НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Методи реалізації криптографічних механізмів

ЛАБОРАТОРНА РОБОТА №4

Виконали студенти
групи ФІ-12мн
Ковалевський Олександр
Ткаченко Артем
Чашницька Марина

Київ — 2021

1 Мета роботи

Отримання практичних навичок побудови гібридних криптосистем

2 Постановка задачі та варіант завдання

Розробити реалізацію асиметричної криптосистеми у відповідності до стандартних вимог Crypto API або стандартів PKCS та дслідити стійкість стндартних криптопровайдерів до атак, що використовують недосконаліть механізмів захисту операційної системи.

Бібліотека PyCrypto під Linux платформу. Стандарт ДСТУ 4145-2002.

3 ДСТУ 4145-2002

Цей стандарт установлює механізм цифрового підписування, оснований на властивостях груп точок еліптичних кривих над полями $GF(2^m)$, та правила застосування цього механізму до повідомлень, що пересилаються каналами зв'язку та/або обробляються у комп'ютеризованих системах загального призначення. Застосування цього стандарту гарантує цілісність підписаного повідомлення, автентичність його автора та неспростовність авторства.

4 Реалізація

```
class API:

    def get_base_point()

    def generate_private_key()
```

```
def generate_public_key()

def export_private_key()

def export_public_key()

def import_private_key()

def import_public_key()

def del_private_key()

def del_public_key()

def sign()

def verify()

def get_public_key()
```

DigitalSignature.py – механізм цифрового підпису

EllipticCurve.py – операції з еліптичною кривою

GaloisField.py – операції з полями Галуа

private key = 0x3147a007bbf7dec2a20c803efa1804687e18e3f07
public key = 0x572d54fcc29816e420824ddd9b1b75ca673004d36
0x2426e61882fb4c31a3911bbd4d003be1d3aa6d859

5 Недоліки

Спеціальні пристрої /dev/random та /dev/urandom надають доступ до засобів генерування випадкових чисел, вбудованих у ядро Linux.

Більшість аналогічних програмних функцій, наприклад функція rand() стандартної бібліотеки мови C, насправді генерують псевдовипадкові числа. Такі числа мають деякі властивості випадкових послідовностей, але їх можна відтворити: достатньо задати те саме ініціалізує значення, щоб отримати однакову послідовність чисел. Така поведінка неминуча, адже внутрішня робота комп'ютера жорстко визначена та передбачувана. Але в ряді програм це вкрай небажано. Можна зламати криптографічний шифр, якщо відтворити послідовність випадкових чисел, що у його основі.

Щоб отримати справжні випадкові числа, необхідне зовнішнє "джерело хаосу". Ядро Linux вимірюючи затримки між діями користувача, зокрема натискання клавіш і переміщеннями миші, здатне генерувати непередбачуваний потік дійсно випадкових чисел. Отримати доступ до цього потопу можна шляхом читання з пристроїв /dev/random та /dev/urandom. Різниця між пристроями виявляється, коли запас випадкових чисел у ядрі Linux закінчується. Якщо спробувати прочитати велику кількість байтів із пристрою /dev/random і при цьому не виконувати жодних дій (не натискати клавіші, не переміщати мишу тощо), система заблокує операцію читання. Тільки коли користувач виявить якусь активність, система згенерує додаткові випадкові числа та передасть їх програмі. На противагу цьому операція читання з пристрою /dev/urandom ніколи не блокується. Якщо в системі закінчуються випадкові числа, Linux використовує криптографічний алгоритм, щоб згенерувати псевдовипадкові числа з останнього ланцюжка випадкових байтів. Проте якщо використовувати емітацію дій користувача, то ці пристрої будут генерувати передбачувану послідовність байтів і це буде впливати на генерацію ключів.

ВИСНОВКИ

Було розроблено реалізацію асиметричної криптосистеми у відповідності до стандартних вимог Crypto API або стандартів PKCS та досліджено стійкість стндартних криптопровайдерів до атак, що використовують недосконалість механізмів захисту операційної системи.

Джерело:

1. https://it.wikireading.ru/34287