Біткоїн та криптовалютні технології
Лекція 2: Основи криптографії 1/2

Юрій Жикін

28 вересня, 2022

Вступ до криптографії 1/2

- **Криптографія** (з дав. грецької, “прихований, таємний” і “писати”) - теорія і практика технік *безпечної комунікації* в умовах присутності *присутності* ***зовнішнього спостерігача***, *якого називають ворогом (супротивником)*.
- **Сучасна криптографія** майже повністю базується на *математичній теорії і комп'ютерних науках*.
- **Сучасні криптографічні алгоритми** базуються на припущення про обчислювальну складність деяких задач.

Вступ до криптографії 2/2

- Сучасна криптографія поділяється на дві категорії:
  - **симетрична криптографія** - обидві сторони мають спільний секретний ключ, який використовується для шифрування та дешифрування,
  - **несиметрична криптографія (криптографія з публічним, або відкритим, ключем)** - ключ складається з публічної, або відкритої, та приватної, або закритою, частин; публічний ключ використовується для шифрування, а приватний ключ - для дешифрування.
- Криптографічні протоколи виконують для двох основних цілей:
  - **приховування інформації** (шифрування/дешифрування),
  - **забезпечення цілісності інформації** (цифрові підписи і їх верифікації).

Симетрична криптографія

- Єдиний вид криптографії до 1976 року.
- Обидві сторони мають спільний таємний ключ, який використовується для шифрування та дешифрування.
- Алгоритми симетричного шифрування дуже швидкі (наприклад *AES*, *Salsa20*, *ChaCha*).
- Більшість популярних алгоритмів симетричного шифрування реалізовано “в залізі” (наприклад *AES* та AES-NI інструкції в процесорах архітектури x86).
- Схема досконалого шифрування:

$$E = M \oplus K,$$
$$D = E \oplus K,$$
$$|M| == |K|$$

Недоліки симетричної криптографії

- Сторони повинні заздалегідь домовитись про таємний ключ через безпечний канал зв'язку - проблема "курки та яйця".
- Симетричність витоку - якщо стається витік ключа у однієї з сторін, обидві сторони скомпрометовано.
- Якщо декілька сторін мають один спільний ключ, симетрія витоку зачіпає всіх учасників.
- Якщо кожен з учасників має окремий ключ для кожного іншого учасника в системі, виникає проблема зберігання ключів.

Асиметрична криптографія

- Фундаментальне відкриття **Вітфілда Діффі**, **Мартіна Гелмана** та **Ральфа Меркла** у 1976 році.
- Повідомлення шифруються публічним (відкритим) ключем, але можуть бути дешифровані лише приватним (закритим) ключем.
- Кожна сторона відповідальна лише за зберігання власного приватного ключа - публічний ключ може бути обчислений з приватного ключа, і може бути надісланий через будь-який незахищений канал зв'язку.

Ймовірнійсть та випадкові великі числа 1/4

- В більшості сучасних криптографічних систем безпека ключа базується на ймовірності вгадування дуже великого числа.
- Для того, щоб зробити вгадування ключа якомога складнішим, числа повинні бути справді випадковими: ймовірнійсть того, що кожен біт в числі є 1 чи 0, повинна бути рівною 0.5.
- Ймовірність вгадування дійсно випадкового числа N становить $1/N$.
- Приблизна оцінка кількості атомів у Всесвіті становить $10^{78} \simeq 2^{259}$, отже вгадування 256-бітного ключа можна порівняти з вгадуванням конкретного атома у Всесвіті.

Ймовірність та випадкові великі числа 2/4

- В той же час, великі числа, що використовуються як криптографічні ключі, можуть бути компактно представлені у 16-ковому чи 64-ковому кодуванні:

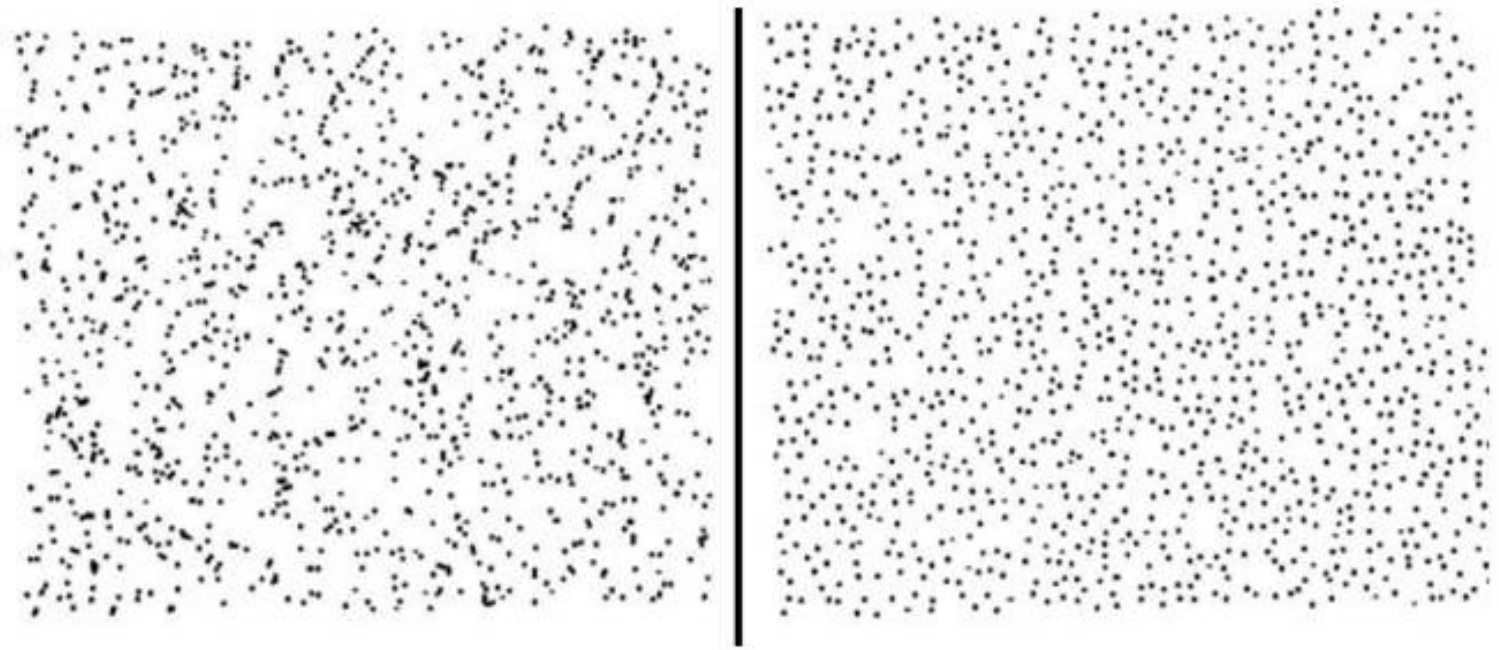
```
import secrets
import base64
bits = secrets.randbits(256)
# 46518555179467323509970270980993648640987722172281263586388328188640792550961
bits_binary = '{0:b}'.format(bits)
# 1100110110110001001000110110101111011010111111010100011110010100000100010010\
# 1111001101010011111101011111001001110001011101010111000110010101110010110000001\
# 1110101101010100000100010000011111101111100110000001100111001001111110101101000\
# 10010000111100011000 1
bits_hex = hex(bits)
# 0x66d891b5ed7f51e5044be6a7ebe4e2eae32b960f5aa0883f7cc0ce4fd6921e31
bits_base64 = base64.b64encode(bits.to_bytes(32, 'little'))
# MR6S1k/OwHw/iKBaD5Yr4+ri5Oun5ksE5VF/7bWR2GY=
```

Ймовірність та випадкові великі числа 3/4

- Комп'ютери детерміністичні, тому генерація дійсно випадкових чисел - це нетривіальна задача.
- В більшості випадків, найбільш безпечний спосіб отримати випадкове число в Unix-подібних операційних системах - це прочитати послідовність байтів з віртуального пристрою `/dev/random` чи `/dev/urandom`.
- Існують спеціальні пристрої, які генерують дійсно випадкові числа на основі ентропії середовища; якщо це можливо, варто віддавати їм перевагу для генерації криптографічних ключів.

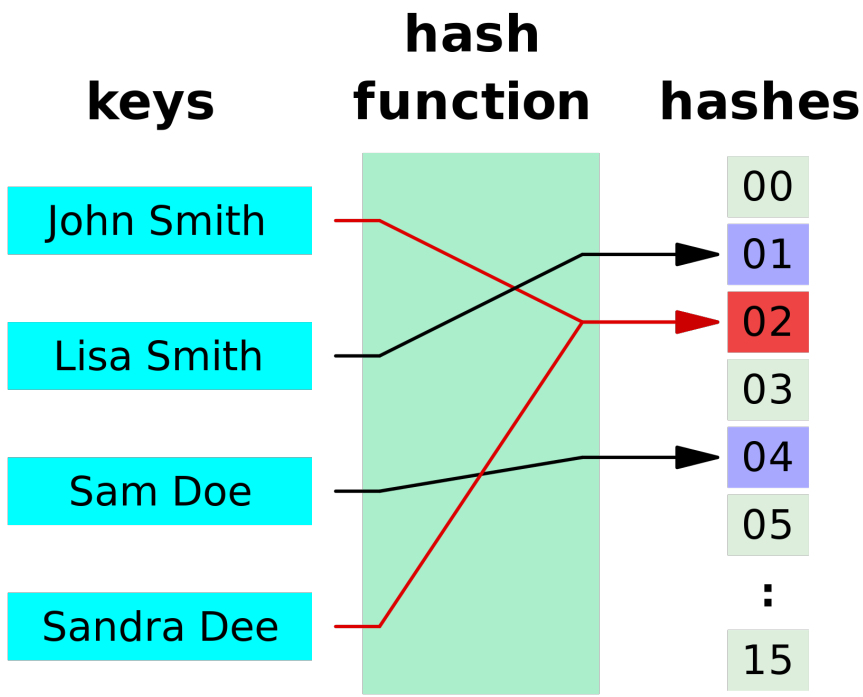
Ймовірнійсть та випадкові великі числа 4/4

- Люди дуже погано вміють сприймають дісно випадкові величини через те, що вони мають еволюційно гіпертрофовану здатність знаходити закономірності.



Хеш-функції

- **Хеш-функція** - це функція, що перетворює значення довільного розміру у значення фіксованого розміру.
- Хеш-функції використовуються для адресації у структурах даних (хеш-таблиці), ймовірнісних фільтрах (фільтри Блума), тощо.



Криптографічні хеш-функції

- **Криптографічні хеш-функції** - це *хеш-функції*, що перетворюють довільні дані у послідовності бітів фіксованого розміру і є **односторонніми функціями**, тобто функціями, які практично неможливо інвертувати:

$$h = H(m) \text{ - ефективно,}$$
$$m = H^{-1}(h) \text{ - } \textbf{дуже} \text{ неефективно}$$

- **Найбільш ефективними** способом інвертувати криптографічну хеш-функцію, тобто знайти вхідне повідомлення m такі, що перетворюються у хеш h даною хеш-функцією, - це пошук "грубою силою", або повний перебір, - обираємо випадкове повідомлення m_i і перевіряємо, чи $H(m_i) = h$.
- Фундаментальний інструмент сучасної криптографії.

Властивості криптографічних хеш-функцій 1/2

- Основні властивості **надійних** криптографічних хеш-функцій:
  - **детермінізм** - той же вхід завжди дає той же вихід (хеш),
  - **ефективність** - хеш даного повідомлення можна обчислити дуже швидко,
  - **дифузія, "лавинна властивість** - зміна одного біта у повідомленні m призводить до зміни кожного біта у h з ймовірністю 0.5,
  - **стійкість до пошуку прообразу** - маючи хеш h, повинно бути складно знайти будь-яке повідомлення m таке, що $h = H(m)$,
  - **стійкість до пошуку другого прообразу** - маючи повідомлення m_1, повинно бути складно знайти будь-яке повідомлення m_2 таке, що $H(m_1) = H(m_2)$,
  - **стійкість до колізій** - повинно бути складно знайти будь-які два повідомлення m_1 і m_2 такі, що $H(m_1) = H(m_2)$.

Властивості криптографічних хеш-функцій 2/2

- Додаткові властивості **надійних** криптографічних хеш-функцій:
  - **стійкість до збільшення довжини** - маючи $h = H(m)$ і $len(m)$, повинно бути складно знайти $h' = H(m||m')$,
  - **сильна стійкість до колізій** - стійкість до атак через **парадокс днів народження**. resistance.

Використання криптографічних хеш-функцій 1/2

- **Коди аутентифікації повідомлень (MAC)** - хеш якогось повідомлення, поєднаного з якимось ключем, дозволяє перевірити цілісність даного повідомлення.
- **Цифрові підписи** - підписування хеша повідомлення є значно більш швидкою операцією, ніж підписування цілого повідомлення.
- **Перевірка паролів** - зберігання паролів у відкритому вигляді призводить до серйозних порушень безпеки, якщо база даних з паролями "витікає" назовні; зберігання хешів паролів дозволяє цього уникнути.
- **Сильні перевірки цілісності даних (чек-суми)** - використовуються замість звичайних некриптографічних хеш-функцій, коли необхідні більш надійні гарантії.
- Приклади: **SHA-2** (**SHA-256**, **SHA-512**), **RIPEMD-160**, **SHA-3**.

Використання криптографічних хеш-функцій 2/2

- **Докази виконаної роботи (Proof-of-Work)** - основа сучасних криптовалютних технологій.
- *Hashcash* - оригінальна ідея, запропонована Адамом Беком у 1997 році як засіб боротьби з спамом в системах електронної пошти та атаками відмови в обслуговуванні.
- Основна ідея **PoW**:
  - для деякого повідомлення m, виконуємо **пошук шляхом "грубої сили"** значенн r такого, що $h = H(m, r)$ задовільняє певний критерій, наприклад

$$h < h_{target}$$

  - критерій пошуку може бути обрано таким, що пошук значення r за допомогою якоїсь обчислювальної потужності займатиме в середньому певну кількість часу.
- Обчислення **доказу виконаної роботи** потребує конкретної кількості енергії, яку можна оцінити, і ця кількість може бути достатно великою, щоб обчислення нового доказу було дорогим.

Корисні ресурси

- Dan Boneh's Cryptography I course from Stanford University - https://www.coursera.org/learn/crypto.
- Serious Cryptography: A Practical Introduction to Modern Encryption - Jean-Philippe Aumasson.
- 8 sets of cryptography problems that introduce various real-life cryptography systems and show practical attacks on them - https://cryptopals.com.

Кінець

Дякую за увагу!