Хешування рядку. Хеш-функції

Йович Анастасія

29.05.2020

Хеш-функція

$$h\colon\ A\ \rightarrow\ B$$

- перетворення даних довільної довжини в фіксовану
- необоротна функція (по хешу неможливо відновити початкову інформацію)
- ймовірність колізій мала
- не має вимагати значних обчислювальних ресурсів

криптографічна хеш-функція

- при наймешних змінах має суттєво змінюватися (хеш Password і password має суттєво відрізнятися)

- якщо є два набори інформації $m, k, m \neq k$, при цьому m - відомий, $\mathrm{h(m)}$ - відомо, то знайти k малоймовірно

Як придумати хорошу хеш-функцію?

Що точно НЕ робити:

- Якщо хеш-функція має суммування, то “stop” і “post” дадуть однакові значення
- Не можна обмежуватися лише першими n значеннями (навіть при хешування рядка з 300 символів, який треба перевести в 40 символів), інакше “Дім” і “Діма” дадуть однакові значення
- Множити на числа, що мають більше 2х дільників $(21=1*3*7) \rightarrow$ краще обирати прості числа

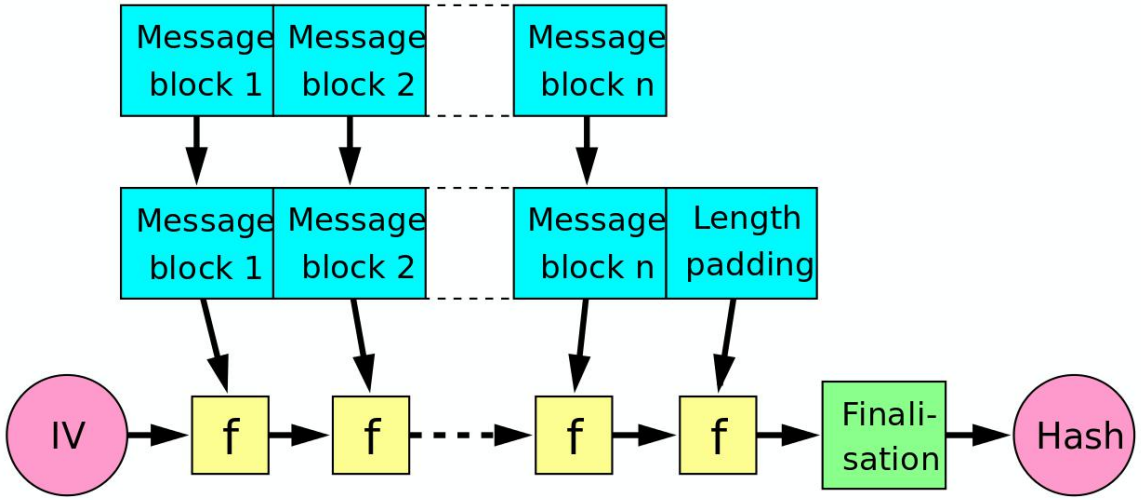
Поліноміальні хеші

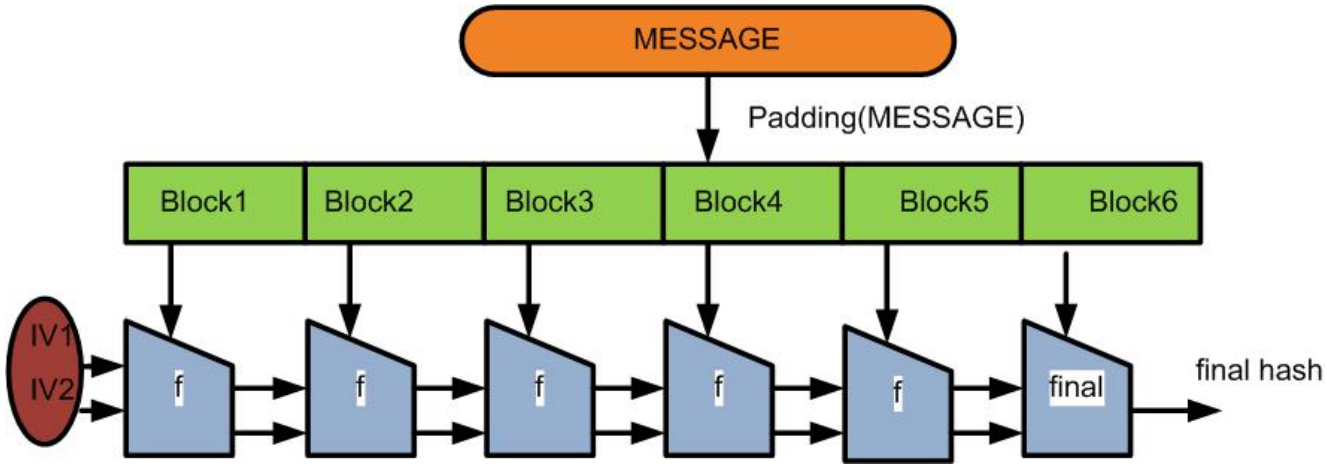
- `h(S) = S[0] + S[1] * P + S[2] * P^2 + S[3] * P^3 + ... + S[N] * P^N`
- P - просте число, найчастіше обирають найближче просте число до кількості букв в алфавіті (29, 31, 33). Коли літери можуть мати різний регістр, то 53, 61, 63
- Легко рахувати хеш підрядків, маючи хеш всього рядка
- Можна легко виконувати такі операції:
  - рахувати хеш від конкретинації a+b
  - видалити префікс a у a+b
  - видалити суфікс b у a+b

Алгоритми хешування (стандарти)

- SHA-1
- SHA-256
- MD5
- контрольна сума
- методи порівняння штрих-кодів

Структура Меркла — Дамгора



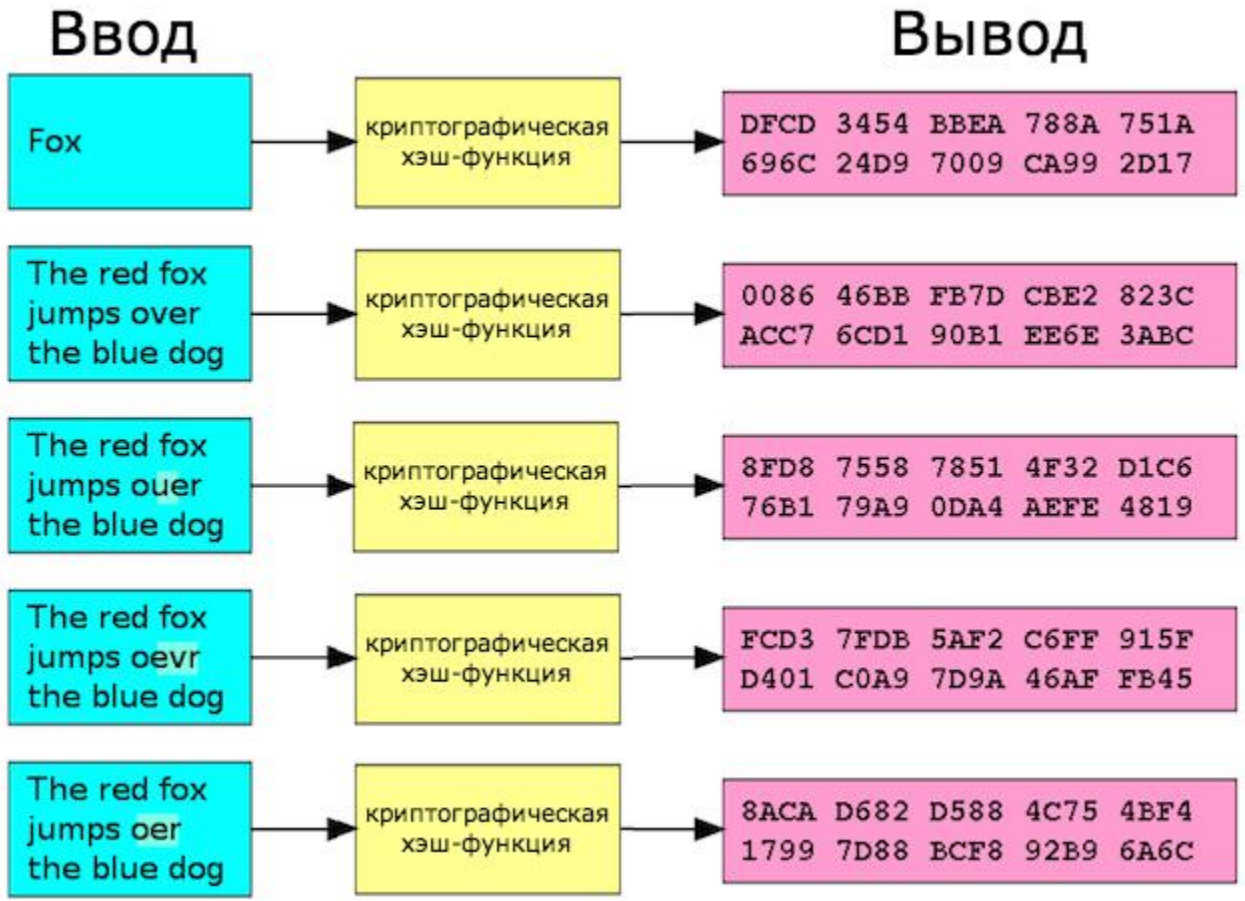


f – функція стиснення
Якщо вона стійка до колізій, то і вся хеш-функція стійка до колізій

Кінцевий результат пропускають через фукнцію фіналізації (англ. finalisation function) для більшої найдійності і стиснення

Алгоритми, що реалізують структуру Меркла-Дамгора – SHA-1, SHA-2, MD5

SHA-1 (security hash algorithm)



Функція f має такі операції:

конкатенація,
+ — сума,
and — побітове «і»,
xor — виключне «або»,
shr (shift right) — логічний здвиг вправо,
rotr (rotate right) — циклічний здвиг вправо.

std::hash

- Ex.1
- Ex.2

Notes

The actual hash functions are implementation-dependent and are not required to fulfill any other quality criteria except those specified above. Notably, some implementations use trivial (identity) hash functions which map an integer to itself. In other words, these hash functions are designed to work with unordered associative containers, but not as cryptographic hashes, for example.

Hash functions are only required to produce the same result for the same input within a single execution of a program; this allows salted hashes that prevent collision denial-of-service attacks. (since C++14)

There is no specialization for C strings. `std::hash<const char*>` produces a hash of the value of the pointer (the memory address), it does not examine the contents of any character array.

Застосування

- паролі, персональні данні
- криптовалюта
- псевдогенератори випадкових чисел
- контроль порушення авторських прав
- контроль завантажень файлів (контрольні суми)
- відслідковування вірусів по хешам їх відомих компонентів

Генератори псевдовипадкових чисел

	Источник энтропии	ГПСЧ	Достоинства	Недостатки
/dev/random в UNIX/Linux	Счётчик тактов процессора, однако собирается только во время аппаратных прерываний	РСЛОС, с хешированием выхода через SHA-1	Есть во всех Unix, надёжный источник энтропии	Очень долго «нагревается», может надолго «застревать», либо работает как ГПСЧ (*/dev/urandom*)
***Yarrow* от Брюса Шнайера**[12]	Традиционные методы	AES-256 и SHA-1 маленького внутреннего состояния	Гибкий криптостойкий дизайн	Медленный
Microsoft CryptoAPI	Текущее время, размер жёсткого диска, размер свободной памяти, номер процесса и NETBIOS-имя компьютера	MD5-хеш внутреннего состояния размером в 128 бит	Встроен в Windows, не «застревает»	Сильно зависит от используемого криптопровайдера (CSP).
Java SecureRandom	Взаимодействие между потоками	SHA-1-хеш внутреннего состояния (1024 бит)	Большое внутреннее состояние	Медленный сбор энтропии
RdRand от intel[13]	Шумы токов	Построение ПСЧ на основе «случайного» битового считывания значений от токов[13]	Очень быстр, не «застревает»	Оригинальная разработка, свойства приведены только по утверждению разработчиков.

Дякую за увагу!