НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря

СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №2

З дисципліни «Методи реалізації криптографічних механізмів»

«Реалізація алгоритмів генерації ключів гібридних криптосистем»

Виконав:

студент групи ФБ-41мн

Білик Д. М.

1 Мета роботи

Метою даної лабораторної роботи є дослідження механізмів генерації криптографічно стійких псевдовипадкових послідовностей, алгоритмів тестування простоти чисел та процедур генерації простих чисел, що використовуються для побудови ключів асиметричних криптосистем. Особлива увага приділяється аналізу ефективності за часом та практичній придатності реалізацій бібліотеки PyCrypto під операційною системою Linux.

2 Хід роботи

2.1 Теоретичні відомості

Генерація ключів в асиметричних криптосистемах базується на використанні криптографічно стійких генераторів псевдовипадкових чисел (CSPRNG), які забезпечують високу ентропію, непередбачуваність та стійкість до компрометації внутрішнього стану.

У бібліотеці PyCrypto під Linux джерелом ентропії виступають системні механізми (/dev/urandom, getrandom()), а також внутрішні алгоритмічні підходи, споріднені до генератора Fortuna.

2.2 Опис досліджуваних функцій

Функція	**Призначення**	**Вхідні дані**	**Вихідні дані**
get_random_bytes(N)	Генерація криптографічно стійких випадкових байтів	N – кількість байтів	bytes довжини N
isPrime(N)	Імовірнісна перевірка простоти	N – ціле число	True/False
getPrime(N)	Генерація N-бітного простого числа	N – кількість біт	Просте число
RSA.generate(bits)	Генерація пари RSA-ключів	bits – довжина ключа	Об’єкт RSAKey

2.3 Методика вимірювання швидкодії

Для оцінки швидкодії використовувалась функція time.perf_counter(). Кожен експеримент повторювався 50–100 разів з подальшим усередненням результатів.

2.4 Код для вимірювання швидкодії

```
import time

import statistics as stats

from Crypto.Random import get_random_bytes

from Crypto.PublicKey import RSA
```

```
def bench(label, func, repeats=50, bytes_size=None, calc_cv=False):
    times = []
    for _ in range(repeats):
        t0 = time.perf_counter()
        func()
        t1 = time.perf_counter()
        times.append(t1 - t0)
    avg = sum(times) / len(times)
    mn = min(times)
    mx = max(times)
    speed = None
    if bytes_size is not None:
        mb = bytes_size / (1024 * 1024)
        speed = mb / avg if avg > 0 else 0.0

    cv = None
    if calc_cv and len(times) >= 2 and avg > 0:
        stdev = stats.stdev(times)
        cv = (stdev / avg) * 100.0

    out = f"{label}: avg={avg:.6f}s, min={mn:.6f}s, max={mx:.6f}s"
    if speed is not None:
        out += f", speed={speed:.2f} MB/s"
    if cv is not None:
        out += f", CV={cv:.2f}%"
```

```
    print(out)

    return times


bench("get_random_bytes(4KB)", lambda: get_random_bytes(4096),
bytes_size=4096)

bench("get_random_bytes(1MB)", lambda: get_random_bytes(1024*1024),
bytes_size=1024*1024)

bench("get_random_bytes(10MB)", lambda: get_random_bytes(1024*1024*10),
bytes_size=1024*1024*10)



bench("\nRSA 1024", lambda: RSA.generate(1024), calc_cv=True)

bench("RSA 2048", lambda: RSA.generate(2048), calc_cv=True)

bench("RSA 4096", lambda: RSA.generate(4096), calc_cv=True)
```

2.5 Результати

Результати тестування швидкодії генератора псевдовипадкових послідовностей

```
(.venv) friedreich@friedreich:~/Documents/Labs$ /home/friedreich/Documents/Labs/.venv/bin/
get_random_bytes(4KB): avg=0.000015s, min=0.000014s, max=0.000023s, speed=268.61 MB/s
get_random_bytes(1MB): avg=0.003182s, min=0.002762s, max=0.003971s, speed=314.31 MB/s
get_random_bytes(10MB): avg=0.032477s, min=0.026603s, max=0.038041s, speed=307.91 MB/s
```

Часові характеристики та варіативність генерації ключів RSA

```
RSA 1024: avg=0.091979s, min=0.023878s, max=0.299804s, CV=60.64%
RSA 2048: avg=0.429563s, min=0.057196s, max=1.313770s, CV=66.98%
RSA 4096: avg=5.649395s, min=0.487341s, max=18.551243s, CV=76.39%
```

3 Висновок

Отримані експериментальні результати демонструють, що швидкість генерації псевдовипадкових послідовностей у бібліотеці PyCrypto є високою та стабільною для різних обсягів даних. Для невеликих буферів (4 KB) середня пропускна здатність становить понад 260 MB/s, тоді як для більших обсягів (1 MB і 10 MB) вона зростає до понад 300 MB/s. Це свідчить про ефективне використання системних джерел ентропії та оптимізовані механізми буферизації.

Незначні коливання між мінімальними та максимальними значеннями часу пояснюються впливом планувальника операційної системи, кешуванням та фоновими процесами. Загалом можна зробити висновок, що реалізація генератора ПВП у PyCrypto забезпечує високу продуктивність і є придатною для використання у криптографічних застосуваннях.

З експериментальних даних видно, що час генерації RSA-ключів зростає нелінійно зі збільшенням їх довжини. Зокрема, при переході від 1024-бітного до 2048-бітного ключа середній час збільшився приблизно у 4,7 раза (з 0,092 с до 0,43 с), а при переході до 4096 біт — майже у 10,8 раза (до 4,65 с).

Також спостерігається значний розкид між мінімальними та максимальними значеннями часу виконання. Для ключів довжиною 4096 біт різниця становить від 0,49 с до понад 15,5 с. Це пояснюється ймовірнісною природою алгоритмів пошуку простих чисел: кількість спроб, необхідних для знаходження придатного простого, є випадковою величиною.

Високі значення коефіцієнта варіації (від 60% до 76%) підтверджують значну стохастичність процесу генерації ключів. Така властивість є типовою для ймовірнісних тестів простоти (наприклад, Міллера–Рабіна) та ускладнює проведення атак, заснованих на аналізі часу виконання.

Ключі довжиною 4096 біт забезпечують значно вищий рівень криптографічної стійкості, однак їх генерація може займати істотний час, що обмежує їх використання у сценаріях із частою зміною ключів. Водночас вони є доцільними для довготривалого зберігання секретів.