КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра програмних систем і технологій

Захист програм та даних

Практична робота № 2.2

Виконали студенти

групи ІПЗ-33

Щука Богдан

Ємець Євгенія

Перевірила

доцент Супрун О.М.

Київ-2018

**Завдання**

Привести послідовність виконання процедур ідентифікації / аутентифікації з використанням таких способів:

- на основі алгоритму RSA;

- за схемою Шнорра;

- за схемою Фейг-Фіата-Шаміра.

При оформленні звіту необхідно привести таблиці генерації ключів і аутентифікації. Для випадкового числа (k або r) прийняти коди, відповідно, 1-ої, 2-ої і 3-ої букви свого прізвища згідно їх положенню в алфавіті.

**Хід роботи**

1. **На основі алгоритму RSA**

Процедура ідентифікації:

1. Користувач Аліса генерує відкриту (e, n) та приватну (d, n) пару ключів;
2. Користувач Аліса передає користувачу Бобу відкритий ключ (e, n).

Процедура аутентифікації:

1. Користувач Боб обирає випадкове число , після чого обчислює та відправляє r користувачу Алісі;
2. Користувач Аліса обчислює та відправляє його користувачу Бобу;
3. Користувач Боб порівнює значення k та p. Якщо вони однакові – аутентифікація пройшла успішно.

**Приклад**

Процедура ідентифікації:

1. Користувач Аліса генерує відкриту (13, 172) та приватну (17, 172) пару ключів;
2. Користувач Аліса передає користувачу Бобу відкритий ключ (13, 172).

Процедура аутентифікації:

1. Користувач Боб обирає випадкове число (положення букви «l» у алфавіті - 12), після чого обчислює (положення букви «u» у алфавіті) та відправляє r користувачу Алісі;
2. Користувач Аліса обчислює та відправляє його користувачу Бобу;
3. Користувач Боб порівнює значення k та p. Значення змінних k (12) та p (12) однакові, тому аутентифікація пройшла успішно.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| e | d | n | k | r | p |
| 13 | 17 | 172 | 12 | 21 | 12 |

Таблиця 1.1 – значення змінних

1. **За схемою Шнорра**

Процедура ідентифікації:

1. Беруться два простих числа p і q для яких ;
2. Береться таємний ключ ;
3. Береться число g для якого ;
4. Обчислюється відкритий ключ y для якого ;
5. Публікація ключа y.

Процедура аутентифікації:

1. Користувач Аліса обирає випадкове число , обчислює число і посилає його користувачу Бобу;
2. Користувач Боб обирає випадкове число e з деяким параметром t, після чого відправляє його користувачу Алісі: ;
3. Користувач Аліса обчислює число і посилає його користувачу Бобу;
4. Користувач Боб перевіряє чи виконується співвідношення . Якщо співвідношення виконується – аутентифікація пройшла успішно.

**Приклад**

Процедура ідентифікації:

1. Беруться два простих числа p (31) і q (11) для яких ;
2. Береться таємний ключ ;
3. Береться число g (3) для якого ;
4. Обчислюється відкритий ключ y (4) для якого ;
5. Публікація ключа y.

Процедура аутентифікації:

1. Користувач Аліса обирає випадкове число , обчислює число і посилає його користувачу Бобу;
2. Користувач Боб обирає випадкове число e з деяким параметром t, після чого відправляє його користувачу Алісі: ;
3. Користувач Аліса обчислює число і посилає його користувачу Бобу;
4. Користувач Боб перевіряє чи виконується співвідношення .

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| p | q | x | g | y | k | e | r | s |
| 31 | 11 | 8 | 1 | 4 | 12 | 4 | 8 | 0 |

Таблиця 2.1 – значення змінних

1. **За схемою Фейга-Фіата-Шаміра**

Обидві процедури виконуються посередником:

Процедура ідентифікації:

1. Посередник обчислює модуль n такий, що n дорівнює добутку двух простих чисел;
2. Посередник обирає відкритий ключ такий, що є квадратичним залишком за модулем n і є зворотнім значенням по модулю n;
3. Посередник визначає таємний ключ s, значення якого є найменшим, що задовольняє співвідношення ;
4. Публікування відкритого ключа та та передача таємного ключа користувачу Алісі.

Процедура аутентифікації:

1. Користувач Аліса обирає випадкове число . Потім обчислює число і посилає його користувачу Бобу;
2. Користувач Боб посилає користувачу Алісі випадкове значення біту;
3. Якщо b = 0, то користувач Аліса посилає користувачу Бобу число r, інакше ;
4. Якщо b = 0, користувач Боб перевіряє співвідношення , інакше - , інакше - .

**Приклад**

Процедура ідентифікації:

1. Посередник обчислює модуль n такий, що n дорівнює добутку двух простих чисел (143);
2. Посередник обирає відкритий ключ такий, що є квадратичним залишком за модулем n і є зворотнім значенням по модулю n (16);
3. Посередник визначає таємний ключ s, значення якого є найменшим, що задовольняє співвідношення (9);
4. Публікування відкритого ключа та та передача таємного ключа користувачу Алісі.

Процедура аутентифікації:

1. Користувач Аліса обирає випадкове число . Потім обчислює число і посилає його користувачу Бобу (1);
2. Користувач Боб посилає користувачу Алісі випадкове значення біту (1);
3. Якщо b = 0, то користувач Аліса посилає користувачу Бобу число r, інакше (в нашому випадку b = 1, тому посилається y = (r \* s));
4. Якщо b = 0, користувач Боб перевіряє співвідношення , інакше - , інакше - .

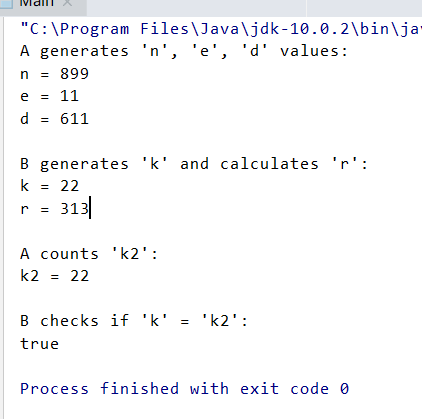
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n | v | s | r | z | b |
| 31 | 11 | 8 | 1 | 4 | 12 |

Таблиця 3.1 – значення змінних

**Лістинг програми**

**import** javafx.util.Pair;  
  
**public class** Main {  
 **public static void** main(String[] args) {  
 **try** {  
*// //RSA  
 //A* System.***out***.println(**"A generates 'n', 'e', 'd' values:"**);  
 Pair<Pair<Integer, Integer>,Pair<Integer, Integer>> keys = RSAAlgorithm.*GetRSAKeys*();  
 Pair<Integer, Integer> publicKey = keys.getKey();  
 Pair<Integer, Integer> privateKey = keys.getValue();  
  
 *//B* System.***out***.println(**"\nB generates 'k' and calculates 'r':"**);  
 Pair<Integer, Integer> Bvalues = RSAAlgorithm.*GetRSAvaluesRandK*(publicKey);  
 **int** r = Bvalues.getKey();  
 **int** k = Bvalues.getValue();  
  
 *//A* System.***out***.println(**"\nA counts 'k2':"**);  
 **int** k2 = RSAAlgorithm.*GetRSAvalueK2*(r, privateKey);  
  
 *//B* System.***out***.println(**"\nB checks if 'k' = 'k2':"**);  
  
 **boolean** RSAauthResult = RSAAlgorithm.*RSAAuthenticationIsValid*(k, k2);  
  
 System.***out***.println(RSAauthResult);  
  
  
 *//Schorr  
// System.out.println("1. Пользователь Алиса генерирует \nзначение публичного ключа y:");  
// int[] shnorrValues = ShnorrAlgorithm.GetShnorrValues();  
// int p = shnorrValues[0];  
// int q = shnorrValues[1];  
// int x = shnorrValues[2];  
// int g = shnorrValues[3];  
// int y = shnorrValues[4];  
//  
// System.out.println("\n2. Пользователь Алиса генерирует \nчисло k и вычисляет число r:");  
// Pair<Integer, Integer> Avalues = ShnorrAlgorithm.GetShnorrValuesKandR(p, q, g);  
// int r = Avalues.getKey();  
// int k = Avalues.getValue();  
//  
// System.out.println("\nПользователь Боб \nвыбирает случайное число e:");  
// int e = ShnorrAlgorithm.GetShnorrValueE();  
//  
// System.out.println("\nПользователь Алиса вычисляет s:");  
// int s = ShnorrAlgorithm.GetShnorrValueS(k, x, e, q);  
//  
// System.out.println("\nПользователь Боб проверяет \nистинность выражения r = (g^s \* y^e):");  
// boolean shnorrResult = ShnorrAlgorithm.AuthenticationIsValid(r, p, g, s, y, e);  
//  
// System.out.println(shnorrResult);  
  
 //Feige-Fiat-Shamir  
// System.out.println("1. Посредник генерирует число n, \nпубличный ключ v, приватный ключ s:");  
// int[] interValues = FeigeFiatShamirAlgorithm.GetFFSIntermediaryValues();  
// int n = interValues[0];  
// int v = interValues[1];  
// int s = interValues[2];  
// int p = interValues[3];  
// int q = interValues[4];  
// System.out.println(String.format("n = %d,\nv = %d,\ns = %d", n, v, s));  
//  
// System.out.println("\n2. Пользователь Алиса получает приватный ключ s.");  
//  
// System.out.println("\nПользователь Алиса выбирает случайное число r \nи вычисляет значение переменной z:");  
// Pair<Integer, Integer> Avalues = FeigeFiatShamirAlgorithm.GetFFSValuesRAndZ(n, p);  
// int r = 21;//Avalues.getKey();  
// int z = Avalues.getValue();  
//  
// System.out.println("\nПользователь Боб отправляет случайное \nзначение бита пользователю Алисе:\n");  
// byte bit = Helper.GetRandomBit();  
// System.out.println("b = " + bit + "\n");  
//  
// System.out.println("Если b = 0 - пользователь Алиса отправляет \nпользователю Бобу r, иначе y:\n");  
// int bitADependedValue = FeigeFiatShamirAlgorithm.GetABitDependedValue(bit, r, s, p);  
//  
// System.out.println("Пользователь Боб проверяет успешность аутентификации:");  
// boolean ffsValid = false;  
// ffsValid = FeigeFiatShamirAlgorithm.AuthentificationIsValid(bit, z, r, p, v, bitADependedValue);  
// System.out.println(true);* } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

Приклад роботи програми



**Висновок**

Протягом виконання лабораторної роботи було проведено ознайомлення з виконанням процедур ідентифікації та аутентифікації на основі алгоритму RSA, за схемою Шнорра та за схемою Фейг-Фіата-Шаміра, була описана послідовність їх виконання.