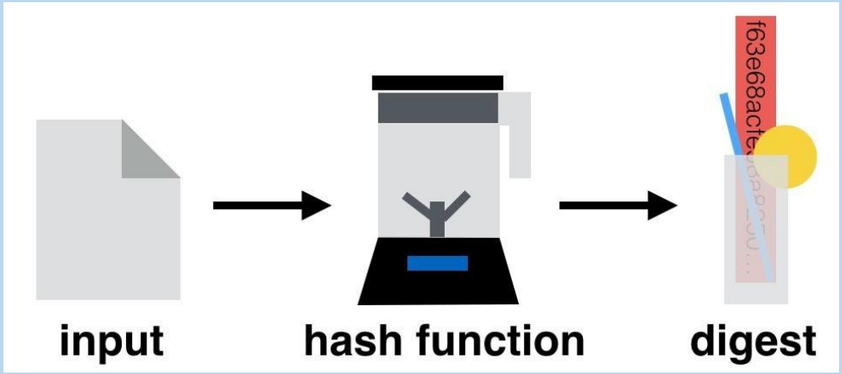
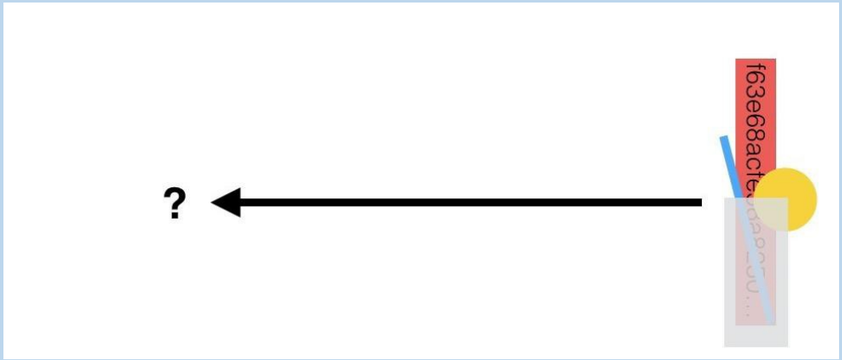
ЛЕКЦІЯ 11

ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Криптографічні хеш-функції



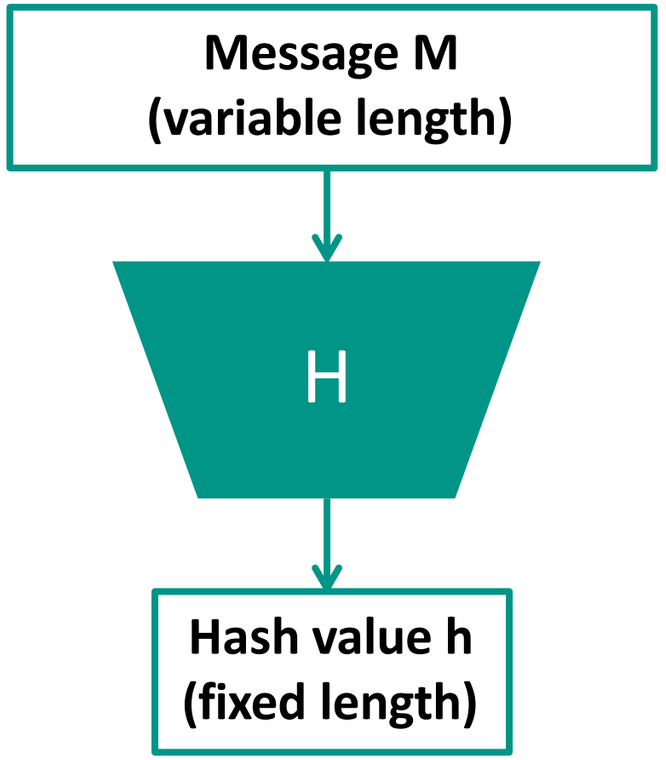


План

1. Поняття хеш-функції
2. Хеш-функція SHA-256
3. Хеш-функція «Купина» (ДСТУ 7564:2014)

1. Поняття хеш-функції

Хеш-функція являє собою функцію, математичну або іншу, що отримує на вхід рядок змінної довжини і перетворює його в рядок фіксованої, зазвичай меншої, довжини

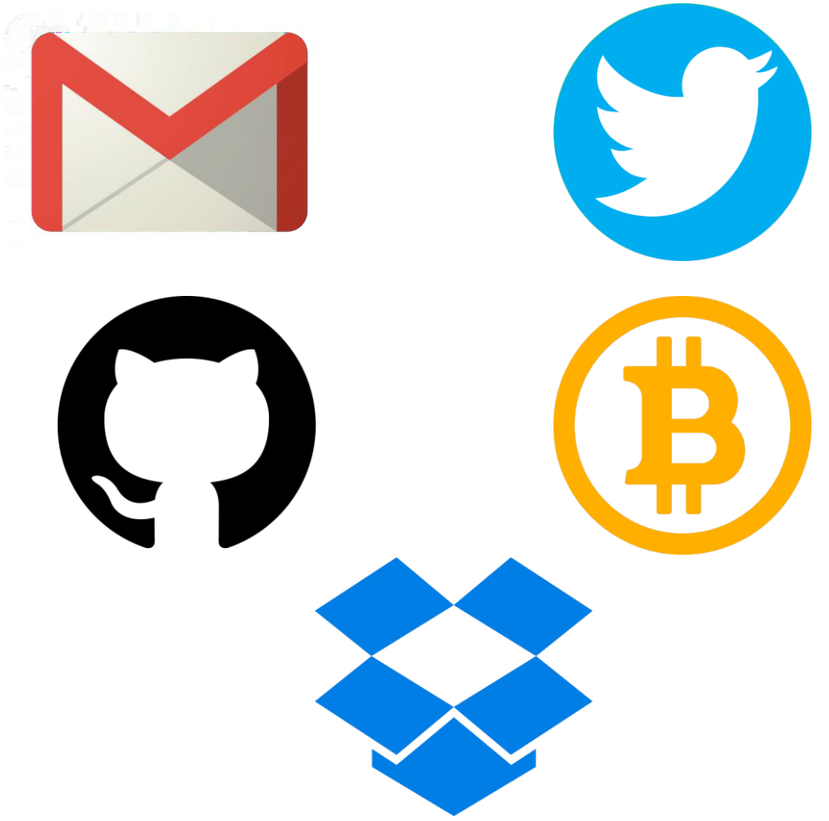


Результат хеш-функції називають хешем, хеш-значенням або дайджестом

1. Поняття хеш-функції

Застосування хеш-функції

- ✓Перевірка цілісності повідомлень та файлів;
- ✓Генерація і перевірка електронного цифрового підпису;
- ✓ Перевірка пароля;
- ✓ Ідентифікатор файлу або даних.



1. Поняття хеш-функції

Принцип роботи криптографічної 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)

1. Поняття хеш-функції

Принцип роботи криптографічної хеш-функцій

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)

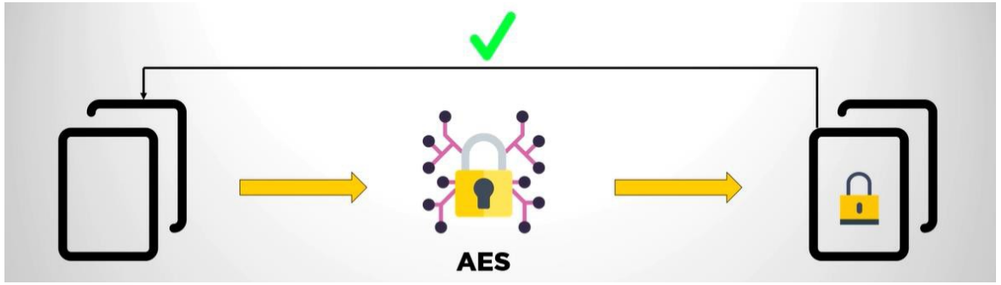
Хеш-значення, обчислене при використанні останнього блоку повідомлення, вважається хеш-значенням усього повідомлення M

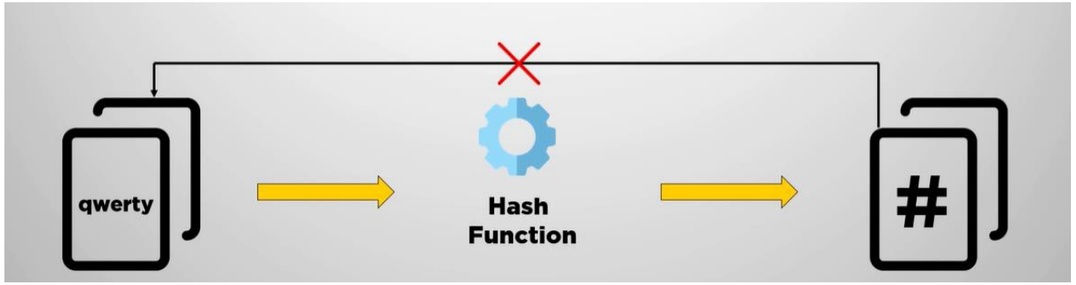
1. Поняття хеш-функції

Основні властивості криптографічної хеш-функції

1) Детермінованість – для однакових повідомлень *M* функція має повертати однакові хеш-значення *h*;

2) Односторонність – за значенням *h* неможливо відновити *M*;

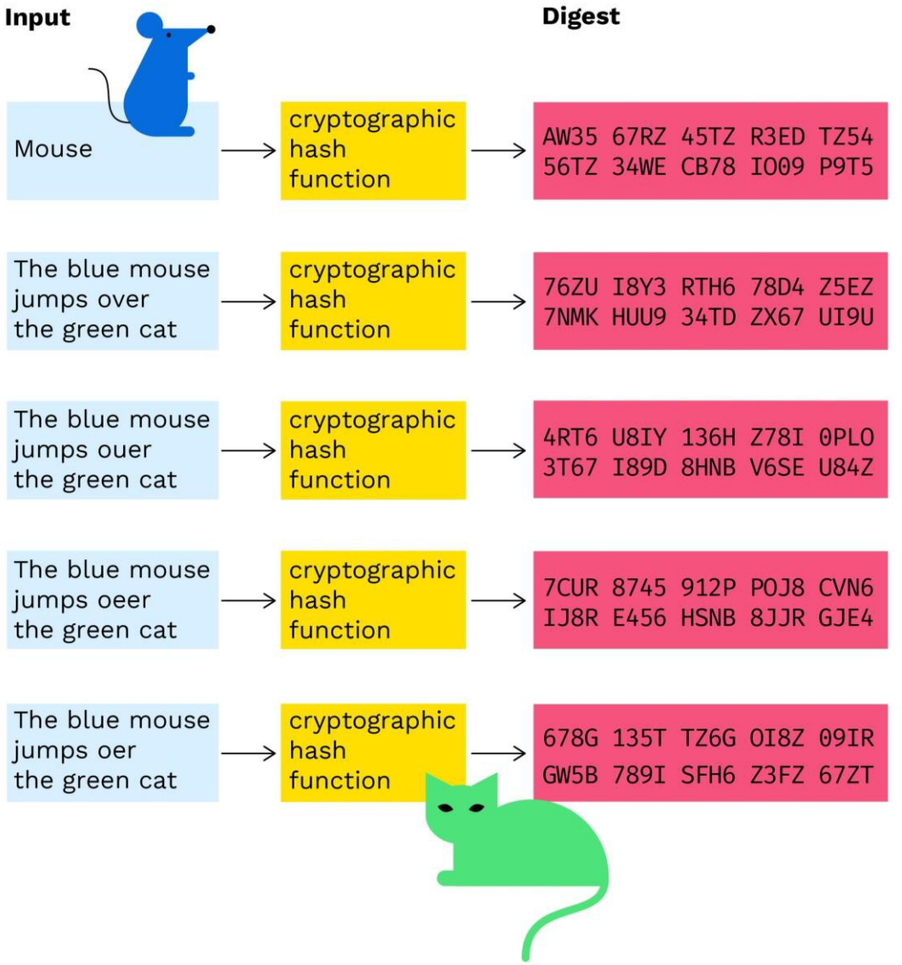




1. Поняття хеш-функції

Основні властивості криптографічної хеш-функції

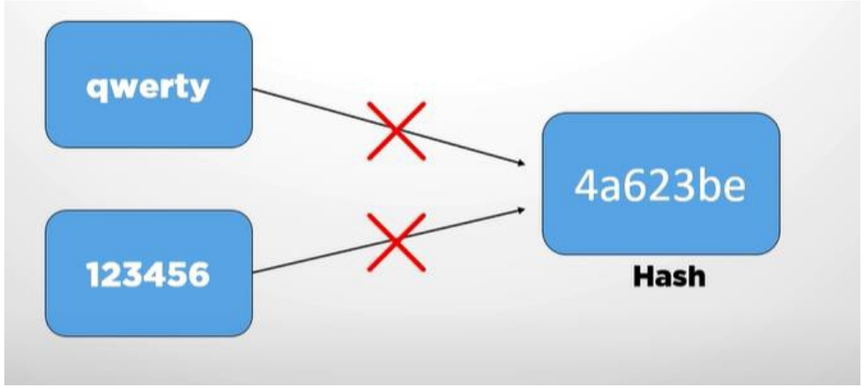
3) Наявність лавинного ефекту – будь-які, навіть незначні, зміни у повідомленні *M* призводять до значних змін у хеш-значенні *h*;



1. Поняття хеш-функції

Основні властивості криптографічної хеш-функції

4) Відсутність колізій (унікальність хеша) – ймовірність співпадіння хеш-значень двох різних повідомлень повинна бути надзвичайно малою;



5) Висока швидкість роботи.

1. Поняття хеш-функції

Хеш-функції

MD4

•**Message Digest 4** — це хеш-функція, розроблена в 1990 році, яка генерує 128-бітне хеш-значення для довільного вхідного повідомлення. Цей алгоритм використовувався для виконання процедур автентифікації для віддалених робочих станцій Windows. Алгоритм MD4 є попередником алгоритму MD5.

MD5

•**Message Digest 5** — це 128-бітний алгоритм хешування, розроблений у 1991 році, призначений для обчислення хеш-значень для повідомлень довільної довжини та перевірки їх автентичності. Його широко використовували для перевірки цілісності інформації та зберігання хеш-значень паролів. MD5 є більш надійною версією алгоритму MD4.

SHA-1

•**Secure Hash Algorithm 1** алгоритм хешування, який генерує 160-бітне (20 байт) хеш-значення для вхідного повідомлення довільної довжини, зазвичай відображається як 40-значне шістнадцяткове число. Він використовується в багатьох криптографічних програмах і протоколах. SHA-1 базується на тих самих принципах, що й MD4.

SHA-2

•**Secure Hash Algorithm версії 2** — це сімейство односторонніх хеш-функцій, що включає алгоритми SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/256 і SHA-512/224. Хеш-функції призначені для обчислення хеш-значень для повідомлень довільної довжини та використовуються в програмах і компонентах, пов'язаних з інформаційною безпекою.

SHA-3

•**Secure Hash Algorithm версії 3 або Keccak** — це алгоритм хешування, який реалізовано на основі підходу, що називається губчастою функцією. Блоки вхідних даних довжиною 1152, 1088, 832 або 576 біт «поглинаються», виробляючи («видавлюючи») хеш-значення 224, 256, 384 або 512 біт відповідно. Стандарт SHA-3 також визначає два алгоритми під назвою SHAKE128 і SHAKE256, які можуть створювати хеші змінної довжини, навіть дуже довгі. SHAKE у цьому випадку означає «Secure Hash Algorithm with Keccak», а рівень безпеки представлено числами 128 і 256.

Купина

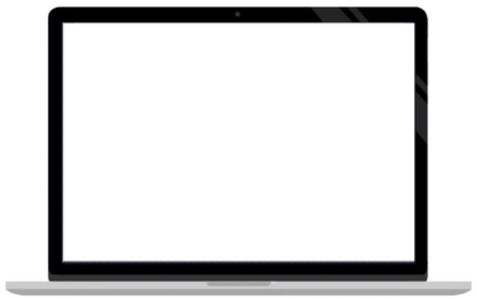
•Ітераційна криптографічна хеш-функція **Купина**, прийнята як національний стандарт України, генерує хеш-значення заданої довжини. Алгоритм заснований на використанні чотирьох S-блоків, структура яких запозичена з шифру Kalyna. Хеш-функція використовується при взаємодії з державними установами та банками.

1. Поняття хеш-функції

Код автентифікації повідомлення

Код автентифікації повідомлення (MAC) — це спеціальний набір символів, який додається до повідомлення та призначений для забезпечення його цілісності та автентифікації джерела даних.

Нехай два абоненти планують обмінятися повідомленнями (ключ шифрування відомий обом сторонам і передається заздалегідь через захищений канал зв’язку або за допомогою алгоритму Діффі-Хеллмана). Тільки сторона-відправник і одержувач знають ключ і алгоритм, і вони не були скомпрометовані.



MAC: Message Authentication Code

If the same MAC is found: then the message is authentic and integrity checked
Else: something is not right.

1)Відправник використовує ключ для обчислення значення MAC з повідомлення. Потім повідомлення та значення MAC надсилаються одержувачу.

2)Одержувач застосовує той самий алгоритм MAC до повідомлення та обчислює значення MAC. Використовується той самий ключ.

3)Одержувач порівнює значення MAC, отримане від відправника, з результатом ободержувач може з упевненістю припустити, числення. Якщо вони ідентичні, що повідомлення не було змінено або підроблено під час передачі, таким чином підтверджуючи цілісність даних.

1. Поняття хеш-функції

Механізм обміну даними за допомогою секретного ключа (наприклад, у MAC) і хеш-функцій називається хеш-кодом автентифікації повідомлення (HMAC). Криптографічна стійкість HMAC залежить від криптографічної стійкості основної хеш-функції, її вихідного розміру хешування, а також розміру та якості ключа. Використання коду автентифікації повідомлення забезпечує цілісність повідомлення та дозволяє перевірити, чи його надіслала довірена сторона.

Алгоритми MAC

На основі алгоритмів хешування.Це найпопулярніші алгоритми, такі як HMAC (MAC на основі хешу, наприклад, HMAC-SHA1 або HMAC-SHA256) і KMAC (MAC на основі Keccak, який також можна використовувати без ключа як звичайну хеш-функцію).

На основі симетричних шифрів. Такі алгоритми, як CMAC (MAC на основі шифру, безкоштовний і не охоплений жодними патентами), GMAC (MAC Galois) і Poly1305 (одноразовий автентифікатор Bernstein).

Інші алгоритми MAC. До них належать UMAC (на основі універсального хешування), VMAC (високопродуктивний MAC на основі блокового шифру) і SipHash (простий, швидкий, безпечний MAC).

2. Хеш-функція SHA-256

SHA-256 є окремим випадком хеш-функції із сімейства криптографічних хеш-функцій SHA-2 (*Secure Hash Algorithm Version 2*) опублікованим АНБ США в 2002 році

Довжина вхідних даних:

до $2^{64} - 1$ біт

Довжина блоку:

512 біт

Довжина хешу:

256 біт

Кількість циклів:

64

2. Хеш-функція SHA-256

Алгоритм SHA-256

1.Попередня обробка, що полягає у доповненні початкового повідомлення та його розбиття на блоки по 512 біт.

2.Ініціалізація значень хешу. Використовуються константи, що представляють собою перші 32 біта дробових частин квадратних коренів перших 8 простих чисел: 2, 3, 5, 7, 11, 13, 17, 19).

```
h0 := 0x6a09e667
h1 := 0xbb67ae85
h2 := 0x3c6ef372
h3 := 0xa54ff53a
h4 := 0x510e527f
h5 := 0x9b05688c
h6 := 0x1f83d9ab
h7 := 0x5be0cd19
```

$$\sqrt{19} = 4{,}35889894354_{10}$$

$$4{,}35889894354_{10} \rightarrow$$
$$\rightarrow 100.0101\ 1011\ 1110 \ldots 1001_2 \rightarrow$$
$$\rightarrow 4{,}5beocd19137e2179_{16}.$$

2. Хеш-функція SHA-256

Алгоритм SHA-256

3. Створення масиву констант k [0..63]. Використовуються ще 64 константи – це перші 32 біта дробових частин кубічних коренів перших 64 простих чисел (2 – 311).

```
k[0..63] :=
   0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
   0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
   0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
   0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
   0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
   0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
   0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
   0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2
```

2. Хеш-функція SHA-256

Алгоритм SHA-256

4. Основний цикл. Наступні кроки будуть виконуватися для кожного 512-бітного блоку вхідних даних. На кожній ітерації циклу буде змінюватися значення h0 – h7:

Крок 1. Кожен блок ділиться на 16 слів (кожне слово – 32 біти) та записується у масив w[0..15];

Крок 2. Додається (в кінець масиву) ще 48 слів, ініціалізованих нулями, щоб отримати масив w[0..63];

Крок 3. Нульові елементи w[16..63] замінюються на нові за алгоритмом:

```
for i from 16 to 63
    s0 := (w[i-15] rightrotate  7) xor (w[i-15] rightrotate 18) xor (w[i-15] rightshift  3)
    s1 := (w[i- 2] rightrotate 17) xor (w[i- 2] rightrotate 19) xor (w[i- 2] rightshift 10)
    w[i] := w[i-16] + s0 + w[i-7] + s1
```

2. Хеш-функція SHA-256

Алгоритм SHA-256

Крок 4. Ініціалізація змінних a, b, c, d, e, f, g, h поточними значенням хешу відповідно h0, h1, h2, h3, h4, h5, h6, h7;

```
a := h0
b := h1
c := h2
d := h3
e := h4
f := h5
g := h6
h := h7
```

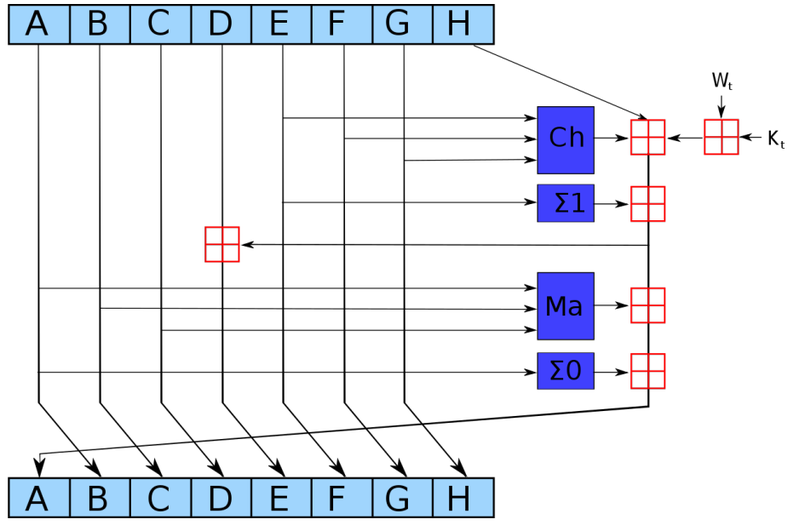
2. Хеш-функція SHA-256

Алгоритм SHA-256

Крок 5. Виконується цикл стиснення, який буде змінювати усі значення від a до h наступним чином:

```
for i from 0 to 63
    S1 := (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)
    ch := (e and f) xor ((not e) and g)
    temp1 := h + S1 + ch + k[i] + w[i]
    S0 := (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)
    maj := (a and b) xor (a and c) xor (b and c)
    temp2 := S0 + maj

    h := g
    g := f
    f := e
    e := d + temp1
    d := c
    c := b
    b := a
    a := temp1 + temp2
```



2. Хеш-функція SHA-256

Алгоритм SHA-256

Крок 6. До значень h0…h7 додаються відповідні змінні a…h (за модулем 2^{32}):

```
h0 := h0 + a
h1 := h1 + b
h2 := h2 + c
h3 := h3 + d
h4 := h4 + e
h5 := h5 + f
h6 := h6 + g
h7 := h7 + h
```

5. Фінальний хеш є конкатенацією:

```
digest := hash := h0 append h1 append h2 append h3 append h4 append h5 append h6 append h7
```

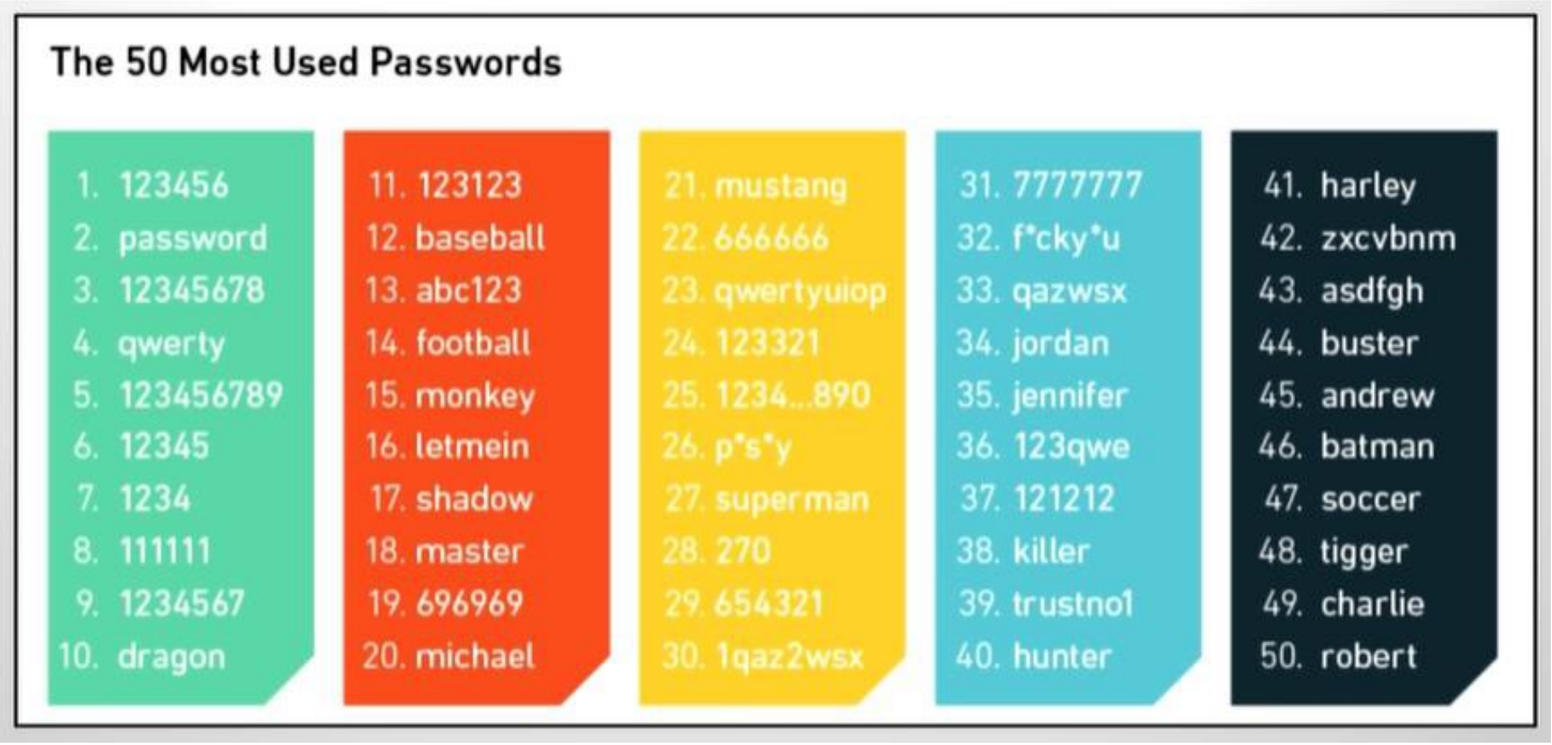
2. Хеш-функція SHA-256

Наприклад, у twitter використовується SHA-256 для збереження паролів користувачів:

Users	Hash
User1	YY9J3IES8K
User2	HTOjXKSLBG
User3	CWBQB3R5G
User4	EGPR20YLY5
User5	CARPNNFIJW
User6	PJLJQDRVCO
User7	CH28YHE5IQ

2. Хеш-функція SHA-256

Обчисливши, хеш-значення найчастіше вживаних паролів, можна підібрати пароль:

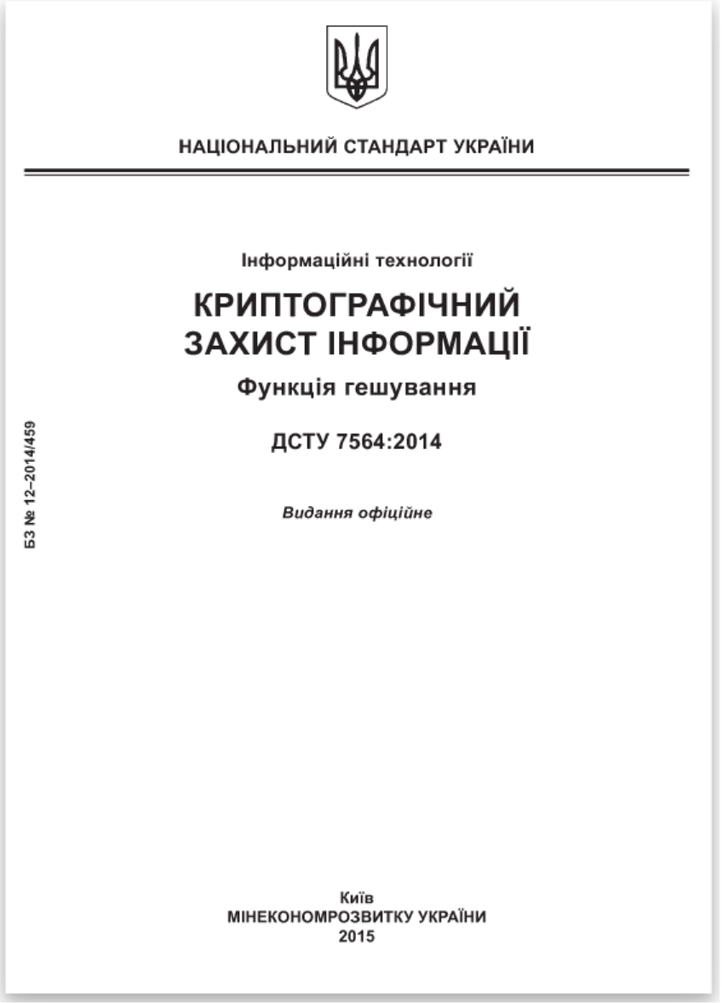


Password	Hash
123456	LRHZAFVUZM
qwerty	R6JTUOGLUG
letmein	YB14YN8280
iloveyou	CARPNNFIJW
654321	4LEJZ8EBB5
mypassword	EAHY7W8LH7
trytohackme	G6GP9LMT99

3. Хеш-функція «Купина» (ДСТУ 7564:2014)

У грудні 2014 року прийнято національний стандарт ***ДСТУ 7564:2014*** (введений в дію 1 квітня 2015 р.), що базується на криптографічній функції хешування «Купина»

Хеш-функція «Купина» використовується зокрема й для створення та перевірки електронного цифрового підпису, що визначений у *ДСТУ 4145*



НАЦІОНАЛЬНИЙ СТАНДАРТ УКРАЇНИ

Інформаційні технології

КРИПТОГРАФІЧНИЙ ЗАХИСТ ІНФОРМАЦІЇ

Функція гешування

ДСТУ 7564:2014

Видання офіційне

БЗ № 12–2014/459

Київ
МІНЕКОНОМРОЗВИТКУ УКРАЇНИ
2015

3. Хеш-функція «Купина» (ДСТУ 7564:2014)

Походження назви:

Купина лікарська – рослина, що росте в хвойних і мішаних лісах, занесена до Червоної книги України

3. Хеш-функція «Купина» (ДСТУ 7564:2014)

Довжина вхідних даних:

до $2^{96} - 1$ біт

Довжина блоку:

512 біт для $8 \leq n \leq 256$
(8 стовпців у матриці стану)

1024 біт для $256 < n \leq 512$
(16 стовпців у матриці стану)

Довжина хешу:

від 8 до 512 біт

Варіант, який повертає n біт, позначається як «Купина-n»

Кількість ітерацій:

10 для $8 \leq n \leq 256$

14 для $256 < n \leq 512$

3. Хеш-функція «Купина» (ДСТУ 7564:2014)

Алгоритм «Купина»

1. Розбиття повідомлення на блоки. Повідомлення M розбивають на t блоків $m_1, m_2, \dots, m_t$ завдовжки l бітів кожен

2. Доповнення останнього блоку. В кінець повідомлення M довжини N додається додаткова інформація, яка містить одиничний біт «1», d нульових біт, які визначаються за формулою:

$$d = (-N - 97) mod\, l$$

Після цього додають ще 96 бітів, в яких міститься значення довжини повідомлення N

3. Хеш-функція «Купина» (ДСТУ 7564:2014)

Алгоритм «Купина»

3. Обчислення хеш-значення за такою ітеративною процедурою:

$$h_0 = IV,$$

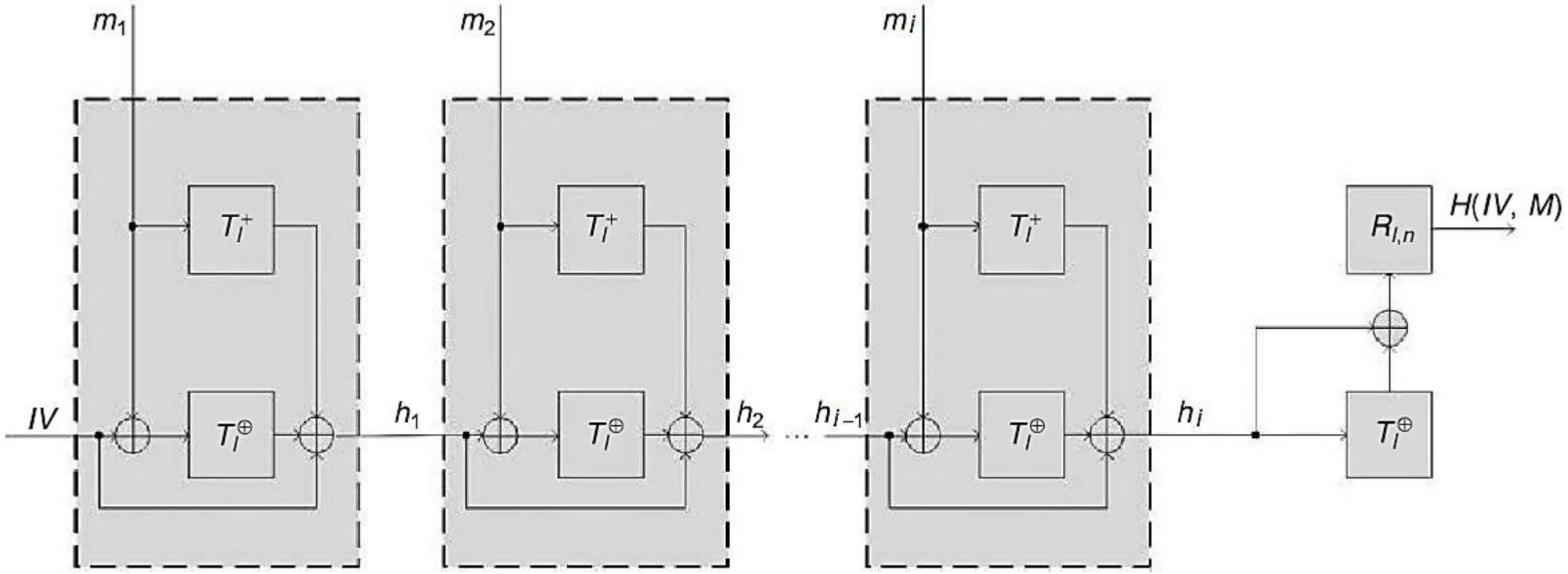
$$h = T^{\oplus}(h_{i-1} \oplus m) \oplus T^{+}(m) \oplus h_{i-1}, \text{ де } i = 1, \dots, t,$$

$$H(IV, M) = R_{l,n}\left(T_l^{\oplus}(h_t) \oplus h_t\right)$$

4. Завершальне перетворення результату хешування являє собою функцію $R_{l,n}(x)$, що повертає n старших біт з вхідного блоку x довжиною l біт ($n < l$), де результат записується в молодші n біт обчисленого значення

3. Хеш-функція «Купина» (ДСТУ 7564:2014)

Алгоритм «Купина»



3. Хеш-функція «Купина» (ДСТУ 7564:2014)

Алгоритм «Купина»

Перетворення $T_l^{\oplus}$ та T_l^{+} виконуються над матрицею стану, кожним елементом якої є один байт та містять такі операції:

1. Додавання з константами ітерації;
2. Підстановка байтів;
3. Зсув рядків;
4. Перемішування стовпців.

Матриця стану 512 біт:

Вхідна послідовність							
In_0	In_8	In_{16}	In_{24}	In_{32}	In_{40}	In_{48}	In_{56}
In_1	In_9	In_{17}	In_{25}	In_{33}	In_{41}	In_{49}	In_{57}
In_2	In_{10}	In_{18}	In_{26}	In_{34}	In_{42}	In_{50}	In_{58}
In_3	In_{11}	In_{19}	In_{27}	In_{35}	In_{43}	In_{51}	In_{59}
In_4	In_{12}	In_{20}	In_{28}	In_{36}	In_{44}	In_{52}	In_{60}
In_5	In_{13}	In_{21}	In_{29}	In_{37}	In_{45}	In_{53}	In_{61}
In_6	In_{14}	In_{22}	In_{30}	In_{38}	In_{46}	In_{54}	In_{62}
In_7	In_{15}	In_{23}	In_{31}	In_{39}	In_{47}	In_{55}	In_{53}

Внутрішній стан функції хешування							
$S_{0,0}$	S_{012}	$S_{0,2}$	$S_{0,3}$	$S_{0,4}$	$S_{0,5}$	$S_{0,6}$	$S_{0,7}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,4}$	$S_{1,5}$	$S_{1,6}$	$S_{1,7}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$	$S_{2,4}$	$S_{2,5}$	$S_{2,6}$	$S_{2,7}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$	$S_{3,4}$	$S_{3,5}$	$S_{3,6}$	$S_{3,7}$
$S_{4,0}$	$S_{4,1}$	$S_{4,2}$	$S_{4,3}$	$S_{4,4}$	$S_{4,5}$	$S_{4,6}$	$S_{4,7}$
$S_{5,0}$	$S_{5,1}$	$S_{5,2}$	$S_{5,3}$	$S_{5,4}$	$S_{5,5}$	$S_{5,6}$	$S_{5,7}$
$S_{6,0}$	$S_{6,1}$	$S_{6,2}$	$S_{6,3}$	$S_{6,4}$	$S_{6,5}$	$S_{6,6}$	$S_{6,7}$
$S_{7,0}$	$S_{7,1}$	$S_{7,2}$	$S_{7,3}$	$S_{7,4}$	$S_{7,5}$	$S_{7,6}$	$S_{7,7}$

3. Хеш-функція «Купина» (ДСТУ 7564:2014)

Алгоритм «Купина»

Додавання за модулем 2 ($T^{\oplus}$) елементів матриці внутрішнього стану та констант ітерацій:

$$C^i = \begin{bmatrix} 00\oplus i & 10\oplus i & 20\oplus i & 30\oplus i & 40\oplus i & 50\oplus i & 60\oplus i & 70\oplus i \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \end{bmatrix};$$

$$C^i = \begin{bmatrix} 00\oplus i & 10\oplus i & 20\oplus i & 30\oplus i & 40\oplus i & 50\oplus i & \dots & f0\oplus i \\ 00 & 00 & 00 & 00 & 00 & 00 & \dots & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & \dots & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & \dots & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & \dots & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & \dots & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & \dots & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & \dots & 00 \end{bmatrix};$$

3. Хеш-функція «Купина» (ДСТУ 7564:2014)

Алгоритм «Купина»

Додавання за модулем 2^{64} (T^{+}) елементів матриці внутрішнього стану та констант ітерацій:

$$C^i = \begin{bmatrix} f3 & f3 & f3 & f3 & f3 & f3 & f3 & f3 \\ f0 & f0 & f0 & f0 & f0 & f0 & f0 & f0 \\ f0 & f0 & f0 & f0 & f0 & f0 & f0 & f0 \\ f0 & f0 & f0 & f0 & f0 & f0 & f0 & f0 \\ f0 & f0 & f0 & f0 & f0 & f0 & f0 & f0 \\ f0 & f0 & f0 & f0 & f0 & f0 & f0 & f0 \\ f0 & f0 & f0 & f0 & f0 & f0 & f0 & f0 \\ 70 \oplus i & 60 \oplus i & 50 \oplus i & 40 \oplus i & 30 \oplus i & 20 \oplus i & 10 \oplus i & 00 \oplus i \end{bmatrix};$$

$$C^i = \begin{bmatrix} f3 & f3 & f3 & f3 & f3 & f3 & \dots & f3 \\ f0 & f0 & f0 & f0 & f0 & f0 & \dots & f0 \\ f0 & f0 & f0 & f0 & f0 & f0 & \dots & f0 \\ f0 & f0 & f0 & f0 & f0 & f0 & \dots & f0 \\ f0 & f0 & f0 & f0 & f0 & f0 & \dots & f0 \\ f0 & f0 & f0 & f0 & f0 & f0 & \dots & f0 \\ f0 & f0 & f0 & f0 & f0 & f0 & \dots & f0 \\ f0 & f0 & f0 & f0 & f0 & f0 & \dots & f0 \\ f0 \oplus i & e0 \oplus i & d0 \oplus i & c0 \oplus i & b0 \oplus i & a0 \oplus i & \dots & 00 \oplus i \end{bmatrix}.$$

3. Хеш-функція «Купина» (ДСТУ 7564:2014)

Алгоритм «Купина»

Підстановка байтів подібно до «Калини»:

Підстановка π_0:

A8	43	5F	06	6B	75	6C	59	71	DF	87	95	17	F0	D8	09
6D	F3	1D	CB	C9	4D	2C	AF	79	E0	97	FD	6F	4B	45	39
3E	DD	A3	4F	B4	B6	9A	0E	1F	BF	15	E1	49	D2	93	C6
92	72	9E	61	D1	63	FA	EE	F4	19	D5	AD	58	A4	BB	A1
DC	F2	83	37	42	E4	7A	32	9C	CC	AB	4A	8F	6E	04	27
2E	E7	E2	5A	96	16	23	2B	C2	65	66	0F	BC	A9	47	41
34	48	FC	B7	6A	88	A5	53	86	F9	5B	DB	38	7B	C3	1E
22	33	24	28	36	C7	B2	3B	8E	77	BA	F5	14	9F	08	55
9B	4C	FE	60	5C	DA	18	46	CD	7D	21	B0	3F	1B	89	FF
EB	84	69	3A	9D	D7	D3	70	67	40	B5	DE	5D	30	91	B1
78	11	01	E5	00	68	98	A0	C5	02	A6	74	2D	0B	A2	76
B3	BE	CE	BD	AE	E9	8A	31	1C	EC	F1	99	94	AA	F6	26
2F	EF	E8	8C	35	03	D4	7F	FB	05	C1	5E	90	20	3D	82
F7	EA	0A	0D	7E	F8	50	1A	C4	07	57	B8	3C	62	E3	C8
AC	52	64	10	D0	D9	13	0C	12	29	51	B9	CF	D6	73	8D
81	54	C0	ED	4E	44	A7	2A	85	25	E6	CA	7C	8B	56	80

Підстановка π_1:

CE	BB	EB	92	EA	CB	13	C1	E9	3A	D6	B2	D2	90	17	F8
42	15	56	B4	65	1C	88	43	C5	5C	36	BA	F5	57	67	8D
31	F6	64	58	9E	F4	22	AA	75	0F	02	B1	DF	6D	73	4D
7C	26	2E	F7	08	5D	44	3E	9F	14	C8	AE	54	10	D8	BC
1A	6B	69	F3	BD	33	AB	FA	D1	9B	68	4E	16	95	91	EE
4C	63	8E	5B	CC	3C	19	A1	81	49	7B	D9	6F	37	60	CA
E7	2B	48	FD	96	45	FC	41	12	0D	79	E5	89	8C	E3	20
30	DC	B7	6C	4A	B5	3F	97	D4	62	2D	06	A4	A5	83	5F
2A	DA	C9	00	7E	A2	55	BF	11	D5	9C	CF	0E	0A	3D	51
7D	93	1B	FE	C4	47	09	86	0B	8F	9D	6A	07	B9	B0	98
18	32	71	4B	EF	3B	70	A0	E4	40	FF	C3	A9	E6	78	F9
8B	46	80	1E	38	E1	B8	A8	E0	0C	23	76	1D	25	24	05
F1	6E	94	28	9A	84	E8	A3	4F	77	D3	85	E2	52	F2	82
50	7A	2F	74	53	B3	61	AF	39	35	DE	CD	1F	99	AC	AD
72	2C	DD	D0	87	BE	5E	A6	EC	04	C6	03	34	FB	DB	59
B6	C2	01	F0	5A	ED	A7	66	21	7F	8A	27	C7	C0	29	D7

3. Хеш-функція «Купина» (ДСТУ 7564:2014)

Алгоритм «Купина»

Підстановка байтів подібно до «Калини»:

Підстановка π_2:

93	D9	9A	B5	98	22	45	FC	BA	6A	DF	02	9F	DC	51	59
4A	17	2B	C2	94	F4	BB	A3	62	E4	71	D4	CD	70	16	E1
49	3C	C0	D8	5C	9B	AD	85	53	A1	7A	C8	2D	E0	D1	72
A6	2C	C4	E3	76	78	B7	B4	09	3B	0E	41	4C	DE	B2	90
25	A5	D7	03	11	00	C3	2E	92	EF	4E	12	9D	7D	CB	35
10	D5	4F	9E	4D	A9	55	C6	D0	7B	18	97	D3	36	E6	48
56	81	8F	77	CC	9C	B9	E2	AC	B8	2F	15	A4	7C	DA	38
1E	0B	05	D6	14	6E	6C	7E	66	FD	B1	E5	60	AF	5E	33
87	C9	F0	5D	6D	3F	88	8D	C7	F7	1D	E9	EC	ED	80	29
27	CF	99	A8	50	0F	37	24	28	30	95	D2	3E	5B	40	83
B3	69	57	1F	07	1C	8A	BC	20	EB	CE	8E	AB	EE	31	A2
73	F9	CA	3A	1A	FB	0D	C1	FE	FA	F2	6F	BD	96	DD	43
52	B6	08	F3	AE	BE	19	89	32	26	B0	EA	4B	64	84	82
6B	F5	79	BF	01	5F	75	63	1B	23	3D	68	2A	65	E8	91
F6	FF	13	58	F1	47	0A	7F	C5	A7	E7	61	5A	06	46	44
42	04	A0	DB	39	86	54	AA	8C	34	21	8B	F8	0C	74	67

Підстановка π_3:

68	8D	CA	4D	73	4B	4E	2A	D4	52	26	B3	54	1E	19	1F
22	03	46	3D	2D	4A	53	83	13	8A	B7	D5	25	79	F5	BD
58	2F	0D	02	ED	51	9E	11	F2	3E	55	5E	D1	16	3C	66
70	5D	F3	45	40	CC	E8	94	56	08	CE	1A	3A	D2	E1	DF
B5	38	6E	0E	E5	F4	F9	86	E9	4F	D6	85	23	CF	32	99
31	14	AE	EE	C8	48	D3	30	A1	92	41	B1	18	C4	2C	71
72	44	15	FD	37	BE	5F	AA	9B	88	D8	AB	89	9C	FA	60
EA	BC	62	0C	24	A6	A8	EC	67	20	DB	7C	28	DD	AC	5B
34	7E	10	F1	7B	8F	63	A0	05	9A	43	77	21	BF	27	09
C3	9F	B6	D7	29	C2	EB	C0	A4	8B	8C	1D	FB	FF	C1	B2
97	2E	F8	65	F6	75	07	04	49	33	E4	D9	B9	D0	42	C7
6C	90	00	8E	6F	50	01	C5	DA	47	3F	CD	69	A2	E2	7A
A7	C6	93	0F	0A	06	E6	2B	96	A3	1C	AF	6A	12	84	39
E7	B0	82	F7	FE	9D	87	5C	81	35	DE	B4	A5	FC	80	EF
CB	BB	6B	76	BA	5A	7D	78	0B	95	E3	AD	74	98	3B	36
64	6D	DC	F0	59	A9	4C	17	7F	91	B8	C9	57	1B	E0	61

3. Хеш-функція «Купина» (ДСТУ 7564:2014)

Алгоритм «Купина»

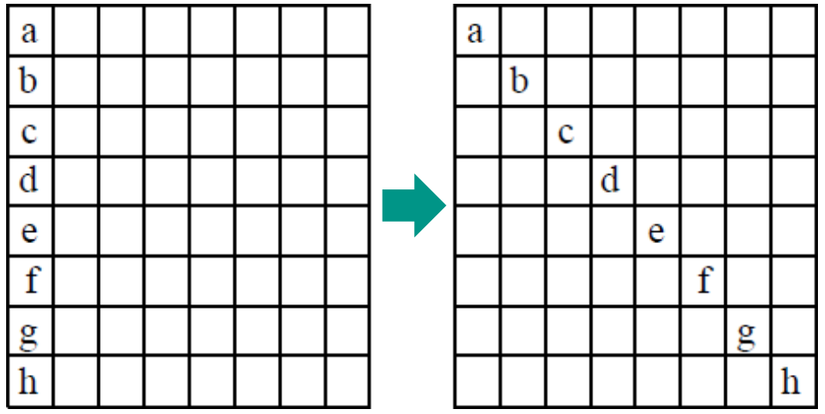
Зсув рядків. Рядки стану циклічно зсувають праворуч на різну кількість байтів, залежно від розміру блока

Номер рядка	Значення зсуву, байтів	
	Внутрішній стан 512 біти	*Внутрішній стан 1024 біти*
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	11

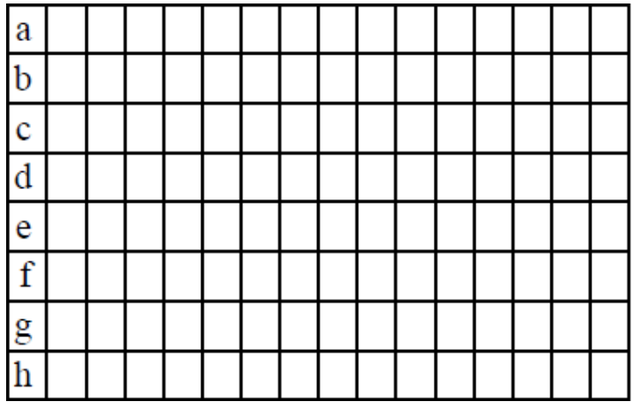
3. Хеш-функція «Купина» (ДСТУ 7564:2014)

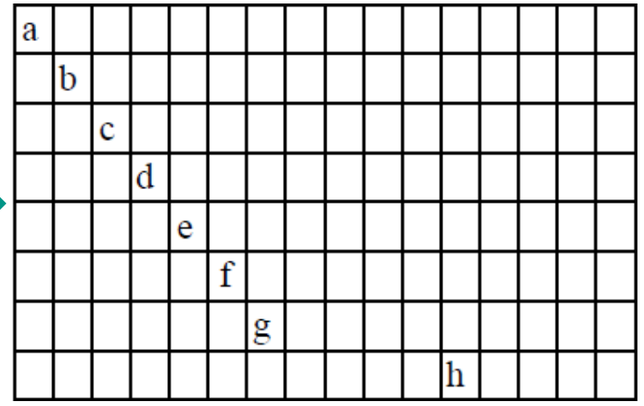
Алгоритм «Купина»

Зсув рядків. Рядки стану циклічно зсувають праворуч на різну кількість байтів, залежно від розміру блока



Зсув рядків 512-бітового блоку





Зсув рядків 1024-бітового блоку

3. Хеш-функція «Купина» (ДСТУ 7564:2014)

Алгоритм «Купина»

Перемішування стовпців. Як і в алгоритмі «Калина» відбувається множення стовпців стану як многочленів над полем $GF(2^8)$ на фіксований многочлен $c(x)$:

$$c(x) = 01_{16} \cdot x^7 + 05_{16} \cdot x^6 + 01_{16} \cdot x^5 + 08_{16} \cdot x^4 + 06_{16} \cdot x^3 + 07_{16} \cdot x^2 + 04_{16} \cdot x + 01_{16}$$