Національний технічний університет України
«Київський політехнічний інститут»

Фізико технічний інститут

Кафедра математичних методів захисту інформації

МЕТОДИ РЕАЛІЗАЦІЇ КРИПТОГРАФІЧНИХ МЕХАНІЗМІВ КОМП'ЮТЕРНИЙ ПРАКТИКУМ №2

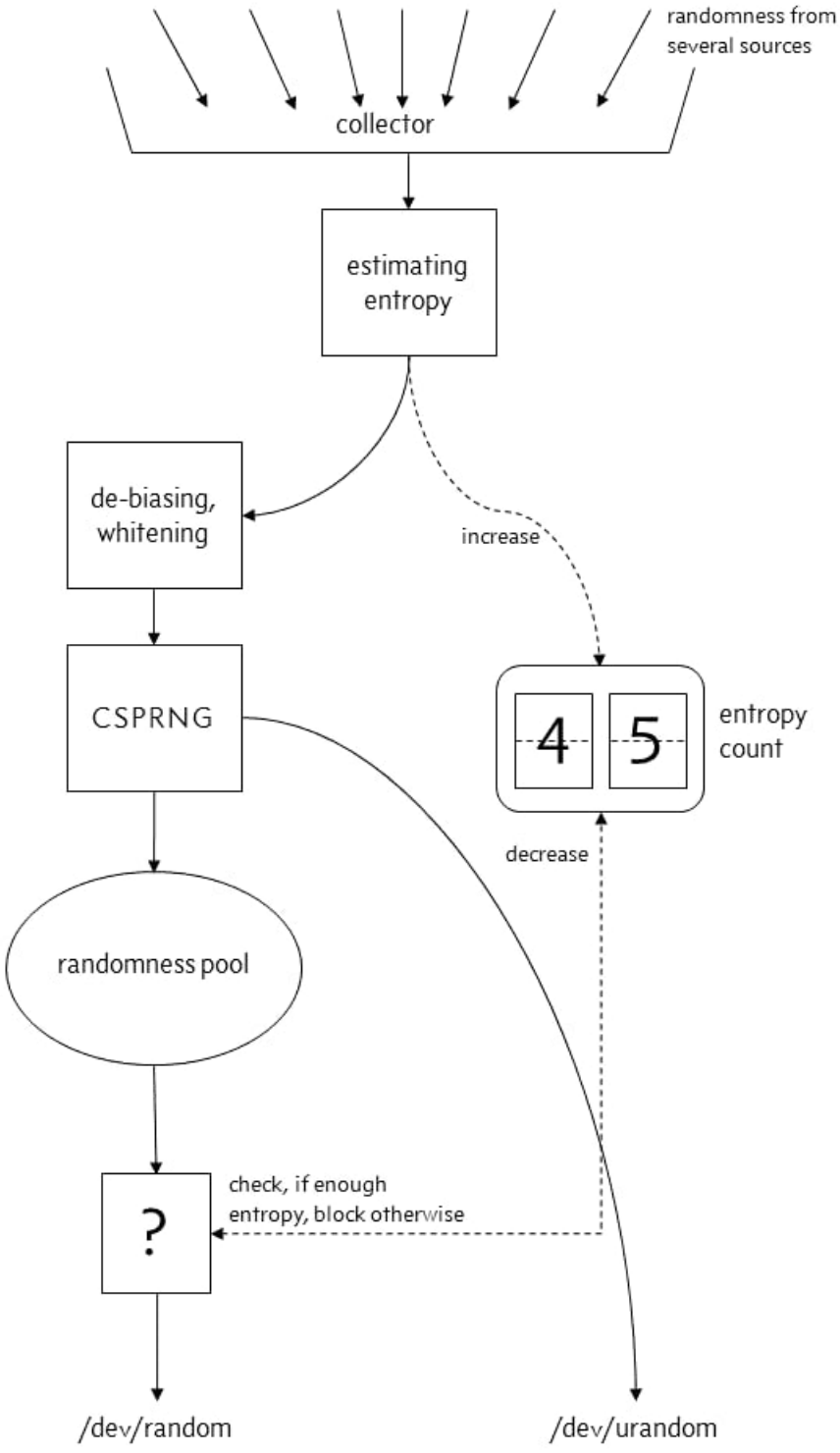
Реалізація алгоритмів генерації ключів гібридних криптосистем

Виконали:
студенти групи ФІ-73
Драга Владислав
Чіхладзе Вахтанг

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



(Усі наступні викладки стосуються Linux систем починаючи із версії ядра 4.8)

У Linux система генерації псевдовипадкових чисел здійснюється через символьні пристрої **/dev/random** і **/dev/urandom**. Дані пристрої мають спільну частину генерації ПВЧ, яка складаєтьс 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 подій оцінюється їхня "випадковість", і отримане значення додається до лічильник ентропії.
3. Зібрані події використовуються для забілювання початкового значення КСГПВЧ шляхом дования 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 [while(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 будуть:

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 умові, що ентропія і початкове значення КСГПВЧ нашої системи відомі.

Особливості

Ми вважаємо, що що ентропія і початкове значення КСГПВЧ нашої системи відомі лише при першому запуску системи, оскільки тоді ці значення рівні 0. При завершені 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 повідомлення. 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 не парними числами не менші за 3, оскільки в іншому випадку буде кинута помилка з відповідним повідомлення.

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 $\left\lfloor \frac{bits}{2} \right\rfloor$.

саме тіло цикла складається з наступних кроків:

1. встановити бітові довжини $size\_q$ і $size\_p$, як $\left\lfloor \frac{bits}{2} \right\rfloor$ і $bits - size\_q$ відповідно
2. Згенерувати випадкове p, до якого висунуті наступні вимоги:
   - p має бути простим
   - $p - 1$ взаємнопросте із відкритою експонентою e
   - p більше ніж 2^{size_p}

   за генерацію простого числа відповідає функція $generate\_probable\_prime$, за дотримання двох останіх вимог відповідає лямбла? функція $filter\_p$.
3. Згенерувати випадкове q, до якого висунуті наступні вимоги:
   - q має бути простим
   - $q - 1$ взаємнопросте із відкритою експонентою e
   - q більше ніж 2^{size_q}
   - різниця $|q - p|$ має бути більша ніж $2^{\left\lfloor \frac{bits}{2} \right\rfloor - 100}$

   за генерацію простого числа відповідає функція $generate\_probable\_prime$, за дотримання трьох останіх вимог відповідає лямбда-функція $filter\_q$.
4. обрахувати $n = p \times q$
5. обрахувати $lcm = \text{lcm}\{p - 1, q - 1\}$
6. обрахувати $d = e^{-1} \mod lcm$
7. переприсвоїти значення $q = \max\{p, q\}$, $p = \min\{p, q\}$
8. обрахувати $u = p^{-1} \mod q$

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 $x \in \{2, .., q-1\}$
4. Обирається відкритий параметр $g \in \{2, ..., p\}$
5. Обраховується $y = g^x mod p$

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

ECC

опис параметрів входу

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

опис генерації ключа

Пара ключів ECC d і Q генерується для набору параметрів домену $(q, FR, a, b, \{domain\_parameter\_seed\}, G, n, h)$, 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 задовільняє стандарту, то переходимо на наступний крок. Інкакше завершити алгоритм.
2. Генеруємо $c \in \{2, ..., n-1\}$
3. Обраховуємо $d = (c mod(n-1)) + 1$
4. Обраховуємо $Q = dG$

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

Висновок

Отже, було досліджено реалізацію алгоритмів генерації ключів гібридних криптосистем RSA, DSA, ECC. Було описано основні кроки, які виконуються при генерації ключів. Також було сгенеровано пари ключі, які записані в файли з розширенням pem.

Використані джерела

[1] PyCrypto RSA documentation

[2] PyCrypto DSA documentation

[3] PyCrypto ECC documentation

[4] FIPS 186-4