Національний технічний університет України

«Київський політехнічний інститут»

Фізико технічний інститут
Кафедра математичних методів захисту інформації

МЕТОДИ РЕАЛІЗАЦІЇ КРИПТОГРАФІЧНИХ МЕХАНІЗМІВ

Лабораторна робота №2

Тема: “Реалізація алгоритмів генерації ключів гібридних криптосистем”

Виконали:
Корж Нікіта ФІ-12мн
Тафтай Анастасія ФІ-12мп
Мазур Анастасія ФІ-12мн

Перевірила
Селюх П.В.

Київ - 2021

Мета роботи: дослідження алгоритмів генерації псевдовипадкових послідовностей, тестування простоти чисел та генерації простих чисел з точки зору їх ефективності за часом та можливості використання для гененерації ключів асиметричних криптосистем.

Завдання: для другого типу лабораторних робіт – аналіз стійкості реалізацій ПВЧ та генераторів ключів для обраної бібліотеки. Опис функції генерації ПСП та ключів бібліотеки з описом алгоритму, вхідних та вихідних даних, кодів повернення. Контрольний приклад роботи з функціями

Хід роботи

Bouncy Castle є альтернативним криптографічним провайдером Java. У версії 1.54 бібліотеку було оновлено новим алгоритмом. Версія 1.53 виконувала генерацію ключів так само, як OpenJDK.

Primes.

Прості числа генеруються шляхом випадкової вибірки, поки не буде знайдено просте число. Оскільки інтервал не обмежений ступенями 2 для відбору кандидата використовується вибірка відхилення.

Розмір кандидата перевіряється перед тим, як методом буде виконано тест на первинність. BigInteger.probablePrime (Miller-Rabin with random bases followed by a Lucas test). Тим не менш, кандидати, як і раніше, будуються з використанням версії конструктора BigInteger, який генерує ймовірні прості числа за допомогою сита. Незважаючи на те, що визначеність (кількість тестів Міллера-Рабіна), що дорівнює 1, тести Міллера-Рабіна і Лукаса все одно забезпечать повернення простого числа з великою ймовірністю, де шукали лише випадкового кандидата. Тому в спробі зберегти тести на первинність виконуються більше, ніж потрібно. Це показує, що java.math.BigInteger може отримати вигоду від методу, який генерує прості числа з довільних інтервалів.

Keys.

Прості пари генеруються з області максимального квадрата і обираються незалежно. Це єдина бібліотека, де обчислюється вага Хеммінга модуля. Як повідомляється, деякі складові числа з малою вагою можуть бути слабкими по відношенню до версії факторизації сита числового поля. Потрібно, щоб вага була більше чверті довжини модуля. Значення модуля представляється у несуміжній формі(non-adjacent form), щоб гарантувати, що подання має мінімальну вагу. Крім того, прості числа повинні відрізнятися в деяких із 100 найстарших значущих біт. Забезпечується, що довжина приватної експоненти не менше половини довжини модуля.

Усі імплементовані генератори ключей в бібліотеці описані в https://javadox.com/org.bouncycastle/bcprov-jdk15on/1.51/org/bouncycastle/crypto/generators/package-summary.html

Програмна реалізація:

key AES:

key: [-10, 96]

key RSA:

keyPair private:

RSA Private CRT Key
[e2:f9:0f:cb:c6:22:7c:38:e4:a5:f6:5f:53:30:f3:a0:1a:f0:d0:2e],[56:66:d1:a4]

modulus: 928101291710863405a6002dd8c39d95e330cee127e789f1748ffa1cf8e66df3

public exponent: 10001

keyPair public:

RSA Public Key [e2:f9:0f:cb:c6:22:7c:38:e4:a5:f6:5f:53:30:f3:a0:1a:f0:d0:2e],[56:66:d1:a4]

modulus: 928101291710863405a6002dd8c39d95e330cee127e789f1748ffa1cf8e66df3

public exponent: 10001

Shawe-Taylor Random_Prime Routine
fbbd0e9c90a0f3a34da5a787807fbafe0f8cc7b546895351841e17ff2392fa892bced87e7467dc5
35964efc5be24747f8f79693abc3c828bdb2b342b6945f78c797822466395ef76da1905141e70b
a235d9234cb17e47a95b25ad7c93a4110c20a5122247c3eebaa616e3d4e542587067a0add2fce
15f182216d8cfc99cb04e28c54ffac280e1d92d8fa7cbdc019c54ce2830ec49d5ba8159bf6fedfe1
10d52230c89a87c13408598c86c2f4c96659a4a4293387be1ef2817fafe1eac0d1d9f22fcaba2db
e536bb7056d1be9938308c386cd48a74d55e3654f0778bd721e6acfb3fcd1213461f3e4bb51ff2
7aae713510998d04a8dd2ec7d6aec1ad21d4536f9

Miller-Rabin Probabilistic Primality Test isMRProbablePrime: true

Висновок:

Було досліджено алгоритми генерації псевдовипадкових послідовностей, тестування простоти чисел та генерації простих чисел з точки зору їх ефективності за часом та було проаналізовано як саме генеруються прості числа та ключі.