

Лабораторна робота №6
RSA Algorithm

Мета: Дослідити і реалізувати механізм асиметричного алгоритму шифрування RSA на основі згенерованих ключів.

Розробити додаток обміну таємними посиланнями між двома клієнтами за допомогою алгоритму шифрування RSA

Алгоритм дій

- творити ключову пару: приватний та публічний ключ. Ключ(i) повинні бути розташовані поряд з виконуючим файлом
- текст повідомлення розташовано в файлі message.txt
- клієнт1 зчитує повідомлення, кодує його за допомогою свого приватного ключа та "відправляє" до його клієнту2 (у найпростішому випадку - розташовує кодований файл поряд з виконуючим файлом клієнта2). Для підтвердження у подальшому коректності дешифрування, клієнт1 повідомляє також геш суму оригінального повідомлення.
- клієнт2 зчитує кодоване повідомлення, дешифрує його за допомогою публічного ключа, знаходить геш суму дешифрованого повідомлення, порівнює її з вказаною. Та, нарешті виводить результат о коректності дешифрування.

1. RSA

RSA (аббревіатура від прізвищ Rivest, Shamir та Adleman) — криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел. RSA став першим алгоритмом такого типу, придатним і для шифрування, і для цифрового підпису. Алгоритм застосовується до великої кількості криптографічних застосунків.

Алгоритм RSA складається з 4 етапів: генерації ключів, шифрування, розшифрування та розповсюдження ключів.

Безпека алгоритму RSA побудована на принципі складності факторизації цілих чисел. Алгоритм використовує два ключі — відкритий (public) і секретний (private), разом відкритий і відповідний йому секретний ключі утворюють пари ключів (keypair). Відкритий ключ не потрібно зберігати в таємниці, він використовується для шифрування даних. Якщо повідомлення було зашифровано відкритим ключем, то розшифрувати його можна тільки відповідним секретним ключем.

В ході лабораторної роботи була розроблена програма в форматі клієнт-сервер. Клієнт генерує публічний та приватний ключі за допомогою алгоритму RSA, підключається до сервера та відправляє публічний ключ серверу. Сервер зберігає публічний ключ клієнта та відкриває message.txt з текстом, отримує з нього розмір та весь текст. Геширує його та відправляє

розмір та геш до клієнта. Після цього починає відправляти весь файл у закодованому форматі. Клієнт тим часом отримує тест з файлу розшифровує його та отримує геш. Після отримання всього файлу перевіряє геш файлу та порахований геш. Якщо співпадають, виводить на екран текст.

Результат роботи клієнта:

```
[danilpetrov@MacBook-Pro bin]$ ./client 127.0.0.1 10100
lenPublicKey = 270
lenPrivateKey = 1193
Size of message = 1035
Hash = 66ce4eefc87a364c90a1b4b11962f56e
Hash trees can be used to verify any kind of data stored, handled and transferred in and
received undamaged and unaltered, and even to check that the other peers do not lie and

Hash trees are used in hash-based cryptography. Hash trees are also used in the IPFS, Bt
rial distributed revision control systems; the Tahoe-LAFS backup system; Zeronet; the Bi
such as Apache Cassandra, Riak, and Dynamo.[8] Suggestions have been made to use hash t

The initial Bitcoin implementation of Merkle trees by Satoshi Nakamoto applies the compr
[danilpetrov@MacBook-Pro bin]$
```

Висновок

Дослідив і реалізував механізм асиметричного алгоритму шифрування RSA на основі згенерованих ключів.