МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем і мереж

Лабораторна робота №1

з дисципліни: «Екстремальне програмування»

Ключовий обмін Діффі-Хеллмана

|  |
| --- |
| Виконав:  Студент групи ЕКП-ІСМ-301  Гасій В. Ю.  Прийняв:  Щербак С. С. |

Львів – 2020

**Алгоритм обміну ключами за схемою Діффі-Хеллмана**

У 1976 році була опублікована робота американських математиків У. Діффі та М.Е. Хеллмана «Нові напрямки в криптографії». У ній вони запропонували конструкцію так званого «відкритого розподілу ключів».

Мета алгоритму полягає в тому, щоб два учасники могли безпечно обмінюватися ключем, який надалі може використовуватися в будь-якому алгоритмі симетричного шифрування. Сам алгоритм Діффі-Хеллмана може застосовуватися лише для обміну ключами. Алгоритм побудований на тому, що складно рахувати дискретні алгоритми.

Безпека обміну ключами в алгоритмі Діффі-Хеллмана витікає з того факту, що, хоча відносно легко обчислити експоненти по модулю простого числа, проте важко обчислити дискретні логарифми. Для великих простих чисел задача вважається нерозв'язною.

Припустимо, що двом абонентам необхідно провести конфіденційну переписку, а в їх розпорядженні немає початково обумовленого секретного ключа. Однак між ними існує канал, що захищений від модифікації, тобто дані, які передаються по ньому, можуть бути «прослухані», але не змінені. У цьому випадку дві сторони можуть створити однаковий секретний ключ, при цьому жодного разу не передавши його по мережі, згідно з наступним алгоритмом.

Алгоритм полягає в наступному:

1. Задаються глобальні відкриті елементи:

1) *n* – випадкове велике просте число;

2) 𝑔 – початковий корінь *n*.

2. Обчислюється ключ абонентом *А*:

1) вибирається велике секретне число();

2) обчислення відкритого значення:: = mod n.

3. Обчислюється ключ абонентом *B*:

1) вибирається велике секретне число ();

2) обчислення відкрите значення: :: = mod n.

4. Обчислюється секретний ключ абонентом *A: K =mod n*

5. Обчислюється секретний ключ абонентом *B*: *K =mod n*

Необхідно відзначити, що алгоритм Діффі-Хеллмана працює тільки на лініях зв'язку, які надійно захищені від модифікації.

**Завдання**

Розробити план реалізації ключового обміну Діффі-Хеллмана. Згенерувати велике n-бітне число.

**Розв’язання**

Щоб реалізувати ключовий обмін Діффі-Хеллмана потрібно:

1. Згенерувати просте велике число:

* Згенерувати велике n-бітне число
* Реалізувати тест Рабіна-Міллера
* Виконати алгоритм генерації простого числа

1. Побудувати початковий корінь по модулю n:

* Реалізувати функцію Ейлера
* Виконати алгоритм побудови початкового кореня по модулю n

Main.cpp

#include "Include.h"

using boost::multiprecision::cpp\_int;

int main()

{

std::cout << create\_number(100) << std::endl;

system("pause");

return 0;

}

Include.h

#pragma once

#include <iostream>

#include <boost/multiprecision/cpp\_int.hpp>

#include "Prime\_Number.h"

Prime\_Number.h

#pragma once

#include <iostream>

#include <cmath>

#include <boost/multiprecision/cpp\_int.hpp>

#include <ctime>

#include <vector>

using boost::multiprecision::cpp\_int;

cpp\_int create\_number(int n);

cpp\_int convert\_binary\_to\_decimal(cpp\_int number);

Prime\_Number.cpp

#include "Prime\_Number.h"

using boost::multiprecision::cpp\_int;

cpp\_int create\_number(int n)

{

srand(time(NULL));

cpp\_int number;

std::string temp = "\0";

temp += "1";

for (int i = 0; i < n - 1; i++)

{

temp += std::to\_string(rand() % 2);

}

number = convert\_binary\_to\_decimal(cpp\_int(temp));

return number;

}

cpp\_int convert\_binary\_to\_decimal(cpp\_int number)

{

cpp\_int decimalNumber = 0, remainder;

int i = 0;

while (number != 0)

{

remainder = number % 10;

number /= 10;

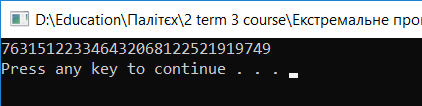
decimalNumber += remainder \* cpp\_int(pow(2, i));

++i;

}

return decimalNumber;

}



**Висновок**: на цій лабораторній роботі було визначено розроблено план реалізації ключового обміну Діффі-Хеллмана та було згенеровано велике 100-бітне число.