

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

Схема подписи Гиллу-Кискате

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

аспирант

подпись, дата

Р. А. Фарахутдинов

Саратов 2023

СОДЕРЖАНИЕ

1 Теоретическая часть.....	3
1.1 Описание алгоритма.....	3
2 Описание программы.....	4
2.1 Пример работы программы.....	4
3 Листинг кода.....	5

1 Теоретическая часть

Цель работы – изучение схемы подписи Гиллу-Кискате.

1.1 Описание алгоритма

Алиса отправляет Бобу свои атрибуты J . Алисе необходимо убедить Боба, что это именно её атрибуты. Для этого она доказывает своё знание секрета B . Для этого сторонам потребуется всего 1 раунд.

Алгоритм создания открытого и закрытого ключей

1. Центр доверия T выбирает 2 различных случайных больших простых числа p и q , после чего вычисляет их произведение $n = p \cdot q$.
2. T выбирает целое число v ($1 < v < \varphi(n)$), взаимно простое со значением функции $\varphi(n)$. Функция $\varphi(n)$ является функцией Эйлера.
3. T вычисляет $s = e^{-1} \bmod \varphi(n)$ и секрет $B = J^s \bmod n$.
4. Двойка $\{n, v\}$ публикуется в качестве открытого ключа.
5. B играет роль закрытого ключа, и передаётся Алисе.

Обмен сообщениями

- (1) Алиса выбирает случайное целое r , находящееся в диапазоне от 1 до $n-1$. Она вычисляет $T = r^v \bmod n$.
- (2) Алиса вычисляет $d = H(M, T)$, где M – подписываемое сообщение, а $H(x)$ – однонаправленная хэш-функция. Значение d , полученное с помощью хэш-функции, должно быть в диапазоне от 0 до $v-1$. Если выход хэш-функции выходит за этот диапазон, он должен быть приведён по модулю v .
- (3) Алиса вычисляет $D = rB^d \bmod n$. Подпись состоит из сообщения M , двух вычисленных значений, d and D , и её атрибутов J . Она посылает подпись Бобу.
- (4) Боб вычисляет $T' = D^v J^d \bmod n$. Затем он вычисляет $d' = H(M, T')$. Если $d = d'$, то Алиса знает B , и её подпись действительна.

2 Описание программы

Программа, представленная ниже, содержит следующие функции:

- $\text{powMy}(x, y)$ – возводит число x в степень y ;
- $\text{powClosed}(x, y, \text{mod})$ – возводит число x в степень y по модулю mod ;
- $\text{usualEuclid}(a, b)$ – вычисление НОД чисел a и b обычным алгоритмом Евклида;
- $\text{advancedEuclid}(a, b)$ – вычисляет НОД(a, b) и представляет его в виде $ax + by$;
- $\text{funEuler}(n)$ – вычисление функции Эйлера от n ;
- $\text{miller_rabin}(n, k = 10)$ – проверка числа n на простоту с вероятностью $\frac{1}{2^k}$;
- $\text{generatesimpleNum}()$ – генерация простого числа размером 32-64 бит;
- $\text{guillouQuisquater}(J, M)$ – подпись сообщения M с атрибутами J ;

2.1 Примеры работы программы

```
Схема подписи Гиллу-Кискате
Введите атрибуты пользователя, который хочет подписать сообщение: Alice
Введите сообщение: Hallow, World!

Хэш подписываемого сообщения M: 2805762339877688587
Хэш атрибутов J: 1313091536507228519

Открытый ключ {n, v} = {19669599361692063377511143, 86796557211961}
Закрытый ключ B = 16984496950541634463935421
J * B^v (mod n) = 1

Алиса выбирает случайное r = 2123719584 и вычисляет T = r^v (mod n) = 7467405151720427380080650
Алиса вычисляет d = H(M, T) = 42595691147390
Алиса вычисляет D = r * B^d (mod n) = 14522788712745407941751032

Алиса создала подпись {M, d, D, J} = {Hallow, World!, 42595691147390, 14522788712745407941751032, Alice}
Боб вычисляет T' = D^v * J^d (mod n) = 7467405151720427380080650
Боб вычисляет d' = H(M, T') = 42595691147390. Если d = d', то подпись Алисы действительна
```

```
Схема подписи Гиллу-Кискате
Введите атрибуты пользователя, который хочет подписать сообщение: Peggi
Введите сообщение: How are you?

Хэш подписываемого сообщения M: 12475862516213478689
Хэш атрибутов J: 8881089579463410375

Открытый ключ {n, v} = {16362560009014054470637081006687, 7483219059563174527}
Закрытый ключ B = 6550803408035680770505251303761
J * B^v (mod n) = 1

Алиса выбирает случайное r = 307392972 и вычисляет T = r^v (mod n) = 12160040270436678762500268020209
Алиса вычисляет d = H(M, T) = 6901744808311389607
Алиса вычисляет D = r * B^d (mod n) = 6495855524621606180876102216212

Алиса создала подпись {M, d, D, J} = {How are you?, 6901744808311389607, 6495855524621606180876102216212, Peggi}
Боб вычисляет T' = D^v * J^d (mod n) = 12160040270436678762500268020209
Боб вычисляет d' = H(M, T') = 6901744808311389607. Если d = d', то подпись Алисы действительна
```

3 Листинг кода

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

using namespace std;
using namespace boost::multiprecision;

cpp_int powMy(cpp_int x, cpp_int y) {
    cpp_int res = 1;
    for (int i = 0; i < y; i++)
        res *= x;
    return res;
}

vector <cpp_int> deg2(cpp_int el, cpp_int n) { //Раскладываем число на степени
двойки
    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)
{ //Умножаем число по модулю
    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) { //Возводим число в степени
по модулю
    if (y == 0)
        return 1;

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

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

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;
}

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 funEuler(cpp_int n) {
    cpp_int res = 1;
    for (int i = 2; i < n; i++)
        if (usualEuclid(n, i) == 1)
            res++;
    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 generateSimpleNum() {
    cpp_int q = rand() % 1000;
    while (funEuler(q) != q - 1)
        q++;

    cpp_int s, n = 2, nDec;
    while (!miller_rabin(n)) {
        string sBin = "";
        int sBinSize = 32 + rand() % 32;
        for (int i = 0; i < sBinSize; i++)
            sBin = sBin + to_string(rand() % 2);
        s = decForm(sBin);

        n = (q * s) + 1;
        nDec = n - 1;
    }
}

```

```

        return n;
    }

void guillouQuisquater(string J, string M) {
    hash <string> hashStr;
    cpp_int hashM(hashStr(M));
    cout << "\nХэш подписываемого сообщения M: " << hashM;
    cpp_int hashJ(hashStr(J));
    cout << "\nХэш атрибутов J: " << hashJ;

    cpp_int p = generateSimpleNum(), q = generateSimpleNum();
    cpp_int n = p * q;
    cpp_int phiN = (p - 1) * (q - 1);

    cpp_int v = generateSimpleNum();
    cout << "\n\nОткрытый ключ {n, v} = {" << n << ", " << v << "}";

    cpp_int vRev = advancedEuclid(v, phiN).first;
    if (vRev < 0)
        vRev += phiN;
    cpp_int B = powClosed(advancedEuclid(hashJ, n).first, vRev, n);
    if (B < 0)
        B += n;
    cout << "\nЗакрытый ключ B = " << B;
    cout << "\nJ * B^v (mod n) = " << hashJ * powClosed(B, v, n) % n;

    cpp_int r = abs(rand() * rand() * rand()) % n;
    cpp_int T = powClosed(r, v, n);
    cout << "\n\nАлиса выбирает случайное r = " << r << " и вычисляет T = r^v (mod n) = " << T;

    hash <cpp_int> hashCpp_int;
    cpp_int d = hashCpp_int(hashM * T) % v;
    cout << "\nАлиса вычисляет d = H(M, T) = " << d;

    cpp_int D = r * powClosed(B, d, n) % n;
    cout << "\nАлиса вычисляет D = r * B^d (mod n) = " << D;

    cout << "\n\nАлиса создала подпись {M, d, D, J} = {" << M << ", " << d
    << ", " << D << ", " << J << "}";

    cpp_int T_ = powClosed(D, v, n) * powClosed(hashJ, d, n) % n;
    cpp_int d_ = hashCpp_int(hashM * T_) % v;
    cout << "\nБоб вычисляет T' = D^v * J^d (mod n) = " << T_ << "\nБоб
    вычисляет d' = H(M, T') = " << d_;
    cout << ". Если d = d', то подпись Алисы действительна";
}

int main() {
    srand(time(0));
    setlocale(LC_ALL, "ru");
    cout << "\tСхема подписи Гиллу-Кискате \nВведите атрибуты пользователя,
    который хочет подписать сообщение: ";
    string J;
    getline(cin, J);
    cout << "Введите сообщение: ";
    string M;
    getline(cin, M);

    guillouQuisquater(J, M);
}

```



```
        cout << endl;  
        return 0;  
    }
```