# КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ТАРАСА ШЕВЧЕНКА



## ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

**Кафедра прикладних інформаційних систем**

**Звіт до лабораторної роботи №12**

# з курсу

**«Безпека мереж і комп’ютерних систем»**

*Студента 2 курсу*

*групи ПП-21 спеціальності 122 «Комп'ютерні науки» ОП «Прикладне програмування»*

%username%

*Викладач:*

проф. Сайко В.Г.

## Київ – 202

**1.Назва роботи**

Асиметричні криптосистеми.Алгоритм RSA. Криптосистема Ель-Гамаля. Криптосистеми, засновані на використанні еліптичних кривих.

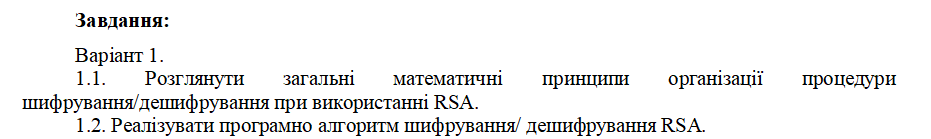
1. **Тема роботи**

Асиметричні криптосистеми.Алгоритм RSA. Криптосистема Ель-Гамаля. Криптосистеми, засновані на використанні еліптичних кривих.

1. **Мета роботи**

Ознайомитись з математичними аспектами алгоритму RSA та навчитися проводити шифрування/дешифрування. Ознайомитися з математичними принципами функціонування алгоритму Ель-Гамаля. Ознайомитися з принципом побудови криптосистем, за використанням еліптичних кривих.

1. **Умова завдання**



1. **Рішення**

Код на С#:

using System;

using System.Security.Cryptography;

namespace Lab12\_RSA\_

{

class Program

{

static void Main(string[] args)

{

var csp = new RSACryptoServiceProvider(2048);

var privKey = csp.ExportParameters(true);

var pubKey = csp.ExportParameters(false);

string pubKeyString;

{

var sw = new System.IO.StringWriter();

var xs = new System.Xml.Serialization.XmlSerializer(typeof(RSAParameters));

xs.Serialize(sw, pubKey);

pubKeyString = sw.ToString();

}

{

var sr = new System.IO.StringReader(pubKeyString);

var xs = new System.Xml.Serialization.XmlSerializer(typeof(RSAParameters));

pubKey = (RSAParameters)xs.Deserialize(sr);

}

csp = new RSACryptoServiceProvider();

csp.ImportParameters(pubKey);

Console.WriteLine("Enter text to encipher:");

var plainTextData = Console.ReadLine();

var bytesPlainTextData = System.Text.Encoding.Unicode.GetBytes(plainTextData);

var bytesCypherText = csp.Encrypt(bytesPlainTextData, false);

var cypherText = Convert.ToBase64String(bytesCypherText);

Console.WriteLine("Enciphered text:");

Console.WriteLine(cypherText);

bytesCypherText = Convert.FromBase64String(cypherText);

csp = new RSACryptoServiceProvider();

csp.ImportParameters(privKey);

bytesPlainTextData = csp.Decrypt(bytesCypherText, false);

plainTextData = System.Text.Encoding.Unicode.GetString(bytesPlainTextData);

Console.WriteLine("Deciphered text:");

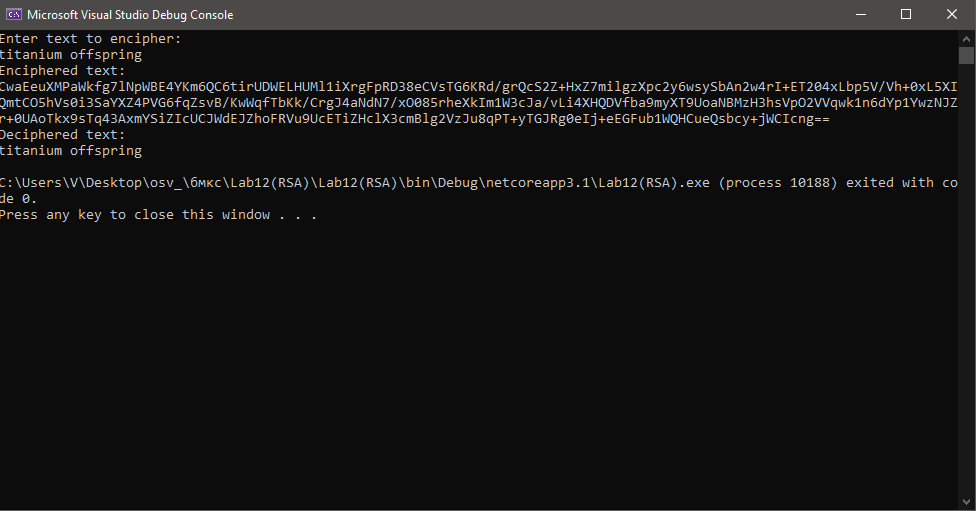
Console.WriteLine(plainTextData);

}

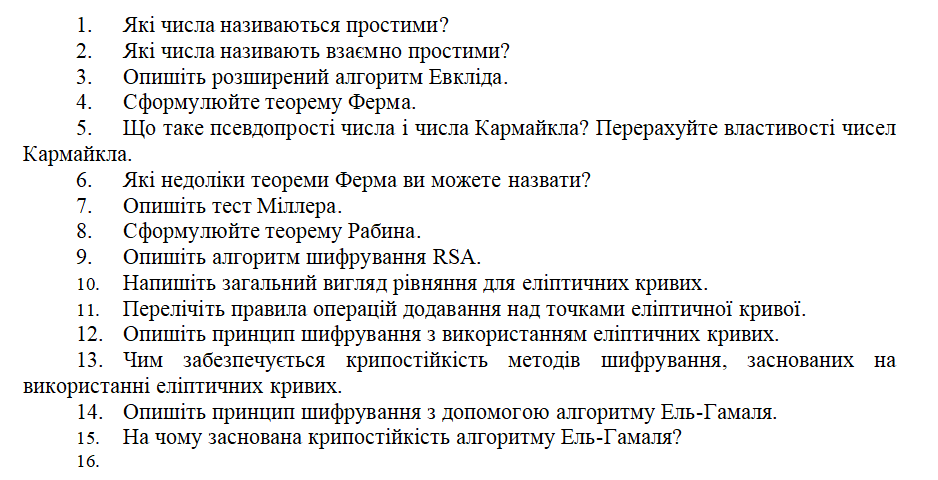
}

}

Результат роботи:



**Контрольні запитання:**



1. Натуральне число, яке має рівно два різних натуральних дільники (лише 1 і саме число)
2. Їх найбільший спільний дільник дорівнює 1.

Алгоритм Евкліда призначений для знаходження найбільшого спільного дільника двох натуральних чисел. Нехай ,, а  – їх найбільший спільний дільник (НСД). Розширений алгоритм Евкліда підраховує не тільки *,* але і два цілих числа  і  таких, що

**

Припустимо, що для знаходження найбільшого спільного дільника чисел  і  ми виконано послідовність ділень з остачею :

* і *

* і *

* і *

* і *

............ . .. . ........... .

* і *

* і *

Визначимо числа  і . Для цього необхідну інформацію зведемо в таблицю:

|  |  |  |  |
| --- | --- | --- | --- |
| ***Остачі*** | ***Частки*** |  |  |
|  |  |  |  |

Відзначимо, що таблиця починається з двох рядків, яких в ній не мало бути. Дійсно, числа, які стоять в першому стовпчику цих рядків не являються остачами в операціях ділення. Ми даємо цим рядкам номери -1 і 0, підкреслюючи цим їх «незаконність».

Cтовпчики  і  заповнюються згідно відношень:

** і *.*

В результаті послідовності ділень з остачею ми отримаємо рівність *НСД* та знайдемо такі цілі числа і , що

**

Отже і .

1. Нехай  – додатне просте число, а  - ціле, що не ділиться на **. Тоді



Зазначимо, що тест Ферма може лише переконати нас в тому, що число має дільники, але не в змозі довести його простоту. *Приклад.* Розглянемо складене число  = 11 × 13 = 341, а основу в тесті Ферма виберемо рівною 2. Тоді 2340 = 1 mod 341, в той час як число 341 не просте. В цьому випадку кажуть, що  *– псевдопросте число* по основі 2.

1. Числа, псевдопростідлявсіх взаємно простих з основ називаються *числами Кармайкла.* Числа Кармайкла мають наступні властивості:

* вони непарні,
* мають якнайменше три простих дільника,
* якщо ділить число Кармайкла , то  ділить .

1. Тест Ферма може лише переконати нас в тому, що число має дільники, але не в змозі довести його простоту.
2. Існує модифікація тесту Ферма, яка обходить проблему псевдопростих чисел – тест Міллера-Рабина.

Нехай  – непарне ціле. За основу візьмемо ціле число , . Перший крок тесту Міллера полягає в визначенні такого показника , для якого:

,

де  – непарне. Далі тест вираховує лишки по модулю  в наступній послідовності степенів:

**

Таким чином, якщо  - просте, то останній лишок в послідовності завжди рівний 1.

1. *Теорема Рабина.* Нехай  – непарне натуральне число. Якщо тест Міллера, застосований до  для більш, ніж основ, не розпізнає в складеного числа, то це число – просте.
2. Алгоритм RSA був запропонований трьома математиками Рональдом Рівестом (R. Rivest), Аді Шаміром (A. Shamir) і Леонардом Адльманом (L. Adleman) в 1977-78 роках.

*Формування ключів*

1. Вибираються два прості (!) числа  і .
2. Обчислюється їх добуток .
3. Обчислюється функція Ейлера .
4. Вибирається довільне число *,* таке, що *НСД*, тобто  взаємнопросте з *.*
5. Методом Евкліда вирішується в цілих числах рівняння .
6. Два числа  – публікуються як відкритий ключ.
7. Пара чисел  –закритий ключ.

*Шифрування*

1. Відправник розбиває своє повідомлення на блоки , рівні  біт (квадратні дужки позначають взяття цілої частини).
2. Подібний блок може бути інтерпретований як число з діапазону . Для кожного такого числа  обчислюється вираз:

**

Послідовність блоків  є зашифрованим повідомленням.

*Дешифрування*

Для того щоб прочитати повідомлення ** досить піднести його до степеня  по модулю :

**

1. 

Для еліптичної групи розглядаються тільки цілі значення від  до  у квадранті невід'ємних чисел, що задовольняють рівнянню за модулем . У загальному випадку список точок створюється за такими правилами:

1. Для кожного значення , що , обчислюється **.

2. Для кожного з отриманих на попередньому кроці значень з'ясовується, чи має це значення квадратний корінь за модулем , Якщо ні, то в немає точок з цим значенням . Якщо ж корінь існує, є два значення у, відповідних операції добування квадратного кореня (винятком є випадок, коли єдиним таким значенням виявляється ). ці значення  і будуть точками.

1. Завданням є шифрування відкритого тексту повідомлення , яке повинне буде пересилатися в вигляді координат точки **.

Припустимо, абонент А хоче відправити абоненту В зашифроване повідомлення. Для цього йому знадобиться відкритий ключ абонента В. Генерація відкритого та закритого ключів абонента В проводиться за наступним алгоритмом:

1. Абонент В вибирає еліптичну групу .*.*

2. Абонент В вибирає генеруючу точку на кривій*.*

3. Абонент В вибирає ціле число ** – його особистий ключ.

4. Абонент В обчислює свій відкритий ключ: *.*

Таємний і відкритий ключ абонента А обчислюються аналогічним чином.

Для того, щоб відправити абоненту В повідомлення **, користувач А вибирає випадкове позитивне ціле число  і обчислює шифрований текст **, що буде складатися з пари точок

**

Зауважимо, що сторона А використовує відкритий ключ сторони В: **. Щоб дешифрувати цей шифрований текст, В помножує першу точку в парі на таємний ключ В і віднімає результат від другої точки:

**

1. Користувач А замаскував повідомлення ** за допомогою додавання до нього **. Ніхто, крім цього користувача, не знає значення , тому, хоча ** і є відкритим ключем, ніхто не зможе прибрати маску *.* Зловмисникові для відновлення повідомлення доведеться обчислити  за даними  і , що представляється складним завданням.

Таким чином, перехід до криптографії за використання еліптичних кривих дозволяє зберегти прийнятну довжину ключа при різкому (на декілька порядків) зрості стійкості криптосистем.

1. Найпростіший алгоритм шифрування, що ґрунтується на дискретному логарифмуванні.

На відміну від RSA в алгоритмі Ель-Гамаля існують деякі відкриті параметри - *параметри домену (ключа)*, які можуть бути використані великим числом користувачів.

*Формування ключів*

1. Вибирається  – велике (1024 біта) просте число.
2. Вибирається  - первісний корінь .

Нагадаємо, що якщо число  є первісним коренем простого числа , то всі числа

**

повинні бути різними й представляти всі цілі числа від 1 до  в деякій перестановці.

Після вибору параметрів домену визначають відкритий і таємний ключі. Таємним ключем може апріорі бути будь-яке натуральне число .

Для визначення відкритого ключа обчислюється число :

*.*

Група чисел  – відкритий ключ.

*Шифрування*

Для шифрування повідомлення  здійснюють наступні кроки:

1. Генерують випадковий ефемерний ключ  - взаємно простий з **, такий що ;
2. Обчислюють ;
3. Знаходять ;
4. Вираховують отриманий шифротекст у вигляді пари *.*

Зауважимо, що при кожному шифруванні застосовується свій короткочасний ключ , тому, шифруючи одне повідомлення двічі, ми отримуємо різні шифротексти.

*Дешифрування*

Щоб розшифрувати , проводять перетворення:

**

*Приклад.* Нехай . В якості пари відкритого і таємного ключа виберемо **та **.

Припустимо, нам потрібно зашифрувати повідомлення, чисельне представлення якого :

1. Генеруємо випадковий ефемерний ключ **;
2. Знаходимо .
3. Рахуємо .
4. Відправляємо шифротекст .

Респондент зможе відновити текст, роблячи наступні обчислення:

**

1. При кожному шифруванні застосовується свій короткочасний ключ , тому, шифруючи одне повідомлення двічі, ми отримуємо різні шифротексти.

**Висновки**

В результаті виконання даної лабораторної роботи я ознайомився з математичними аспектами алгоритму RSA та навчився проводити шифрування/дешифрування. Ознайомився з математичними принципами функціонування алгоритму Ель-Гамаля, з принципом побудови криптосистем з використанням еліптичних кривих. Створив програмний засіб, що реалізовує схожий шифр, використовуючи мову програмування С#. Створив методи у функції main, викликав їх, отримав та оцінив результат.

Вважаю дану лабораторну роботу виконаною в повному обсязі.