МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Алгоритм Мэсси-Омура

ОТЧЁТ
ПО ДИСЦИПЛИНЕ
«КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ»

студента 5 курса 531 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Алексеева Александра Александровича

Преподаватель
профессор, д.ф.-м.н. _____ В. Е. Новиков
подпись, дата

Саратов 2023

СОДЕРЖАНИЕ

1 Теоретическая часть

Цель работы – изучение алгоритма Мэсси-Омуры.

1.1 Первоначальный вариант

Изначально протокол Мэсси-Омуры был описан применительно к мультипликативной группе Z_p^*, где p – простое число, и представлял собой аналог передачи секрета с помощью запираемых на один или два замка ящиков. Суть схемы заключается в следующем: абонент Alice запирает ящик с письмом своим ключом и пересылает ящик абоненту Bob. Абонент Bob, в свою очередь, запирает его своим ключом, и отправляет обратно к Alice. Alice снимает свой замок и направляет ящик к Bob, который снимает свой замок.

1.2 Описание алгоритма

• Выбирается в качестве системного параметра большое простое число p. Абоненты Alice и Bob выбирают случайные числа e_A и e_B (между 0 и p – 1), взаимно простые с $p - 1 = \varphi(p)$, где φ – функция Эйлера.

• С помощью расширенного алгоритма Евклида вычисляются d_A и d_B, обратные числам e_A и e_B по модулю $p - 1$:

$$d_A = e_A^{-1} \bmod (p - 1),$$

$$d_B = e_B^{-1} \bmod (p - 1).$$

Иначе говоря, должны выполняться условия:

$$e_A \cdot d_A \equiv 1 \bmod (p - 1),$$

$$e_B \cdot d_B \equiv 1 \bmod (p - 1).$$

Пары чисел (e_A, d_A), (e_B, d_B) являются секретными ключами абонентов.

$m^{e_A d_A} = m\ (mod\ p)$, так как

$$m^{e_A d_A} = m^{j \cdot \varphi(p)+1} = m^{j \cdot \varphi(p)} \cdot m = m$$

(Первый сомножитель равен 1 по теореме Эйлера). Аналогично $m^{e_B d_A} = m\ (mod\ p)$.

• Абонент Alice посылает сообщение m ($0 < m < p - 1$) абоненту Bob.

Alice шифрует своё сообщение первым ключом: $m_1 = m^{e_A} \pmod{p}$ ($0 < m_1 < p$) и пересылает m_1 абоненту Bob.

• Bob шифрует вторым ключом: $m_2 = m_1^{e_B} \pmod{p}$ и пересылает обратно к Alice.

• Alice «снимает первый замок» с помощью второго секретного ключа:

$$m_3 = m_2^{d_A} \pmod{p} = m^{e_A e_B d_A} \pmod{p}.$$

• Bob «снимает свой первый замок» с помощью второго секретного ключа:

$$m_4 = m_3^{d_B} \pmod{p} = m^{d_B e_A e_B d_A} \pmod{p}.$$

Итого: абоненту Bob доставлено секретное сообщение m от Alice.

2 Описание программы

Программа, представленная ниже, содержит следующие функции:

• powClosed(*x*, *y*, *mod*) – возводит число *x* в степень *y* по модулю *mod*;

• usualEuclid(*a*, *b*) – вычисление НОД чисел *a* и *b* обычным алгоритмом Евклида;

• advancedEuclid(*a*, *p*) – вычисление обратного элемента для *a* в поле *p* расширенным алгоритмом Евклида;

• miller_rabin(*n*, *k* = 10) – проверка числа *n* на простоту с вероятностью $\frac{1}{2^k}$;

• generateP(*m*) – генерация открытого ключа *p* на основе сообщения *m*;

• masseyOmura(*str*) – реализация протокола Мэсси-Омура при передаче сообщения *str*.

2.1 Примеры работы программы

```
        Алгоритм Мэсси-Омура
Введите сообщение: Hallow, Bob!

Сообщение m = 675972727584362261756123
          p = 12173151214491575413614787

Ключи Alice: eA = 1052276489, dA = 7861252877647710500856815
Ключи   Bob: eB = 1998058085, dB = 3056950262259785418132527

Alice шифрует сообщение m1 = 774960925997757153514857 7 и отправляет его абоненту Bob
Bob шифрует сообщение m2 = 17248283793759116626389 08 и отправляет его абоненту Alice
Alice расшифровывает сообщение m3 = 45165563741011156770225 38 и отправляет его абоненту Bob
Bob расшифровывает сообщение m4 = 675972727584362261756123 = hallow, bob! и получает секретное сообщение от Alice
```

```
        Алгоритм Мэсси-Омура
Введите сообщение: How are you?

Сообщение m = 677584225978642286758246
          p = 45316338138089064947687351

Ключи Alice: eA = 1737259521, dA = 6282999667329937220260081
Ключи   Bob: eB = 1287440287, dB = 2516477853381420557981 1773

Alice шифрует сообщение m1 = 637806980043192646609373 5 и отправляет его абоненту Bob
Bob шифрует сообщение m2 = 43238172690167682709450672 и отправляет его абоненту Alice
Alice расшифровывает сообщение m3 = 31299800300062199245140775 и отправляет его абоненту Bob
Bob расшифровывает сообщение m4 = 677584225978642286758246 = how are you? и получает секретное сообщение от Alice
```

3 Листинг кода

```
#include "iostream"
#include "vector"
#include "map"
#include "string"
#include "set"
#include "boost/multiprecision/cpp_int.hpp"

using namespace std;
using namespace boost::multiprecision;

map <char, string> book{ {'0', "11"}, {'1', "12"}, {'2', "13"}, {'3', "14"},
{'4', "15"}, {'5', "16"}, {'6', "17"}, {'7', "18"}, {'8', "19"},
                         {'9', "21"}, {' ', "22"}, {'!', "23"}, {'"', "24"},
{'#', "25"}, {'$', "26"}, {'%', "27"}, {'^', "28"}, {'&', "29"},
                         {'\'', "31"}, {'(', "32"}, {')', "33"}, {'*', "34"},
{'+', "35"}, {',', "36"}, {'-', "37"}, {'.', "38"}, {'/', "39"},
                         {':', "41"}, {';', "42"}, {'<', "43"}, {'=', "44"},
{'>', "45"}, {'?', "46"}, {'@', "47"}, {'[', "48"}, {'\\', "49"},
                         {']', "51"}, {'_', "52"}, {'`', "53"}, {'{', "54"},
{'}', "55"}, {'|', "56"}, {'~', "57"}, {'\n', "58"}, {'a', "59"},
                         {'b', "61"}, {'c', "62"}, {'d', "63"}, {'e', "64"},
{'f', "65"}, {'g', "66"}, {'h', "67"}, {'i', "68"}, {'j', "69"},
                         {'k', "71"}, {'l', "72"}, {'m', "73"}, {'n', "74"},
{'o', "75"}, {'p', "76"}, {'q', "77"}, {'r', "78"}, {'s', "79"},
                         {'t', "81"}, {'u', "82"}, {'v', "83"}, {'w', "84"},
{'x', "85"}, {'y', "86"}, {'z', "87"} };

map <string, char> bookRvs{ {"11", '0'}, {"12", '1'}, {"13", '2'}, {"14", '3'},
{"15", '4'}, {"16", '5'}, {"17", '6'}, {"18", '7'}, {"19", '8'},
                           {"21", '9'}, {"22", ' '}, {"23", '!'}, {"24", '"'},
{"25", '#'}, {"26", '$'}, {"27", '%'}, {"28", '^'}, {"29", '&'},
                           {"31", '\''}, {"32", '('}, {"33", ')'}, {"34", '*'},
{"35", '+'}, {"36", ','}, {"37", '-'}, {"38", '.'}, {"39", '/'},
                           {"41", ':'}, {"42", ';'}, {"43", '<'}, {"44", '='},
{"45", '>'}, {"46", '?'}, {"47", '@'}, {"48", '['}, {"49", '\\'},
                           {"51", ']'}, {"52", '_'}, {"53", '`'}, {"54", '{'},
{"55", '}'}, {"56", '|'}, {"57", '~'}, {"58", '\n'}, {"59", 'a'},
                           {"61", 'b'}, {"62", 'c'}, {"63", 'd'}, {"64", 'e'},
{"65", 'f'}, {"66", 'g'}, {"67", 'h'}, {"68", 'i'}, {"69", 'j'},
                           {"71", 'k'}, {"72", 'l'}, {"73", 'm'}, {"74", 'n'},
{"75", 'o'}, {"76", 'p'}, {"77", 'q'}, {"78", 'r'}, {"79", 's'},
                           {"81", 't'}, {"82", 'u'}, {"83", 'v'}, {"84", 'w'},
{"85", 'x'}, {"86", 'y'}, {"87", 'z'} };

set <char> upSymbs{ {'A'}, {'B'}, {'C'}, {'D'}, {'E'}, {'F'}, {'G'}, {'H'},
{'I'}, {'J'}, {'K'}, {'L'}, {'M'}, {'N'}, {'O'}, {'P'}, {'Q'}, {'R'}, {'S'},
                    {'T'}, {'U'}, {'V'}, {'W'}, {'X'}, {'Y'}, {'Z'} };


vector <cpp_int> deg2(cpp_int el, cpp_int n) {
    vector <cpp_int> res;
    while (n != 0) {
        if (n / el == 1) {
            res.push_back(el);
            n -= el;
            el = 1;
        }
        else
            el *= 2;
    }
    return res;
```

```
}


cpp_int multMod(cpp_int n, cpp_int mod, vector <pair <cpp_int, cpp_int>> lst) {
    if (lst.size() == 1) {
        cpp_int res = 1;
        for (int i = 0; i < lst[0].second; i++)
            res = res * lst[0].first % mod;
        return res;
    }
    else if (lst[0].second == 1) {
        cpp_int el = lst[0].first;
        lst.erase(lst.begin());
        return (el * multMod(n, mod, lst)) % mod;
    }
    else {
        for (int i = 0; i < lst.size(); i++)
            if (lst[i].second > 1) {
                lst[i].first = (lst[i].first * lst[i].first) % mod;
                lst[i].second /= 2;
            }
        return multMod(n, mod, lst);
    }
}


cpp_int powClosed(cpp_int x, cpp_int y, cpp_int mod) {
    if (y == 0)
        return 1;

    vector <cpp_int> lst = deg2(1, y);
    vector <pair <cpp_int, cpp_int>> xDegs;
    for (int i = 0; i < lst.size(); i++)
        xDegs.push_back(make_pair(x, lst[i]));

    cpp_int res = multMod(x, mod, xDegs);
    return res;
}


cpp_int usualEuclid(cpp_int a, cpp_int b) {
    if (a < b)
        swap(a, b);
    if (a < 0 || b < 0)
        throw string{ "Выполнение невозможно: a < 0 или b < 0" };
    else if (b == 0)
        return a;

    cpp_int r = a % b;
    return usualEuclid(b, r);
}


pair <cpp_int, cpp_int> advancedEuclid(cpp_int a, cpp_int b) {
    if (a < 0 || b < 0)
        throw string{ "Выполнение невозможно: a < 0 или b < 0" };

    cpp_int q, aPrev = a, aCur = b, aNext = -1;
    cpp_int xPrev = 1, xCur = 0, xNext;
    cpp_int yPrev = 0, yCur = 1, yNext;
    while (aNext != 0) {
        q = aPrev / aCur;
        aNext = aPrev % aCur;
```

```
        aPrev = aCur; aCur = aNext;

        xNext = xPrev - (xCur * q);
        xPrev = xCur; xCur = xNext;

        yNext = yPrev - (yCur * q);
        yPrev = yCur; yCur = yNext;
    }

    return make_pair(xPrev, yPrev);
}


cpp_int decForm(string x) {
    cpp_int res = 0, deg = 1;
    if (x.back() == '1')
        res += 1;
    for (int i = 1; i < x.length(); i++) {
        deg = deg * 2;
        if (x[x.length() - i - 1] == '1')
            res += deg;
    }
    return res;
}


bool miller_rabin(cpp_int n, int k = 10) {
    if (n == 0 || n == 1)
        return false;

    cpp_int d = n - 1;
    cpp_int s = 0;
    while (d % 2 == 0) {
        s++;
        d = d / 2;
    }

    cpp_int nDec = n - 1;
    for (int i = 0; i < k; i++) {
        cpp_int a = rand() % nDec;
        if (a == 0 || a == 1)
            a = a + 2;

        cpp_int x = powClosed(a, d, n);
        if (x == 1 || x == nDec)
            continue;

        bool flag = false;
        for (int j = 0; j < s; j++) {
            x = (x * x) % n;
            if (x == nDec) {
                flag = true;
                break;
            }
        }
        if (!flag)
            return false;
    }

    return true;
}
```

```
cpp_int generateP(cpp_int m) {
    cpp_int q = rand();
genP:
    while (!miller_rabin(q))
        q++;

    cpp_int s, p = 2, pDec;
    while (!miller_rabin(p)) {
        string sBin = "";
        int sBinSize = rand() % 50 + 1;
        for (int i = 0; i < sBinSize; i++)
            sBin = sBin + to_string(rand() % 2);
        s = decForm(sBin);

        p = (q * s) + 1;
        pDec = p - 1;
    }

    if (m >= p - 1) {
        q *= 2;
        goto genP;
    }
    return p;
}


void masseyOmura(string str) {
    string codeSymbs = "";
    for (int i = 0; i < str.length(); i++) {
        if (upSymbs.find(str[i]) != upSymbs.end())
            codeSymbs += book[char(str[i] + 32)];
        else
            codeSymbs += book[str[i]];
    }

    cpp_int m(codeSymbs);
    cpp_int p = generateP(m);

    cpp_int eA = abs(rand() * rand() * rand()) % (p - 1);
    while (usualEuclid(eA, p - 1) != 1)
        eA = (eA + 1) % p;
    cpp_int dA = advancedEuclid(eA, p - 1).first;
    while (dA < 0)
        dA += p - 1;

    cpp_int eB = abs(rand() * rand() * rand()) % (p - 1);
    while (usualEuclid(eB, p - 1) != 1)
        eB = (eB + 1) % p;
    cpp_int dB = advancedEuclid(eB, p - 1).first;
    while (dB < 0)
        dB += p - 1;

    cout << "\nСообщение m = " << m << "\n           p = " << p;
    cout << "\n\nКлючи Alice: eA = " << eA << ", dA = " << dA << "\nКлючи   Bob:
eB = " << eB << ", dB = " << dB;

    cpp_int m1 = powClosed(m, eA, p);
    cout << "\n\nAlice шифрует сообщение m1 = " << m1 << " и отправляет его
абоненту Bob";
    cpp_int m2 = powClosed(m1, eB, p);
    cout << "\nBob шифрует сообщение m2 = " << m2 << " и отправляет его абоненту
Alice";
    cpp_int m3 = powClosed(m2, dA, p);
```

```
    cout << "\nAlice расшифровывает сообщение m3 = " << m3 << " и отправляет его
абоненту Bob";

    cpp_int m4 = powClosed(m3, dB, p);
    codeSymbs = to_string(m4);
    string res = "";
    for (int i = 0; i < codeSymbs.length(); i += 2)
        res += bookRvs[codeSymbs.substr(i, 2)];
    cout << "\nBob расшифровывает сообщение m4 = " << m4 << " = " << res << " и
получает секретное сообщение от Alice";
}


int main() {
    srand(time(0));
        setlocale(LC_ALL, "ru");
        cout << "\tАлгоритм Мэсси-Омура \nВведите сообщение: ";
    string str;
    getline(cin, str);

    masseyOmura(str);
    cout << endl;
    return 0;
}
```