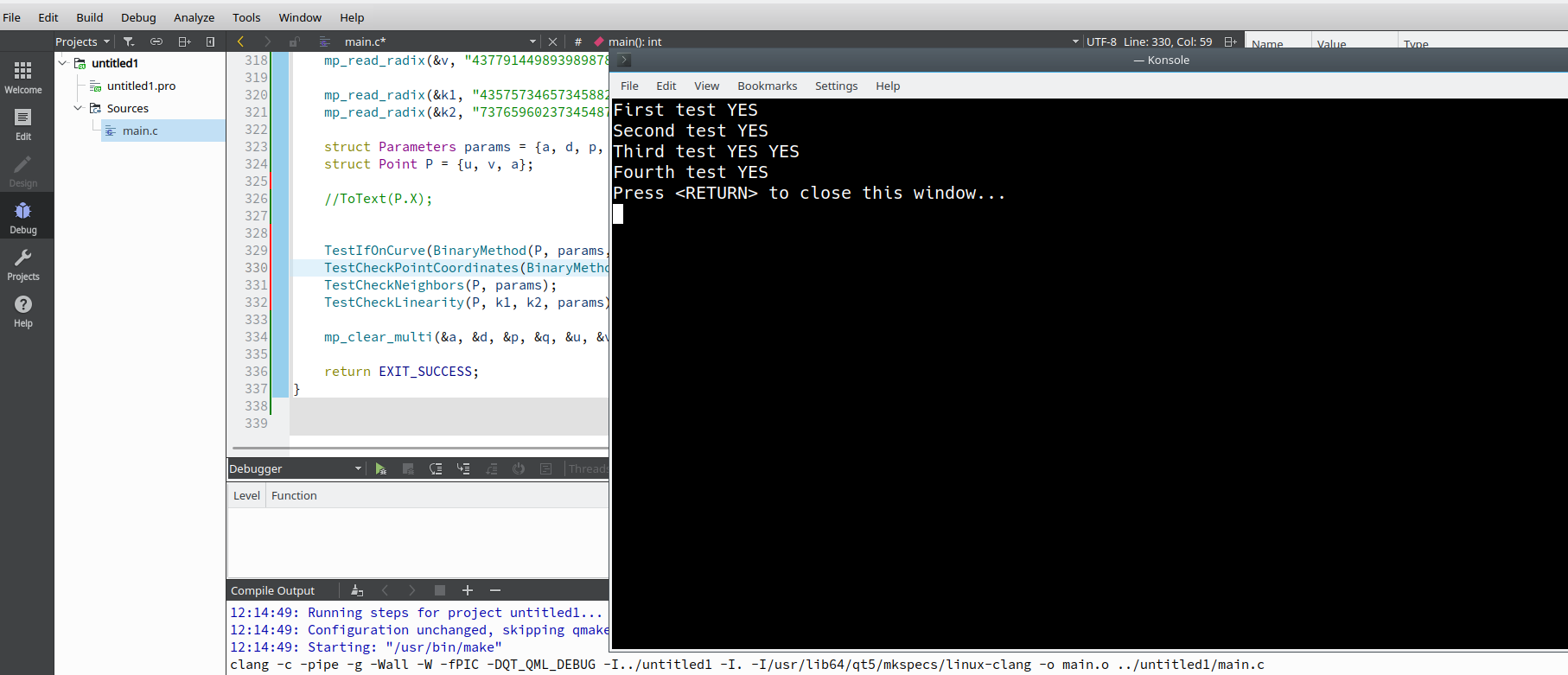
Аналогично для второго;

void TestCheckLinearity(struct Point P, mp\_int k1, mp\_int k2, struct Parameters PR) //READY

Функция для случайных k1 и k2 проверяет выполнение следующего равенства: [k1]P + [k2]P = [k1 + k2]P ;

При выполнении равенства выводит “YES”, иначе – “NO”;

**Выполнение программы**

****

**КОД**

Все функции проверок и вычислений расположены в едином файле **main.c**

В функции **int main()** задаются значения параметров, а также тестирование программы.

**https://github.com/VadimSerb/Project**

**Источники**

Официальная документация библиотеки LibTomMath доступна по ссылке: [https://docviewer.yandex.ru/view/158126255/?page=30&\*=7xuiznXmFLC1blVp%&lang=en](https://docviewer.yandex.ru/view/158126255/?page=30&*=7xuiznXmFLC1blVp%&lang=en)

Установка пакетов в Gentoo:

<https://losst.ru/ustanovka-paketov-v-gentoo>

Р 1323565.1.024—2019:

<https://files.stroyinf.ru/Data2/1/4293730/4293730019.pdf>

Twisted Edwards Curves - Daniel J. Bernstein1, Peter Birkner2, Marc Joye3, Tanja Lange2, and Christiane Peters2:

<https://eprint.iacr.org/2008/013.pdf>

Лекции