Тема 12. ПРИКЛАДИ ХЕШ-ФУНКЦІЙ

1. Основні ідеї хеш-функцій MD2, MD4, MD5 та MD6

Три алгоритми серії MD розроблені Рівестом (Масачусетський Технологічний інститут) у 1989, 1990, 1991 роках відповідно. Всі вони перетворюють текст будь – якої довжини в 128 – бітну сигнатуру.

Алгоритм MD2 передбачає:

- доповнення тексту до довжини, кратної 128 бітам;
- визначення 16 – бітної контрольної суми (старші розряди відкидаються);
- додавання контрольної суми до тексту;
- повторне визначення контрольної суми.

Алгоритм MD4 передбачає:

- доповнення тексту до довжини, рівної 448 біт за модулем 512;
- додавання довжини тексту в 64 – бітному вираженні;
- 512 – бітні блоки піддають процедурі Damgard – Merkle, для чого кожен блок бере участь у трьох різних циклах.

В алгоритмі MD4 доволі швидко були знайдені “дірки”, тому він був заміненим алгоритмом MD5, в якому кожний блок бере участь не в трьох, а в чотирьох різних циклах.

Після того, як MD5 був взломаний, на його заміну запропонували алгоритм MD6.

MD6 – алгоритм хешування з тої самої серії MD, розроблений професором Рівестом. Використовується для перевірки цілісності і, частково, для перевірки справжності опублікованого повідомлення. Вперше був представлений на конференції Crypto 2008 в якості кандидата на стандарт. Однак, пізніше в 2009 році на цій же конференції Рівест заявив, що MD6 ще не готова хеш-функція, щоб стати стандартом. На офіційному сайті MD6 він заявив, що, хоча, формально, заявка не відізвана, в алгоритмі ще залишаються проблеми зі швидкістю і нездатністю забезпечити безпеку у версії із зменшеною кількістю раундів. В результаті MD6 не пішов на друге коло змагання. Раніше в грудні 2008, дослідник з Fortify Software відкрив помилку,

пов’язану з переповненням буферу в оригінальній реалізації MD6. 19 лютого 2009 року професор Рівест опублікував дані про помилку, а також представив виправлені реалізації.

В алгоритмі хеш-функції використані оригінальні ідеї. За один прохід обробляється 512 байт. Крім традиційного безлючового режиму, застосовується хешування з ключем 512-біт. Не лінійність функції основана на використанні простих операцій: xor, додавання і зміщення. Кількість раундів: r = 40 + (d / 4), що не традиційно багато. [2, 5]

2. Алгоритм обчислення хеш-функції MD5

Алгоритм MD5 (Message Digest) – це алгоритм хешування, розроблений у 1991, опублікований в 1992 році.

В якості вхідних даних береться повідомлення (потік даних) довільної довжини, і обчислюється 128 – розрядний хеш-код (сигнатура, дайджест). Припускається, що не існує двох повідомлень, які мають однакові сигнатури, або не можливо створити повідомлення з наперед заданою сигнатурою. Цей алгоритм застосовується в криптографії при формуванні цифрових підписів та генерації ключів шифрування. По суті, алгоритм хешування MD5 – це спосіб перевірки цілісності даних, набагато надійніший ніж контрольна сума або інші методи.

Алгоритм під час розробки був оптимізований для 32 – розрядних систем і не потребує великих об’ємів пам’яті. MD5 є дещо повільніший, ніж MD4, але більш стійкий до криптографічних атак.

Опис алгоритму

Перед тим, як розглянути детально сам алгоритм, визначимо основні терміни та поняття, що ним використовуються. Під терміном “слово” будемо розуміти кількість інформації, яка представлена 32-ома розрядами (бітами), а під терміном “байт” – 8 розрядів (біт). Послідовність біт як послідовність байт, де кожна група із 8 – ми розрядів є окремим байтом, причому старший розряд байта іде першим. Аналогічно представляється послідовність байт, тільки молодший байт іде першим.

В якості вхідного потоку будемо мати потік даних N розрядів. N - невід'ємне ціле число (можливо 0), не обов'язково кратне 8.

На рис. 12.1 зображена схема логіки виконання MD5.

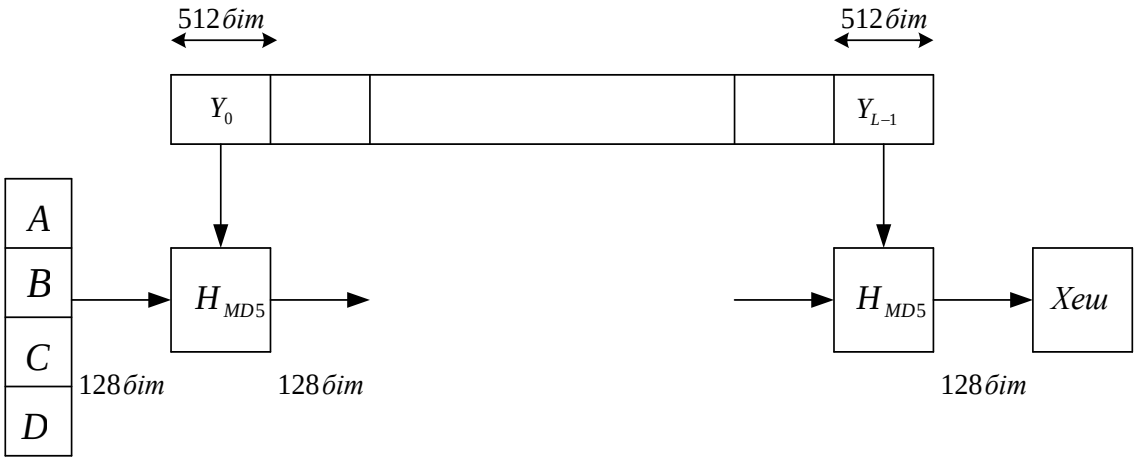

Рис. 12.1 Логіка виконання MD5.

Для обчислення MD5 хеш – функції необхідно виконати наступні 5 кроків:

Крок 1: Вирівнювання потоку

Повідомлення доповнюється таким чином, щоб його довжина була рівною 448 по модулю 512 (довжина = 448 mod 512). Це означає, що довжина доданого повідомлення на 64 розряди меньша ніж число кратне 512. Додавання розрядів проводиться завжди, навіть тоді, коли повідомлення має потрібну довжину. Наприклад, якщо довжина повідомлення 448 розрядів, то воно доповнюється 512 розрядами до 960 розрядів. Отже число доданих розрядів знаходиться в межах від 1 до 512. Вирівнювання відбувається наступним чином: потік доповнюється одним бітом '1', а за ним біти '0' до тих пір, поки довжина потоку не буде рівною 448 по модулю 512.

Крок 2: Додавання довжини

64 – розрядне представлення довжини вихідного (до добавлення) повідомлення в бітах приєднується до результату першого кроку. Якщо початкова довжина більша ніж 2^{64}, то використовуються молодші 64 розряди. Іншими словами поле містить довжину вихідного повідомлення по модулю 2^{64}.

В результаті перших двох кроків утворюється повідомлення, довжина якого кратна 512 розрядам. Це розширене повідомлення представляється, як послідовність 512 – бітних блоків $Y_0, Y_1, ..., Y_{L-1}$, при цьому загальна довжина розширеного повідомлення рівна $L * 512$ розрядам. Таким чином довжина отриманого розширеного повідомлення кратна шістнадцяти 32 – бітним словам.

повідомлення	*доповнення від* 1 *до* 448	*довжина вихідного повідомлення*

Рис. 12.2 Структура розширеного повідомлення.

Крок 3: Ініціалізація MD-буфера

Для збереження проміжних і кінцевих результатів хеш-функції використовується 128-розрядний буфер. Він представляється, як чотири 32 – розрядні регістри (A, B, C, D). В якості ініціалізуючих значень використовуються наступні шістнадцяткові числа:

$$A = 01234567$$
$$B = 89ABCDEF$$
$$C = FEDCBA98$$
$$D = 76543210$$

Крок 4: Обробка послідовності 512 – розрядних (по 16 слів) блоків

Основою алгоритму є блок, який складається із чотирьох циклів. Він позначається як H_{MD5}. Чотири цикли мають подібну структуру, але для кожного циклу застосовується своя елементарна логічна функція f_F, f_G, f_H і f_I відповідно.

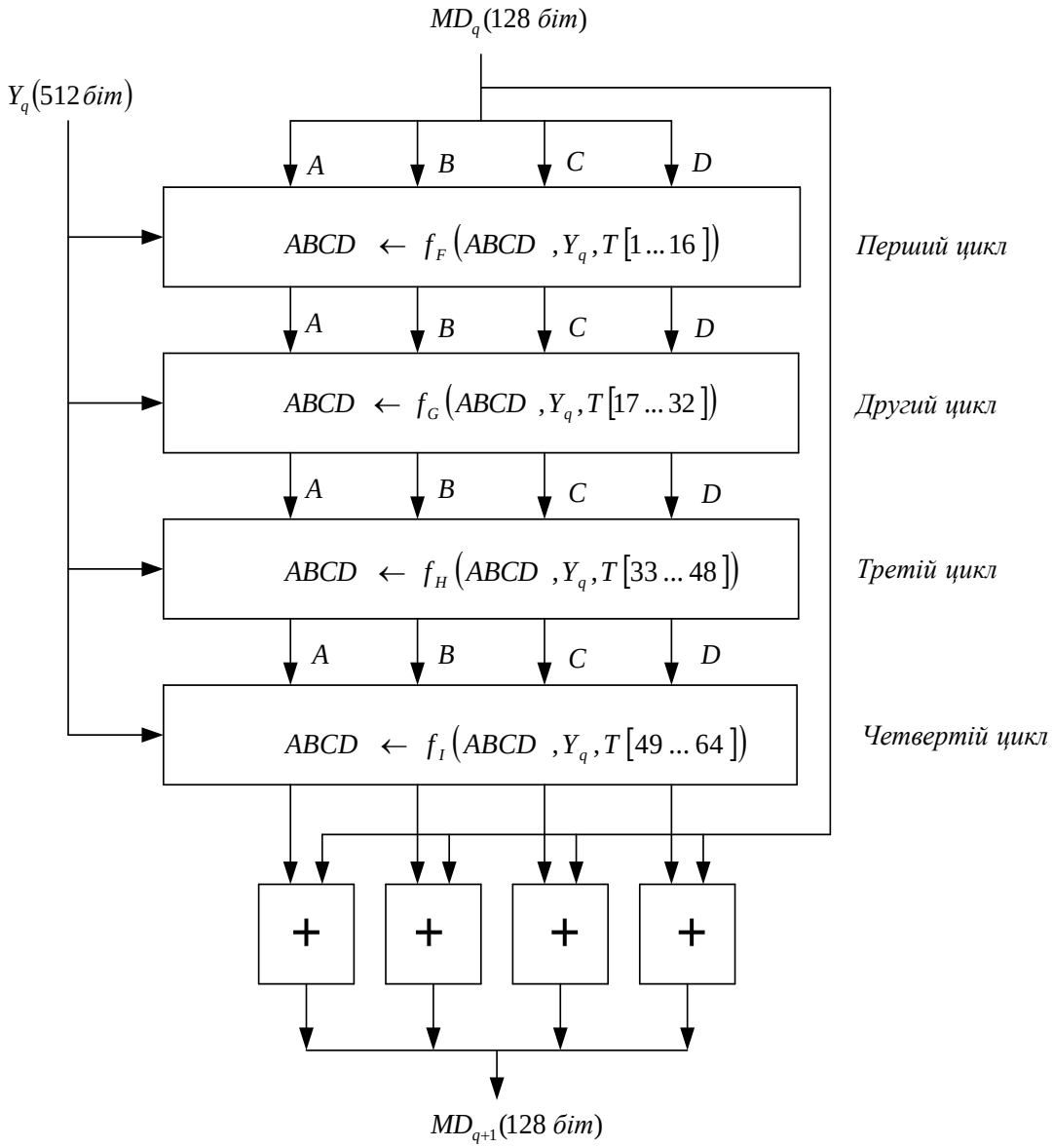

Рис. 12.3 Обробка чергового 512-розрядного блоку.

Кожний цикл отримує в якості вхідних даних поточний 512 – розрядний блок Y_q та 128 – розрядне значення буфера $ABCD$, що є проміжним значенням хеш-функції, та змінює вмістиме цього буфера. Кожний цикл використовує четверту частину 64 – елементної таблиці $T[1...64]$, побудованої на основі функції sin, $T[i] = \text{int}(4294967296 * abs(\sin(i)))$, де int( ) - ціла частина.

Наприклад: $T[1] = \text{int}(4294967296 * abs(\sin(i))) = \text{int}(3614090360{,}282...) = 3614090360$, i - ий елемент T, який позначається $T[i]$, приймає значення рівне цілій частині від $2^{32} * abs(\sin(i))$, число i задається в радіанах. Оскільки функція $abs(\sin(i))$ приймає значення, які знаходяться в проміжку від 0 до 1, то кожний елемент T є цілим, яке можна представити 32 розрядами. Для отримання MD_{q+1} вихід

чотирьох циклів додається по модулю 2^{32} з MD_q. Додавання виконується незалежно для кожного з чотирьох слів в буфері.

Крок 5: Вихід

Після обробки всіх L, 512 - розрядних блоків виходом L-ої стадії є 128 – розрядний дайджест повідомлення.

Розглянемо детальніше логіку кожного із чотирьох циклів обробки одного 512 – розрядного блоку. Цикл обробки складається із 16 кроків, які оперують з буфером $ABCD$.

Кожний крок можна подати у вигляді:

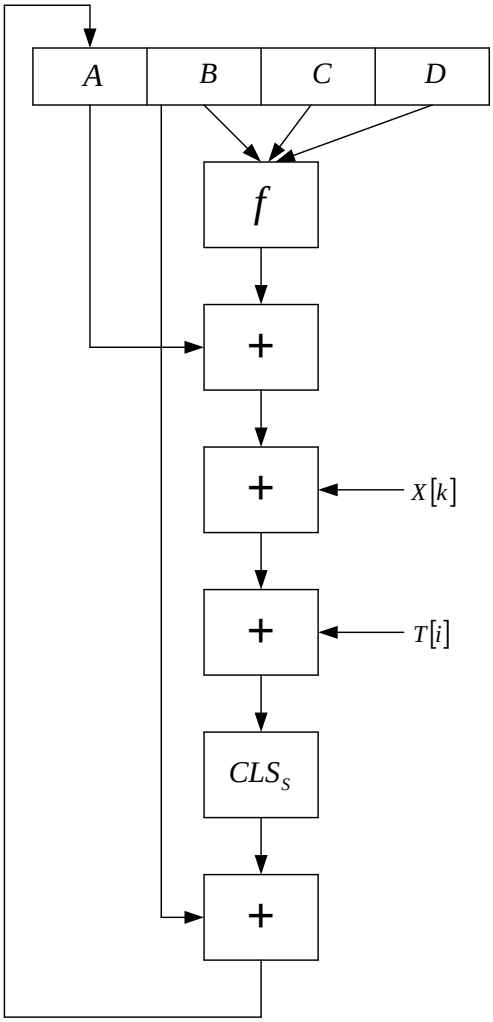

Рис. 12.4 Логіка виконання окремого кроку.

$$A \leftarrow B + CLS_s \ \left(A + f\left(B,C,D\right) + X[k] + T[i]\right),$$

де

A,B,C,D - чотири слова буфера; після виконання кожного окремого кроку відбувається циклічний зсув вліво на одне слово.
f - одна з елементарних функцій f_F, f_G, f_H, f_I.
CLS_s - циклічний зсув вліво на s розрядів 32-розрядного аргументу.
$X[k]$ - k-те 32-розрядне слово в q-му 512-розрядному блоці повідомлення.
$T[i]$ - i-те 32-розрядне слово в матриці T.
$+$ - операція додавання по модулю 2^{32}.

В кожному з чотирьох циклів алгоритму застосовується одна з чотирьох елементарних логічних функцій. Кожна елементарна функція отримує три 32-розрядних слова на вході і на виході формує одне 32-розрядне слово. Елементарні функції мають наступний вигляд:

$$f_F = (B \& C) \vee (not\ B \& D)$$
$$f_G = (B \& D) \vee (C \& not\ D)$$
$$f_H = B \oplus C \oplus D$$
$$f_I = C \oplus (B \& not\ D)$$

В цих формулах застосовуються такі логічні операції: логічне множення (&, порозрядне І), логічне додавання ($\vee$, порозрядне АБО), додавання по модулю 2 ($\oplus$, порозрядне виключне АБО) та інверсія (not, побітове заперечення).

Масив з 32-ох розрядних слів $X[0...15]$ приймає значення поточного 512 – розрядного блоку Кожний цикл виконується 16 раз. Оскільки кожен блок вхідного повідомлення обробляється в чотирьох циклах, то кожний блок вхідного повідомлення обробляється по схемі, яка зображена на рис.4, 64 рази.

Якщо подати вхідний 512 – розрядний блок у вигляді шістнадцяти 32-розрядних слів, то кожне вхідне 32-розрядне слово використовується чотири рази, по одному разу в кожному циклі, і кожний елемент таблиці T, який складається з 64-ьох 32-розрядних слів використовується тільки один раз. Після кожного кроку циклу відбувається циклічний зсув вліво чотирьох слів A,B,C і D. На кожному кроці змінюється тільки одне з чотирьох слів буфера $ABCD$. Відповідно кожне слово буфера змінюється 16 раз, 17-ий раз в кінці для отримання остаточного виходу поточного блоку. Результат обчислення (хеш)

представлений за допомогою чотирьох 32-розрядних слів - A, B, C, D (молодші записуються в A, старші в D).

Приклад псевдокоду алгоритму обчислення MD5 хеш функції наведено нижче.

```
    // обробка вхідного потоку даних блоками по 16 слів
    for i = 0 to N/16 - 1 do
{
  // копіювання i-го блоку в X
  for j = 0 to 15 do
    X[j] = M[i * 16 + j]
  // Збереження  значень A, B, C, D
  AA = A
  BB = B
  CC = C
  DD = D
  // цикл 1
  // нехай [abcd k s i] означають операцію
  //     a = b + ((a + F(b, c, d) + X[k] + T[i]) <<< s)
  // виконати 16 наступних операцій
  [ABCD  0  7  1]  [DABC  1 12  2]  [CDAB  2 17  3]  [BCDA  3 22  4]
  [ABCD  4  7  5]  [DABC  5 12  6]  [CDAB  6 17  7]  [BCDA  7 22  8]
  [ABCD  8  7  9]  [DABC  9 12 10]  [CDAB 10 17 11]  [BCDA 11 22 12]
  [ABCD 12  7 13]  [DABC 13 12 14]  [CDAB 14 17 15]  [BCDA 15 22 16]
  // цикл 2
  // нехай [abcd k s i] означають операцію
  //     a = b + ((a + G(b, c, d) + X[k] + T[i]) <<< s)
  // виконати 16 наступних операцій
  [ABCD  1  5 17]  [DABC  6  9 18]  [CDAB 11 14 19]  [BCDA  0 20 20]
  [ABCD  5  5 21]  [DABC 10  9 22]  [CDAB 15 14 23]  [BCDA  4 20 24]
  [ABCD  9  5 25]  [DABC 14  9 26]  [CDAB  3 14 27]  [BCDA  8 20 28]
  [ABCD 13  5 29]  [DABC  2  9 30]  [CDAB  7 14 31]  [BCDA 12 20 32]
```

```
    // цикл 3
    // нехай [abcd k s i] означають операцію
    //     a = b + ((a + H(b, c, d) + X[k] + T[i]) <<< s)
    // виконати 16 наступних операцій
    [ABCD  5  4 33]  [DABC  8 11 34]  [CDAB 11 16 35]  [BCDA 14 23 36]
    [ABCD  1  4 37]  [DABC  4 11 38]  [CDAB  7 16 39]  [BCDA 10 23 40]
    [ABCD 13  4 41]  [DABC  0 11 42]  [CDAB  3 16 43]  [BCDA  6 23 44]
    [ABCD  9  4 45]  [DABC 12 11 46]  [CDAB 15 16 47]  [BCDA  2 23 48]
    // цикл 4
    // нехай [abcd k s i] означають операцію
    //     a = b + ((a + I(b, c, d) + X[k] + T[i]) <<< s)
    // виконати 16 наступних операцій
    [ABCD  0  6 49]  [DABC  7 10 50]  [CDAB 14 15 51]  [BCDA  5 21 52]
    [ABCD 12  6 53]  [DABC  3 10 54]  [CDAB 10 15 55]  [BCDA  1 21 56]
    [ABCD  8  6 57]  [DABC 15 10 58]  [CDAB  6 15 59]  [BCDA 13 21 60]
    [ABCD  4  6 61]  [DABC 11 10 62]  [CDAB  2 15 63]  [BCDA  9 21 64]

    A += AA
    B += BB
    C += CC
    D += DD
} [1]
```

3. Американський стандарт хеш-функції (SHS або SHA)

Американський стандарт хеш-функції SHS (Secure Hash Standatr або SHA Secure Hash Algorithm) розроблений NIST (National Institute of Standard and Technology) на основі алгоритму MD4 та прийнятий у 1993 році. Призначений для використання сумісно з цифровим підписом, визначений у стандарті DSS. При введені повідомлення M, алгоритм виробляє 160-бітове вихідне повідомлення, що називається згорткою (Message Digest), яке і використовується при створення цифрового підпису.

Розглянемо роботу алгоритму детальніше.

Спочатку вхідне повідомлення доповнюється так, щоб його довжина стала кратна 512 бітам. При цьому повідомлення доповнюється навіть тоді, коли його довжина вже кратна вказаній. Процес відбувається наступним чином: добавляється одиниця, потім стільки нулів, скільки необхідно для отримання повідомлення, довжина якого на 64 біта менша, ніж кратна 512, і потім добавляється 64-бітове представлення довжини вхідного повідомлення.

На наступному кроці ініціалізуються п’ять 32-бітових змінних такими шістнадцятковими константами:

A=67453201

B=EFCDAB89

C=98BADCFE

D=10325476

E=C3D2E1F0

Ці змінні копіюються у нові змінні *a*, *b*, *c*, *d*, *e* – відповідно.

Головний цикл може бути досить просто описаний на псевдокоді таким чином:

For (t=0; t<80; t++) {

temp=(a<<<5)+f_t(b,c,d)+e+W_t+K_t;

e=d; d=c; c=b<<<30; b=a; a=temp;

}

Де <<< - операція циклічного зміщення вліво;

K_t – шістнадцяткові константи, визначаються за наступними формулами:

$$K_t = \begin{cases} 5A827999, & t = 0..19 \\ 6ED9EBA1, & t = 20..39 \\ 8F1BBCDC, & t = 40..59 \\ CA62C1D6, & t = 60..79 \end{cases},$$

функції $f_t(x,y,z)$ задаються виразами:

$$f_t(x,y,z) = \begin{cases} X \wedge Y \vee \neg X \wedge Z, & t = 0..19 \\ X \oplus Y \oplus Z, & t = 20..39,60..79 \\ X \wedge Y \vee X \wedge Z \vee Y \wedge Z, & t = 40..59 \end{cases},$$

значення W_t отримуються із 32-бітових під блоків 512 бітового блоку розширеного повідомлення за правилом:

$$W_t = \begin{cases} M_t, & t = 0..19 \\ (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1, & t = 16..79 \end{cases}.$$

Після закінчення головного циклу значення *a, b, c, d, e* додаються з початковими *A, B, C, D, E* відповідно і здійснюється перехід до обробки наступного 512-бітового блоку розширеного повідомлення. Вихідні значення хеш-функції – це конкатенація значень *A, B, C, D, E.*[13]

Розроблені також модифікації алгоритму SHA: SHA-256, SHA-512 та SHA-384.

(Хеш-функцію ГОСТ Р 34.11-94 розглянути самостійно [13])