Комп'ютерні науки

УДК 004.6

Куц Михайло Сергійович
студент Навчально-наукового комплексу
«Інститут Прикладного Системного Аналізу»
НТУУ "КПІ", Україна, м. Київ

Куц Михаил Сергеевич
студент Учебно-научного комплекса
«Институт Прикладного Системного анализа»
НТУУ "КПИ", Украина, г. Киев

Kuts M.
Student of IASA
National Technical University of Ukraine “KPI”
Kiev, Ukraine

РОЗРОБКА АЛГОРИТМУ ДЛЯ ПРИСКОРЕННЯ ПОРІВНЯННЯ ФАЙЛІВ

РАЗРАБОТКА АЛГОРИТМА ДЛЯ УСКОРЕНИЯ СРАВНЕНИЯ ФАЙЛОВ

ALGORITHM DEVELOPMENT TO ACCELERATE FILE COMPARISON

Анотація: Дана стаття присвячена аналізу алгоритмів хешування та розробці оптимального алгоритму для порівняння файлів, використовуючи хеш-алгоритми.

Ключові слова: хеш, алгоритм, криптографія, функція.

Аннотация: Данная статья посвящена анализу алгоритмов хеширования и разработке оптимального алгоритма для сравнения файлов, используя хэш-алгоритмы.

Ключевые слова: хэш, алгоритм, криптография, функция.

Annotation: This article is devoted to analysis of hashing algorithms and develop optimal algorithm for comparing files using a hash algorithm.

Key words: hash algorithm, cryptography function.

На сьогодні в порівнянні файлів найчастіше застосовують побайтове порівняння їх вмісту. Це найбільш просте та очевидне вирішення задачі порівняння двох файлів, що полягає в проходженні по всім байтам обох файлів і їх порівнянні.

Тому метою моєї роботи був огляд та аналіз можливості використання алгоритмів хешування для прискорення процесу порівняння файлів.

В якості хеш алгоритму для формування хешу файлу мною був взятий SHA1 (Secure Hash Algorithm 1), оскільки він є доволі нескладним в реалізації та забезпечує потрібний рівень унікальності та надійності сформованого хеш-значення.

Короткий опис SHA-1

Це алгоритм криптографічного хешування. Для вхідного повідомлення довільної довжини (максимум 2^{64} біт [2, с. 432-433], що приблизно дорівнює 2 ексабайта), алгоритм генерує 160-бітове хеш-значення, зване також дайджестом повідомлення. Використовується в багатьох криптографічних додатках і протоколах.

SHA-1 реалізує хеш-функцію, побудовану на ідеї функції компресії. Входами функції компресії є блок повідомлення довжиною 512 біт і вихід попереднього блоку повідомлення. Вихід є значення всіх хеш-блоків до цього моменту. Іншими словами $\text{ХешБлоку}_i = f(\text{Блок}_i, \text{ХешБлоку}_{i-1})$ [1, с. 816-817]. Хеш-значенням всього повідомлення є вихід останнього блоку.

Оригінал тексту розбивається на блоки по 512 біт в кожному. Останній блок доповнюється до довжини, кратної 512 біт. Спочатку додається 1 біт, а потім нулі, щоб довжина блоку стала рівною (512 - 64 = 448) біт. В останні 64 біта записується довжина вихідного повідомлення в бітах (в big-endian форматі). Якщо останній блок має довжину понад 448, але менше 512 біт, то додаток виконується в такий спосіб: спочатку додається 1 біт, потім нулі аж до кінця 512-бітного блоку; після цього створюється ще один 512-бітний блок, який заповнюється аж до 448 біт нулями, після чого в останні 64 біта

записується довжина вихідного повідомлення в бітах (в little-endian форматі [3, с. 3-5]). Доповнення останнього блоку здійснюється завжди, навіть якщо повідомлення вже має потрібну довжину.

Ініціалізуються п'ять 32-бітових змінних.

A = a = 0x67452301

B = b = 0xEFCDAB89

C = c = 0x98BADCFE

D = d = 0x10325476

E = e = 0xC3D2E1F0

Визначаються чотири нелінійні операції і чотири константи.

Таблиця 1 – Нелінійні операції і константи

$F_t(m,l,k) = (m \wedge l) \vee (\neg m \wedge k)$	K_t = 0x5A827999	0≤t≤19
$F_t(m,l,k) = m \oplus l \oplus k$	K_t = 0x6ED9EBA1	20≤t≤39
$F_t(m,l,k) = (m \wedge l) \vee (m \wedge k) \vee (l \wedge k)$	K_t = 0x8F1BBCDC	40≤t≤59
$F_t(m,l,k) = m \oplus l \oplus k$	K_t = 0xCA62C1D6	60≤t≤79

Головний цикл ітераційно обробляє кожен 512-бітний блок. Ітерація складається з чотирьох етапів по двадцять операцій в кожному. Блок повідомлення перетвориться з 16 32-бітових слів M_i в 80 32-бітових слів W_j за таким правилом:

$W_t = M_t$ при 0≤t≤15

$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) << 1$ при 16≤t≤79

для t от 0 до 79

```
temp = (a<<5) + F_t(b,c,d) + e + W_t + K_t
e = d
d = c
c = b<<30
b = a
a = temp
```

Рисунок 1 – правило перетворення

Де << - циклічний зсув вліво.

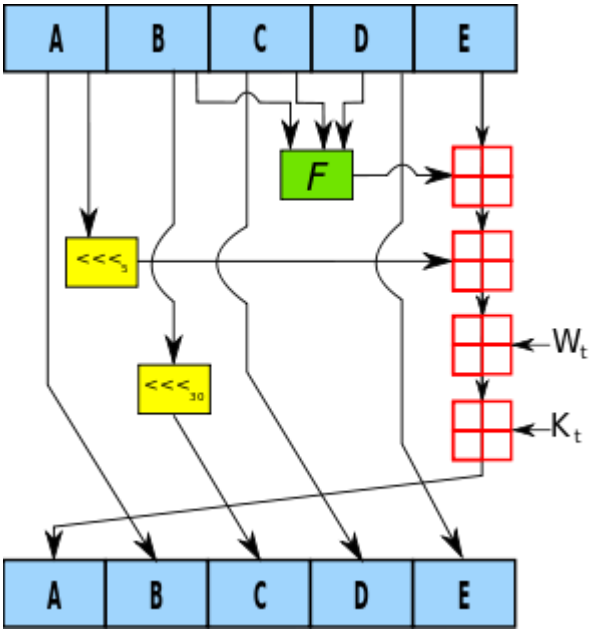

Рисунок 2 – одна ітерація алгоритму SHA1

Після цього a, b, c, d, e додаються до A, B, C, D, E відповідно. Починається наступна ітерація.

Підсумковим значенням буде об'єднання п'яти 32-бітових слів в одне 160-бітове хеш-значення.

Порівняння файлів по хеш-значенню

Алгоритм порівняння файлів полягає в формуванні хеш-значення для кожного з порівнюваних файлів і подальшому порівнянні цих хеш-значень.

В першу чергу хеш-функції є математичною функцією, яка перетворює обсяг даних великого розміру в набагато менший набір даних. Цей набір є відображенням актуальних даних, тому він ідеально підходить для порівняння даних. Шанси генерації двох однакових хешів для різних файлів практично неможливо [4, с. 5-7]. Крихітна зміна в файлі призводить до досить великих і непередбачуваних змін в сформованому хеш.

Таким чином, хеш файлу є хеш-представленням файлу. Хеш на вигляд являє собою довгий числовий або буквено-цифровий вираз, такі як, наприклад:

- b7404b4dd5e4d1b67869226dcbc2da09
- 29-B4-1C-B3-54-F3-14-19-16-EE-0D-6A-F5-73-56-9F-DA-3F-D5-47

Програмна реалізація виглядає наступним чином:

```
public static bool CompareFileHashes(string fileName1, string fileName2)
{
   // Create an instance of System.Security.Cryptography.HashAlgorithm
   HashAlgorithm hash = HashAlgorithm.Create();
   // Declare byte arrays to store our file hashes
   byte[] fileHash1;
   byte[] fileHash2;
   // Open a System.IO.FileStream for each file.
   // Note: With the 'using' keyword the streams
   // are closed automatically.
   using (FileStream fileStream1 = new FileStream(fileName1, FileMode.Open),
                fileStream2 = new FileStream(fileName2, FileMode.Open))
   {
      // Compute file hashes
      fileHash1 = hash.ComputeHash(fileStream1);
      fileHash2 = hash.ComputeHash(fileStream2);
   }
   return BitConverter.ToString(fileHash1) == BitConverter.ToString(fileHash2);
}
```

Порівняння швидкодії алгоритмів порівняння файлів

Таблиця 2 – Порівняння побайтового та хеш алгоритмів

Розмір файла, МБ	Побайтовий алгоритм, с	Хеш алгоритм, с
0.01	0.001	0.002
0.1	0.002	0.003
1	0.021	0.011
10	0.175	0.103
100	1.613	0.988
1000	17.132	10.285

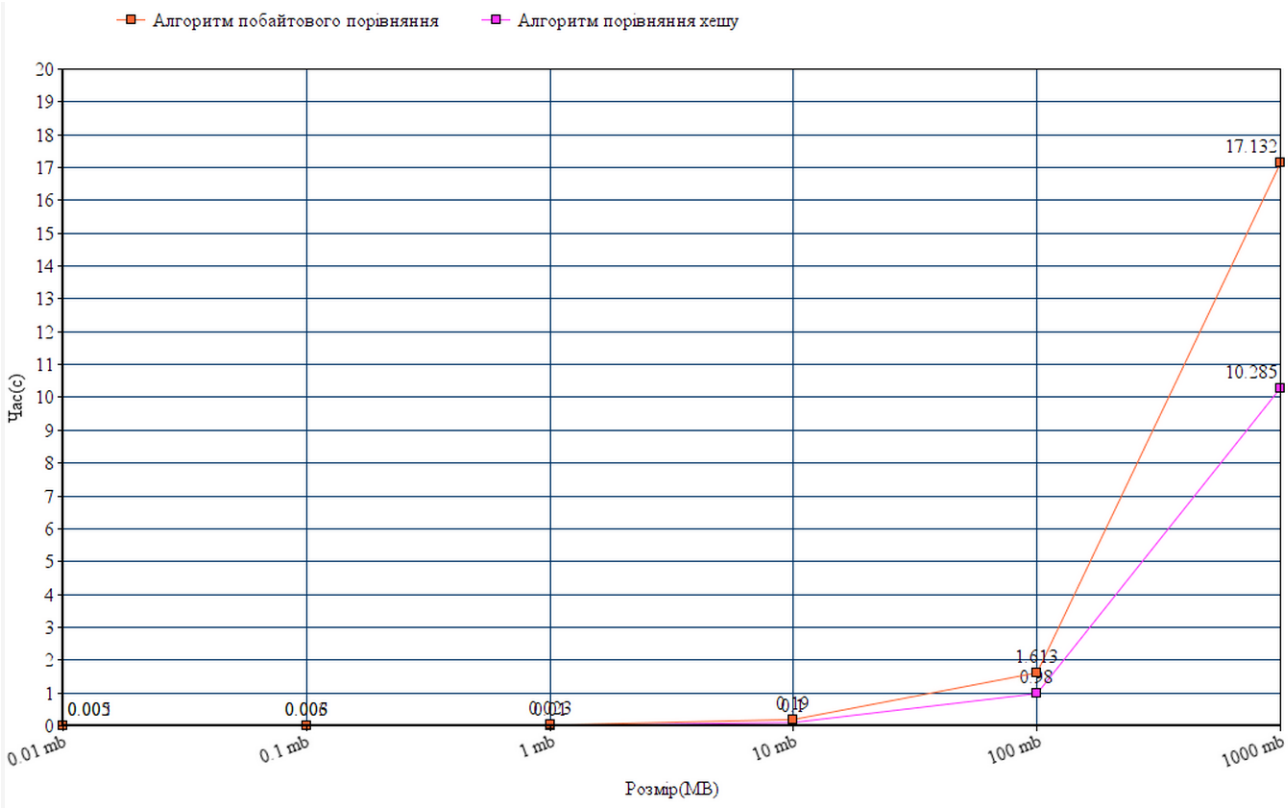

Рисунок 3 – Порівняння побайтового та хеш алгоритмів

Як видно з результатів, немає практично ніякої різниці між методами порівняння для невеликих файлів, але коли було порівняно два великих файли, 100 і більше МБ, можна чітко бачити різницю між методами. Побайтовий методом виконувався близько 17 секунд, щоб завершити порівняння великого файлу, в той час як метод порівняння хеша файлу займає близько 10 секунд.

ЛІТЕРАТУРА

1. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си = Applied Cryptography. Protocols, Algorithms and Source Code in C. — М.: Триумф, 2002. — 816 с.
2. Нильс Фергюсон, Брюс Шнайер. Практическая криптография = Practical Cryptography: Designing and Implementing Secure Cryptographic Systems. — М. : Диалектика, 2004. — 432 с.
3. Finding Collisions in the Full SHA-1 [Електронний ресурс] – Режим доступу:

http://www.infosec.sdu.edu.cn/uploadfile/papers/Finding%20Collisions%20in%20the%20Full%20SHA-1.pdf .

4. Finding SHA-1 Characteristics: General Results and Applications [Електронный ресурс] Режим доступа: http://link.springer.com/chapter/10.1007%2F11935230_1