



МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Кафедра Кібербезпеки

## **ЛАБОРАТОРНА РОБОТА**

з дисципліни «Методи реалізації криптографічних механізмів»

Тема: Аналіз стійкості реалізацій ПВЧ

та генераторів ключів для бібліотеки PyCrypto під Linux платформу

Виконав:

студент 2 курсу

групи ФБ-11мн

Мошонько М.М.

## Мета:

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

## Завдання:

Підгрупа 2В. Аналіз стійкості реалізацій ПВЧ та генераторів ключів для бібліотеки PyCrypto під Linux платформу.

## Хід роботи:

### Система генерації ПВЧ у Linux

У Linux система генерації псевдовипадкових чисел здійснюється через символічні пристрої `/dev/random` і `/dev/urandom`. Дані пристрої мають спільну частину генерації ПВЧ, яка складається із наступних кроків:

1. Із викликів драйверів, пристроїв вводу і інших девайсів збираються події, які потім будуть використані для забілювання початкового значення КСГПВЧ.
2. У зібраних подіях оцінюється їхня "випадковість", і отримане значення додається до лічильника ентропії.
3. Зібрані події використовуються для забілювання початкового значення КСГПВЧ шляхом додавання їх до теперішнього стану.

Наступні кроки будуть розрізнятися для пристроїв `/dev/random` і `/dev/urandom`, тому будуть описані окремо.

### */dev/random*

Для пристрою `/dev/random` наступними кроками генерації ПВЧ будуть:

\*дані кроки можна вважати зацикленими в циклі `while`, який буде виконуватись поки не згенерована достатня кількість чисел.

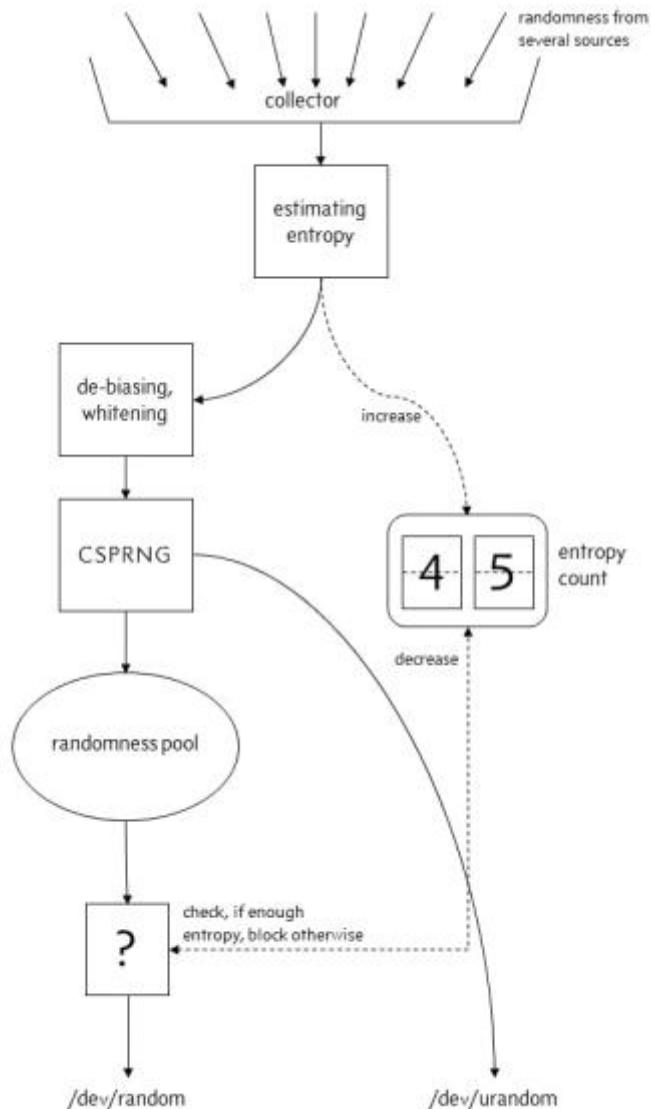
4. Перевірити значення лічильника ентропії.
5. Якщо значення лічильника більше, ніж задане мінімальне значення ентропії, то:
  - 5.1 за допомогою КСГПВЧ згенерувати випадкове число;
  - 5.2 зменшити значення лічильника;
  - 5.3 перевірити умову `while`.
6. В іншому випадку:
  - 6.1 блокувати пристрій поки не буде згенерована достатня кількість ентропії;

6.2 перейти до кроку 5.1;

*/dev/urandom*

Для пристрою */dev/urandom* наступними кроками генерації ПВЧ буде:

4. За допомогою КСГПВЧ згенерувати достатню кількість випадкових чисел.



### Недоліки підходів

Недоліком генерації ПВЧ за допомогою */dev/random* можна вважати блокування системи до поки не буде зібрана достатня кількість ентропії, що в системах, які вимагають миттєвої відповіді є неприпустимим.

Недоліком генерації ПВЧ за допомогою */dev/urandom* можна вважати те, що при генерації не враховується значення лічильника ентропії, але це не зовсім так. Твердження є справедливим лише при умови, що ентропія і початкове значення КСГПВЧ нашої системи відомі.

## Особливості

Ми вважаємо, що ентропія і початкове значення КСГПВЧ нашої системи відомі лише при першому запуску системи, оскільки тоді ці значення рівні 0. При завершенні сеансу роботи машини ентропія і початкове значення зберігаються, тому при наступному сеансі роботи ми починаємо роботу не із нульовим(стандартним) пулом ентропії.

## RSA

За генерацію ключів для алгоритму RSA відповідає функція *generate* у файлі *RSA.py*.

### Опис параметрів входу

На вхід функція приймає три параметри: довжину ключа у бітах *bits*, функцію для генерації випадкових значень *randfunc* і значення параметра відкритої експоненти *e*. Перший параметр є обов'язковий, і має бути не меншим ніж 1024, оскільки в іншому випадку буде кинута помилка з відповідним повідомленням. Другий параметр не є обов'язковим, по замовчуванню в ньому використовується функція *Random.get\_random\_bytes*, яка є обгорткою над функцією *urandom* бібліотеки *os*. Третій параметр також не є обов'язковим, по замовчуванню в ньому стоїть значення 65537, передані значення, в цей параметр, мають бути непарними числами не меншими за 3, оскільки в іншому випадку буде кинута помилка з відповідним повідомленням.

### Опис генерації ключа

Сам процес генерації відбувається в циклі *while* допоки не буде виконані дві умови: згенеровано складений модуль *n* довжини *bits*, і закрита експонента *d* довжина якої в бітах має бути меншою ніж  $\lfloor bits/2 \rfloor$ .

Саме тіло цикла складається з наступних кроків:

1. Встановити бітові довжини *size\_q* і *size\_p*, як  $\lfloor bits/2 \rfloor$  і  $[bits - size_q]$  відповідно

2. Згенерувати випадкове *p*, до якого висунуті наступні вимоги:

- *p* має бути простим
- *p* – 1 взаємнопросте із відкритою експонентною *e*
- *p* більше ніж  $2^{(size\_p)}$

за генерацію простого числа відповідає функція *generate\_probable\_prime*, за дотримання двох останніх вимог відповідає лямбда? функція *filter\_p*.

3. Згенерувати випадкове  $q$ , до якого висунуті наступні вимоги:

- $q$  має бути простим
- $q - 1$  взаємнопросто із відкритою експонентною  $e$
- $q$  більше ніж  $2^{(size\_q)}$
- різниця  $|q - p|$  має бути більша ніж  $2^{\lceil \frac{bits}{2} \rceil - 100}$

за генерацію простого числа відповідає функція *generate\_probable\_prime*, за дотримання трьох останніх вимог відповідає лямбда-функція *filter\_q*.

4. Обрахувати  $n = p \times q$

5. Обрахувати  $lcm = lcm \{p - 1, q - 1\}$

6. Обрахувати  $d = e^{-1} \bmod lcm$

7. Переприсвоїти значення  $q = \max\{p, q\}$ ,  $p = \min\{p, q\}$

8. Обрахувати  $u = p^{-1} \bmod q$

Функція повертає об'єкт *key* ініціалізований значеннями  $n, e, d, p, q, u$ .

## DSA

### Опис параметрів входу

На вхід функція *bits* - довжина ключа або довжина параметру  $p$ , що є простим числом. Повинно бути 1024, 2048 або 3072. *randfunc* (callable) – функція генерації випадкових чисел. Воно приймає одне ціле число  $N$  і повертає рядок випадкових даних завдовжки  $N$  байтів. Якщо не вказано, використовується стандартний генератор з бібліотеки *PyCrypto*. *Domain* – параметри домену DSA  $p, q$  і  $g$  у вигляді списку з 3 цілих чисел. Розміри  $p$  і  $q$  повинні відповідати стандарту FIPS 186-4. Якщо не вказано, параметри створюються заново.

### Опис генерації ключа

1. Генерується просте число  $p$  довжина якої задається вхідним параметром *bits*.
2. Генерується велике просте число  $q < p$ .
3. Генерується секрет  $x \in \{2, \dots, q - 1\}$
4. Обирається відкритий параметр  $g \in \{2, \dots, p\}$
5. Обраховується  $y = g^{(x)} \bmod p$

Сгенеровані відкриті параметри  $(p, q, g, y)$  та закритий параметр  $x$  зберігаються в об'єкті *key*.

## ЕСС

### Опис параметрів входу

В генеруванні ключів в криптосистемі ЕСС (Elliptic Curve Cryptography) на вхід йде лише назва стандарту, яка інкапсулює у собі довжину ключа. У лабораторній роботі було використано "NIST P-256".

### Опис генерації ключа

Пара ключів ЕСС  $d$  і  $Q$  генерується для набору параметрів домену  $(q, F, R, a, b, \{domain\_parameter\_seed\}, G, n, h)$ , яке йде на вхід генерації ключа, де  $n$  – просте число,  $G$  - точка еліптичної кривої. Еліптичні криві визначені в стандарті FIPS 186-4.

1. Обраховуємо  $N = len(n)$  і робимо перевірку безпеки цього параметра. Якщо задовольняє стандарту, то переходимо на наступний крок. Інакше завершити алгоритм.
2. Генеруємо  $c \in \{2, \dots, n - 1\}$
3. Обраховуємо  $d = (c \bmod (n - 1)) + 1$
4. Обраховуємо  $Q = dG$

На виході пара ключів  $(d, Q)$  інкапсулюються у об'єкт key.

## ВИСНОВКИ

У цій роботі було досліджено реалізацію алгоритмів генерації ключів гібридних криптосистем RSA, DSA, ЕСС у бібліотеці PyCrypto під Linux платформу. Було описано основні кроки, які виконуються при генерації ключів. Також було згенеровано пари ключів, які записані в файли з розширенням .pem.