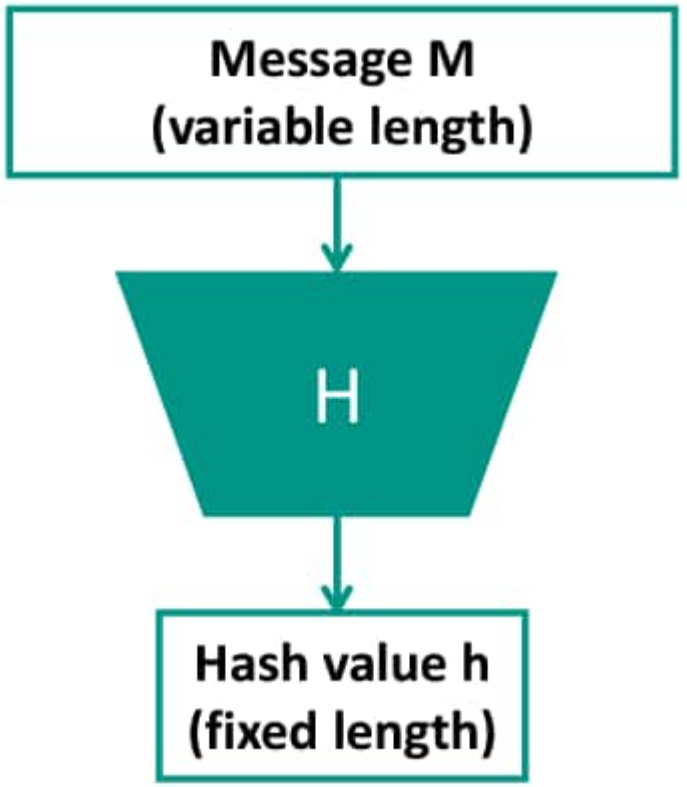
Математична криптологія

Криптографічні хеш-функції

Хешування

- **Хешування** («гешування», англ. hashing) - перетворення за детермінованим алгоритмом вхідного масиву *M* даних довільної довжини у вихідний бітовий рядок фіксованої довжини *H*(*M*). Такі перетворення також називаються хеш-функціями або функціями згортки (стискання), а їх результати називають хешем, хеш-кодом або зведенням повідомлення (англ. message digest). Якщо два рядки хеш-коди різні, рядки гарантовано різняться, якщо однакові – рядки імовірно збігаються.
- Вибір тієї чи іншої хеш-функції визначається специфікою задачі, що розв'язується. Найпростішими прикладами хеш-функцій можуть бути контрольна сума або CRC .
- У загальному випадку однозначної відповідності між вихідними даними та хеш-кодом немає. Тому існує безліч масивів даних, що дають однакові хеш-коди - так звані колізії. Імовірність виникнення колізій грає важливу роль оцінці «якості» хеш-функцій.



Застосування хешування

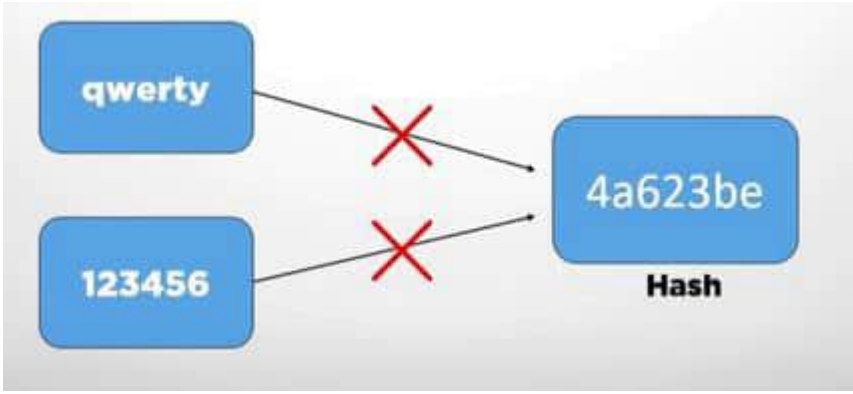
Загалом застосування можна описати, як перевірка деякої інформації на ідентичність оригіналу, без використання оригіналу. Для звіряння використовується хеш-значення інформації, що перевіряється. Розрізняють два основні напрямки цього застосування:

- **Перевірка на наявність помилок:** наприклад, контрольна сума може бути передана каналом зв'язку разом з основним текстом. На приймальному кінці контрольна сума може бути розрахована заново і її можна порівняти з переданим значенням.
- **Перевірка паролю:** У більшості випадків парольні фрази не зберігаються у відкритому вигляді, зберігаються лише їх хеш-значення. Зберігати парольні фрази недоцільно, тому що у разі несанкціонованого доступу до файлу з фразами зловмисник дізнається всі паролі і відразу зможе ними скористатися, а при зберіганні хеш-значень він дізнається лише хеш-значення, які не оборотні у вихідні дані, в даному випадку парольну фразу. У ході процедури аутентифікації обчислюється хеш-значення введеного паролю і порівнюється зі збереженим.
- **Прискорення пошуку даних:** Наприклад, при записі текстових полів у базі даних може розраховуватися їхній хеш-код і дані можуть поміщатися в розділ, що відповідає цьому хеш-коду. Тоді при пошуку даних треба буде спочатку обчислити хеш-код тексту і відразу стане відомо, у якому розділі їх треба шукати, тобто шукати треба буде не по всій базі, а лише по одному її розділу (це прискорює пошук).

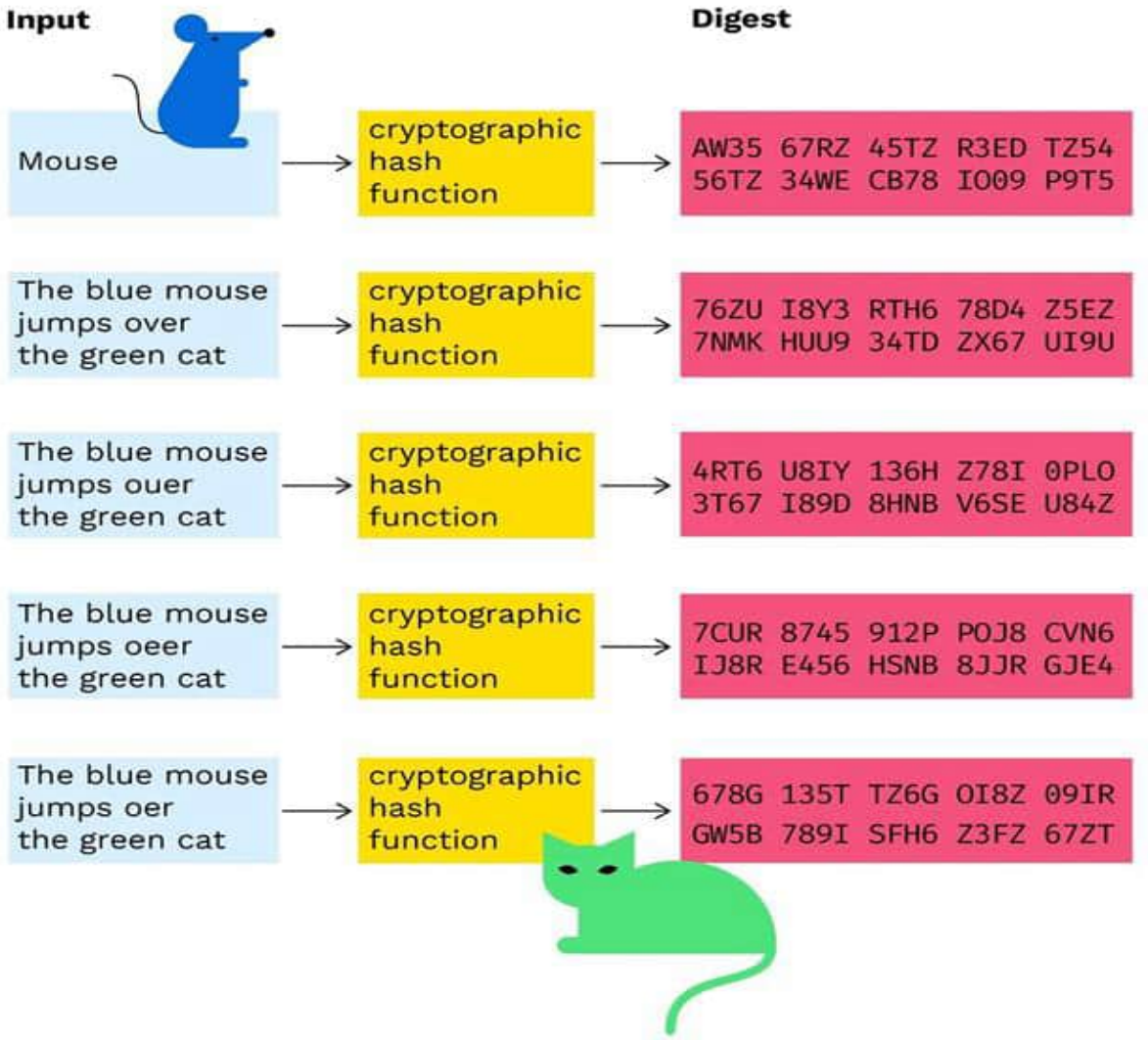
Криптографічні хеш-функції

Серед безлічі існуючих хеш-функцій прийнято виділяти криптографічно-стійкі, які застосовуються в криптографії. Криптографічні хеш функції повинні володіти наступними властивостями:

- **Детермінованість** – для однакових повідомлень *M* функція має повертати однакові хеш-значення *hash* = *H*(*M*);
- **Односторонність** –за значенням *hash* неможливо відновити *M*;
- **Стійкість до колізій першого роду:** для заданого повідомлення *M* має бути практично неможливо підібрати друге повідомлення *N*, для якого *H*(*N*)=*H*(*M*);
- **Стійкість до колізій другого роду:** має бути практично неможливо підібрати пару повідомлень (*M1*, *M2*) які мають однаковий хеш;
- Висока швидкість роботи.



Основні властивості хеш-функцій



- **Наявність лавинного ефекту:** будь-які, навіть незначні, зміни у повідомленні M призводять до значних змін у хеш-значенні *h*;

Принцип роботи криптографічних хеш-функцій

- Повідомлення M має бути представлене у двійковій формі і розбите на окремі блоки M_i довжиною n біт кожний;
- Більшість хеш-функцій мають вигляд: $h_i = H(M_i, h_{i-1})$, де M_i–черговий блок повідомлення M; h_{i-1}–хеш-значення усіх попередніх блоків M (має довжину також n біт);
- При обчисленні хеш-значення для першого блоку M_1 використовується деяке початкове хеш-значення h_0 , яке можна вибрати випадковим IV або фіксованим (наприклад, $h_0 = 0$ – у найпростішому випадку);
- Хеш-значення, обчислене при використанні останнього блоку повідомлення, вважається хеш-значенням усього повідомлення M

Криптографічні хеш-функції

Хеш-функція	Рік	Розробники	Довжина блоку	Довжина дайджесту	Кількість раундів
MD5	1992	Ronald Rivest	512	128	64
RIPEMD	1992	The RIPE Consortium	512	128	48
SHA-1	1995	NSA	512	160	80
SHA-256	2002	NSA	512	256	64
SHA-512	2002	NSA	1024	512	80
Whirlpool	2004	Vincent Rijmen, Paulo Barreto	512	512	10
BLAKE-256	2008	Jean-Philippe Aumasson, Luca Henzen, Willi Meier, Raphael C.-W. Phan	512	256	14
Купина	2014	ПАТ «Інститут інформаційних технологій»	512	від 8 до 512 біт	10 або 14

Алгоритм стійкого хешування *SHA-1*

Алгоритм Secure Hash Algorithm (*SHA* – алгоритм стійкого хешування) є частиною стандарту SHS (Secure Hash Standard), прийнятого 1993 року Національним інститутом стандартів і технологій США (NIST)

- Робота алгоритму *SHA*-1 розпочинається з того, що вхідна послідовність ділиться на блоки по 512 біт;
- Далі ініціалізовується п’ять 32-розрядних змінних: A = 67452301; B = EFCDAB89; C = 98BADCFE; D = 10325476; E = C3D2E1F0. при цьому стартовий вектор хешування (синхронадсилання) є результат конкатенації цих змінних, тобто SHA_0 = (A, B, C, D, E).
- На вхід *i*-того циклу перетворення SHA_i надходить *i*-тий блок інформаційної послідовності та результат роботи попереднього циклу SHA_{i-1}, тобто $SHA_i = H(m_i, SHA_{i-1})$.
- Кожна операція становить собою набір нелінійних функцій від трьох змінних (B, C і D) та операцій циклічного зсуву й підсумовування. Ці функції мають таку форму:

1-й раунд	$f_1(B, C, D) = (B \wedge C) \vee (\overline{B} \wedge D)$	за $0 \le t \le 19$;
2-й раунд	$f_2(B, C, D) = B \oplus C \oplus D$	за $20 \le t \le 39$;
3-й раунд	$f_3(B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$	за $40 \le t \le 59$;
4-й раунд	$f_4(B, C, D) = B \oplus C \oplus D$	за $60 \le t \le 79$.

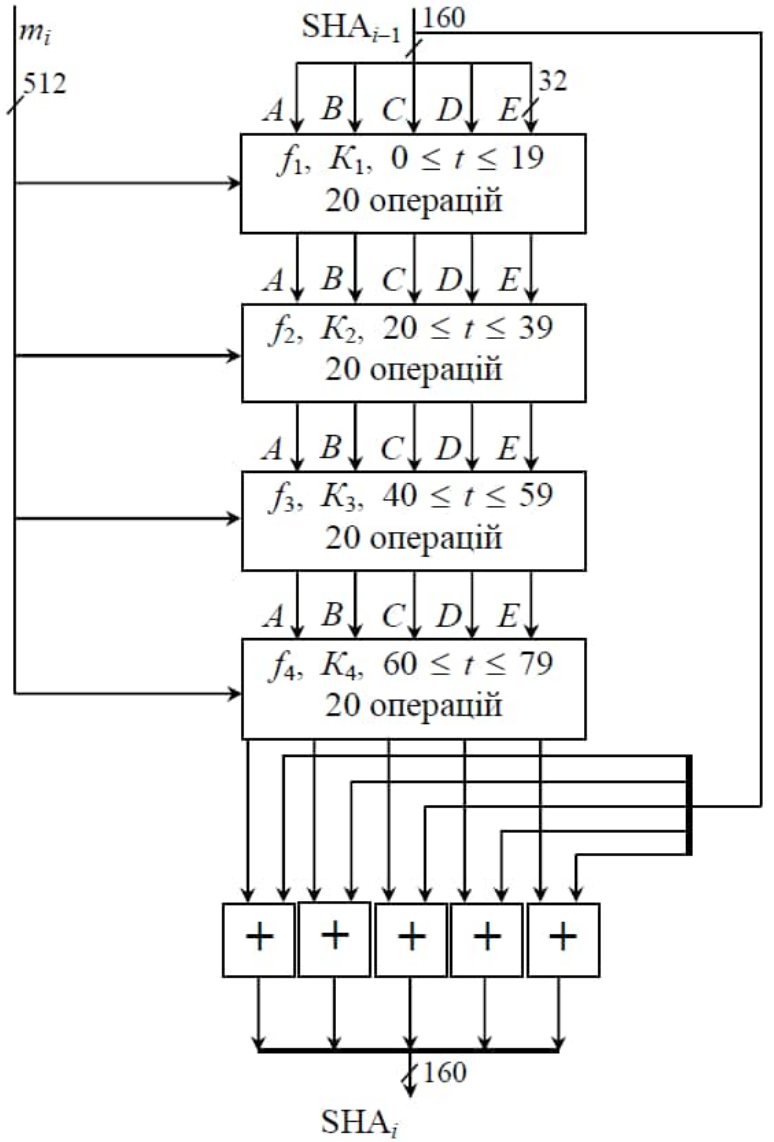
Алгоритм стійкого хешування *SHA*

$K_1 = 5A827999h = [2^{30}\sqrt{2}]$ при $0 \le t \le 19$;

$K_2 = 6ED9EBA1h = [2^{30}\sqrt{3}]$ при $20 \le t \le 39$;

$K_3 = 8F1BBCDCh = [2^{30}\sqrt{5}]$ при $40 \le t \le 59$;

$K_4 = CA62ClD6h = [2^{30}\sqrt{10}]$ при $60 \le t \le 79$,



Алгоритм стійкого хешування *SHA-1*

Розглянемо логіку будь-якої з 80-ти операцій опрацювання одного 512-бітового блока. Кожна операція має форму (рис. 3.2)

$$A, B, C, D, E \leftarrow (E + f_t(B, C, D) + \mathrm{Rol}^5(A) + W_t + K_t), A, \mathrm{Rol}^{30}(B), C, D,$$

де $\mathrm{Rol}^5(A)$ – циклічний зсув ліворуч 32-бітового аргументу A на 5 біт;

W_t – 32-бітове слово, вилучене з поточного 512-бітового блока введення;

$\mathrm{Rol}^{30}(B)$ – циклічний зсув ліворуч 32-бітового аргументу B на 30 біт;

$+$ – додавання за модулем 2^{32}.

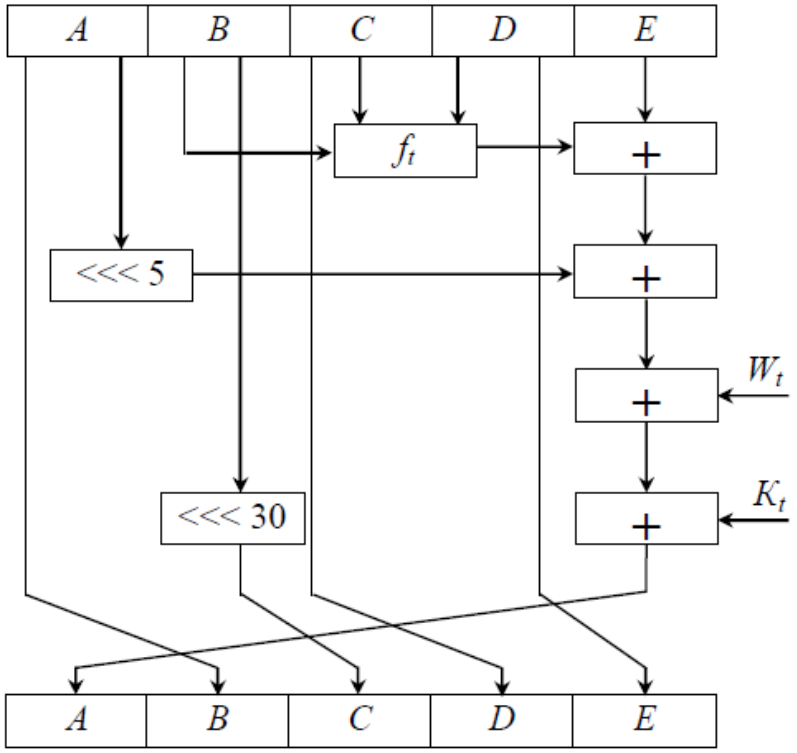


Рисунок 3.2 – Схема однієї операції SHA-1

Парадокс днів народження

- У теорії ймовірностей парадокс днів народження оцінює ймовірність того, що у випадково вибраній групі людей збігатимуться дні народження в якоїсь пари.
- Позначимо $P(A')$ ймовірність того, що в групі немає двох людей з однаковим днем народження та $P(A)$ ймовірність збігу щонайменше двох днів народження, тоді $P(A) = 1 - P(A')$

$$1 \times \frac{364}{365} \times \frac{363}{365} \times \dots \times \frac{365-(N-1)}{365}$$

$$\frac{1}{365^N} \prod_{i=0}^{N-1} (365 - i)$$

Парадокс днів народження

- Імовірність збігу щонайменше двох днів народжень: $P(A) = 1 - P(A')$

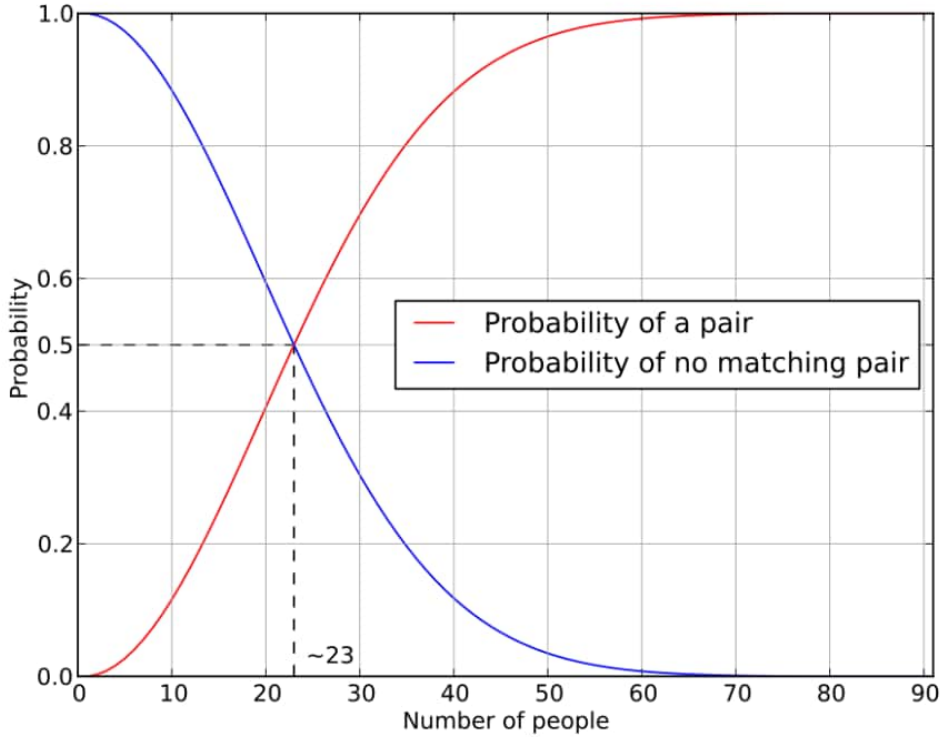
$$1 - [1 \times \frac{364}{365} \times \frac{363}{365} \times \dots \times \frac{365 - (N-1)}{365}]$$

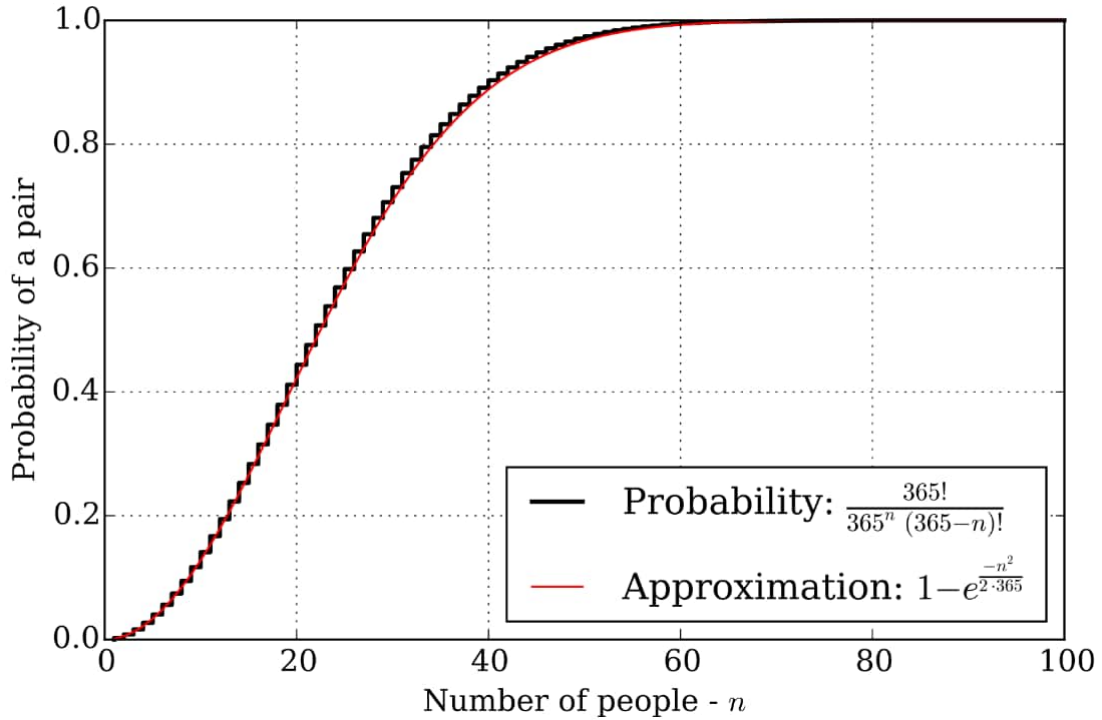
$$1 - [\frac{1}{365^N} \prod_{i=0}^{N-1} (365 - i)]$$

$$0.5 = 1 - [\frac{1}{365^N} \prod_{i=0}^{N-1} (365 - i)]$$

Парадокс днів народження

- В групах кількістю не менших 23 випадково вибраних людей, ймовірність збігу днів народження в якоїсь пари становить більше 50 %. Такий результат суперечить інтуїтивній уяві більшості.
- Для 57 людей, ймовірність становить більше ніж 99 %, і досягає 100 % коли, ігноруючи високосний рік, кількість людей у групі становить 366 (принцип Діріхле). Такий розподіл ймовірностей призвів до широко відомої криптографічної атаки відомої як атака «днів народження».





Атака «днів народження»

- Атака «днів народження» — це різновид криптографічної атаки, яка використовує математичне підґрунтя парадоксу днів народження в теорії ймовірностей. Цю атаку можна використати для втручання в зв'язок між двома або більше учасниками. Атака покладається на високу ймовірність знаходження колізій між випадковими спробами.
- Нехай $Q(H)$ буде очікуваною кількістю значень, що ми маємо обрати для отримання першої колізії. Якщо функція $f(x)$ породжує будь-яке значення H рівноімовірно і H достатньо велика множина, тоді ми очікуємо отримати пару різних аргументів x_1 та x_2 з $f(x_1)=f(x_2)$ після обчислення функції для близько $1.25\sqrt{H}$ різних аргументів в середньому.

$$Q(H) \approx \sqrt{\frac{\pi}{2}H}.$$

- Як приклад, якщо використовується 64-бітовий хеш, то існує близько 1.8 × 1019 різних виходів. Якщо вони всі рівноймовірні (найкращий випадок), тоді потрібно лише 5.1 × 109 спроб для отримання колізії, із використанням грубої сили. Це число відоме як межа днів народження (англ. birthday bound) і **для n-бітових кодів її можна обрахувати як $2^{n/2}$**.

Атака «днів народження»

- Таблиця показує кількість хешів $n(p)$ необхідних для досягнення заданої ймовірності успіху, припускається, що всі хеші рівноймовірні.

Бітів	Можливих виходів (приблизно)(H)	Бажано ймовірність випадкової колізії (приблизно) (p)									
		10^{-18}	10^{-15}	10^{-12}	10^{-9}	10^{-6}	0.1%	1%	25%	50%	75%
16	6.6×10^{4}	2	2	2	2	2	11	36	1.9×10^{2}	3.0×10^{2}	4.3×10^{2}
32	4.3×10^{9}	2	2	2	2.9	93	2.9×10^{3}	9.3×10^{3}	5.0×10^{4}	7.7×10^{4}	1.1×10^{5}
64	1.8×10^{19}	6.1	1.9×10^{2}	6.1×10^{3}	1.9×10^{5}	6.1×10^{6}	1.9×10^{8}	6.1×10^{8}	3.3×10^{9}	5.1×10^{9}	7.2×10^{9}
128	3.4×10^{38}	2.6×10^{10}	8.2×10^{11}	2.6×10^{13}	8.2×10^{14}	2.6×10^{16}	8.3×10^{17}	2.6×10^{18}	1.4×10^{19}	2.2×10^{19}	3.1×10^{19}
256	1.2×10^{77}	4.8×10^{29}	1.5×10^{31}	4.8×10^{32}	1.5×10^{34}	4.8×10^{35}	1.5×10^{37}	4.8×10^{37}	2.6×10^{38}	4.0×10^{38}	5.7×10^{38}
384	3.9×10^{115}	8.9×10^{48}	2.8×10^{50}	8.9×10^{51}	2.8×10^{53}	8.9×10^{54}	2.8×10^{56}	8.9×10^{56}	4.8×10^{57}	7.4×10^{57}	1.0×10^{58}
512	1.3×10^{154}	1.6×10^{68}	5.2×10^{69}	1.6×10^{71}	5.2×10^{72}	1.6×10^{74}	5.2×10^{75}	1.6×10^{76}	8.8×10^{76}	1.4×10^{77}	1.9×10^{77}

Захист від колізій

Існує кілька методів захисту від зламу, підробки паролів, підписів, сертифікатів, навіть якщо зловмиснику відомі методи побудови колізій для якої-небудь хеш-функції.

- Одним з методів є метод «salt» (сіль), який застосовується при збереженні UNIX-паролів — додавання деякої послідовності символів перед хешуванням. Іноді ця послідовність додається до отриманого хеша. Після такої процедури, підсумкові хеш-таблиці значно складніше аналізувати, а через те, що послідовність секретна, істотно підвищується складність побудови колізій — зловмиснику має бути відома послідовність «salt».
- Іншим методом є конкатенація хешів, отриманих від двох різних хеш-функцій. При цьому, для того щоб підібрати колізії до хеш-функції $C(x)= y(x)||z(x)$, яка є конкатенацією хеш-функції $y(x)$ та $z(x)$, необхідно знати методи побудови колізій для $y(x)$ та $z(x)$. Недоліком конкатенації є збільшення розміру хеша, що часто неприйнятно в реальних застосунках.

Використання колізій для зламу

- Як приклад можна розглянути процедуру автентифікації користувача:

- при реєстрації в системі користувач вводить свій пароль, до якого застосовується деяка хеш-функція, значення якої записується в базу даних;
- при кожному введені паролю, до нього застосовується та сама хеш-функція, а результат порівнюється з тим, що записаний в БД.

При такому підході, навіть якщо зловмисник отримає доступ до БД, він не зможе відновити паролі користувачів (при умові незворотності хеш-функції). Однак, якщо зловмисник вміє знаходити колізії для цієї хеш-функції, він зможе знайти підробний пароль, який буде мате ту саму хеш-суму, що і пароль користувача.

- Можна використати колізії для підробки повідомлень: інформація про валютні операції, наприклад, часто шифрується за допомогою хеш-функцій; зловмисник, володіючи методом знаходження колізій цієї хеш-функції, може підмінити повідомлення підробним і тим самим вплинути на хід валютної операції.
- Схожим чином можна використати колізії для підробки цифрового підпису і сертифіката.

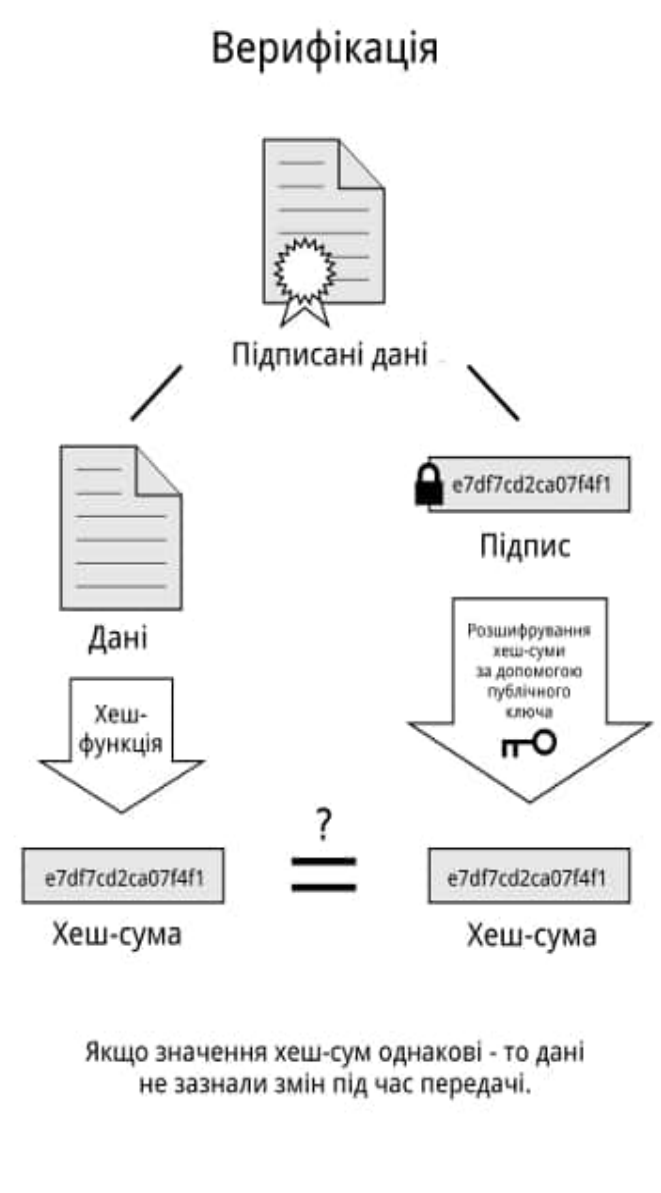
Електронний цифровий підпис

Електронний цифровий підпис (ЕЦП) (англ. digital signature) — вид електронного підпису, отриманого за результатом криптографічного перетворення набору електронних даних, який додається до цього набору або логічно з ним поєднується і дає змогу підтвердити його цілісність та ідентифікувати підписувача. Електронний цифровий підпис накладається за допомогою особистого ключа та перевіряється за допомогою відкритого ключа.

Електронний цифровий підпис призначений для використання фізичними та юридичними особами — суб'єктами електронного документообігу:

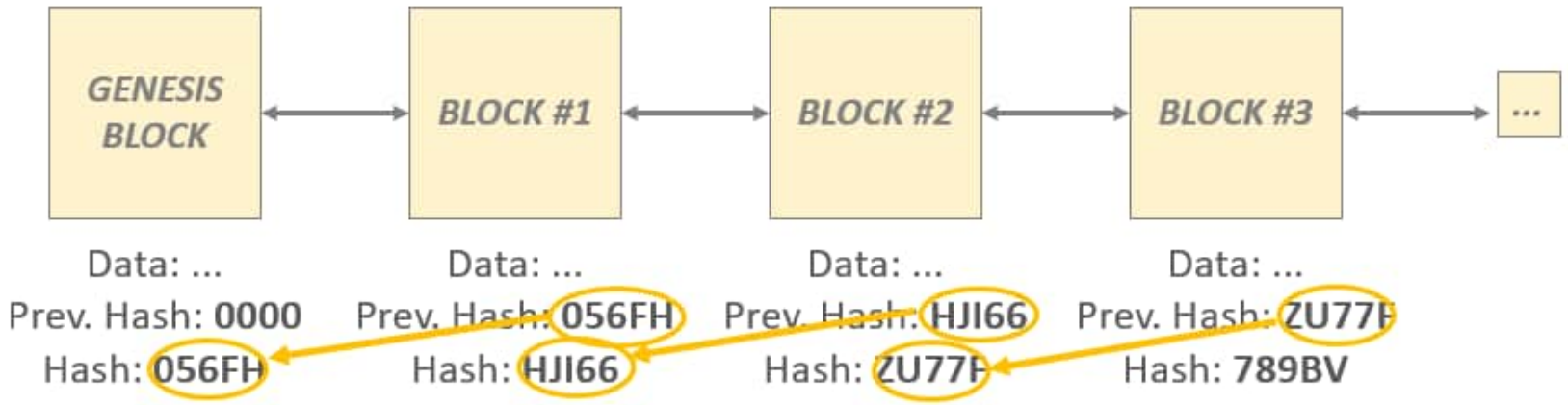
- для аутентифікації підписувача;
- для підтвердження цілісності даних в електронній формі.

Відкритий ключ використовується для перевірки ЕЦП одержуваних документів (файлів). Відкритий ключ працює тільки в парі з приватним ключем. Відкритий ключ міститься в ***Сертифікаті відкритого ключа***, і підтверджує приналежність відкритого ключа ЕЦП певній особі. З метою забезпечення цілісності представлених у Сертифікаті даних він підписується особистим ключем Центру сертифікації ключів.



Цифрова валюта

- **Криптовалюта (Bitcoin) -** принциповою особливістю криптовалют є збереження інформації у блокчейні



→ so the blockchain itself is **a linked list** with hash-pointers

→ every node in the blockchain has **2** hash values – own hash and the hash value of the previous block (**SHA256** hashes)