УДК 681.3.07

Обґрунтування архітектури функції хешування з використанням паралельних обчислень

А. О. Бойко, І. Д. Горбенко
ЗАТ «Інститут інформаційних технологій», Україна

Пропонуються критерії, показники та методика оцінки перспективних функцій гешування з підвищеною швидкодією. Обґрунтований вибір архітектури для проекта національного стандарта.

Ключові слова: *геш-функція, паралельні обчислення.*

Предлагаются критерии, показатели и методика оценки перспективных функций хеширования с повышенным быстродействием. Обоснован выбор архитектуры для проекта национального стандарта.

Ключевые слова: *хеш-функция, параллельные вычисления.*

Proposed criteria, indicators and evaluation techniques of high performance hash functions. The choice of architecture for national standard project was reasoned.

Key words: *hash function, parallel computing.*

Вступ

В таких криптографічних додатках як електронний цифровий підпис [1-3], певні генератори псевдовипадкових послідовностей [4,5] та інших, суттєве значення має складність (швидкість) гешування. Це пояснюється тим, що кожен раз при виконанні та перевірянні цифрового підпису необхідно обчислювати геш значення інформації, що підписується. Також в визнаних перспективними генераторах псевдовипадкових послідовностей [5] необхідно безперервно виконувати гешування певних проміжних значень. В той же час поряд з вимогами відносно забезпечення стійкості до функцій гешування висуваються і вимоги мінімально допустимої складності або максимізації швидкості гешування. В той же час аналіз ряду робіт [6-8] та попередній порівняльний аналіз, що проведений нами, дозволив зробити висновок, що діючий в Україні стандарт гешування ГОСТ 34.311-95 не відповідає вимогам ні по швидкодії, ні по стійкості. Також проміжні результати виконання міжнародного проекту NIST SHA-3 Conpetition [9,10] підтвердили існування проблеми необхідності підвищення швидкодії і відносно існуючих стандартів функцій гешування. Тому, на наш погляд, актуальними є задачі удосконалення функцій гешування по критерію швидкодії, при забезпеченні вимог по стійкості.

Одним з перспективних способів збільшення швидкодії функції гешування є застосування паралельних обчислень.

На основі опублікованих результатів проекту можна виділити два основні підходи реалізації паралельних обчислень:

–використання паралельних обчислень всередині функції стискання;

–виконання паралельних обчислень на рівні побудови архітектури функції гешування.

Перший підхід може бути реалізований за рахунок використання специфічних інструкцій в сучасних процесорах, у зв’язку з чим він є досить обмеженим для застосування. В цій роботі розглядається в якості основного

другий підхід, як більш універсальний і такий, що в перспективі, на наш погляд, дозволить досягнути більшої швидкодії.

Метою цієї статі є визначення перспективної архітектури проекту стандарту функції гешування, яка дозволяє у найбільшій мірі мінімізувати складність обчислення геш-значень, і як наслідок максимізувати швидкість гешування.

1. Основні вимоги, критерії і показники оцінки.

Результати аналізу робіт що опубліковані в ході виконання проекту NIST SHA-3 [9,10] та наші дослідження [6] дозволяють висунути до архітектури перспективної функції гешування такі вимоги:

1) не зменшувати по відношенню до традиційної ітеративної архітектури Меркле-Дамгарда рівень захищеності від існуючих атак;
2) забезпечувати максимально можливий коефіцієнт прискорення обчислення геш - значень;
3) забезпечувати найбільш можливу ефективність використання обчислювальних ядер (з мінімальною кількістю простоїв, синхронізації між потоками виконання і надлишкових обчислень);
4) допускати просту і ефективну реалізацію на різноманітних програмних, програмно-апаратних і апаратних платформах.

Враховуючи рекомендації [11], в якості основних показників, за якими можуть бути оцінені різні архітектури реалізації обчислень для функцій гешування, вибрано коефіцієнт прискорення і ефективність паралельних обчислень.

Під коефіцієнтом прискорення будемо розуміти величину, що показує відношення часу виконання послідовного алгоритму до часу виконання паралельного алгоритма на заданому числі обчислювальних ядер. Коефіцієнт прискорення $K(p)$ обчислюється як

$$K(p) = \frac{T_{посл}}{T_{пар}(p)}, \tag{1}$$

де $T_{посл}$ - час виконання послідовного алгоритма, $T_{пар}(p)$ - час виконання паралельного алгоритму на обчислювальних ядрах, а p - кількість задіяних обчислювальних ядер.

Під ефективністю паралельних обчислень $E(p)$ будемо вважати величину, що показує відношення коефіцієнту прискорення до кількості використаних обчислювальних ядер

$$E(p) = \frac{K(p)}{p}. \tag{2}$$

Також , в якості показника, що характеризує відповідність архітектури вимозі «4», можна використати такий показник як питома витрата пам’яті на одне ядро $RM(p)$. Вона може бути обчисленою з використанням виразу

$$RM(p) = \frac{M(p)}{p}, \tag{3}$$

де $M(p)$ - витрати пам’яті на зберігання всіх проміжних даних алгоритму. Необхідно відмітити, названий показник важливий при оцінці реалізації

алгоритму на пристроях з обмеженою кількістю пам'яті, наприклад мікроконтролерах, ПЛІС тощо.

З урахуванням наведених вище вимог і показників та рекомендацій [11], виберемо для порівняння архітектури функцій гешування і оптимізації параметрів такі критерії:

1) у відповідності з вимогою 1) не повинно існувати відомих криптоаналітичних атак, спрямованих на знаходження колізії, прообразу, другого прообразу та ін. зі складністю меншою, ніж складність відповідних атак грубою силою;
2) згідно з вимогою 2) оптимальна архітектура повинна мати по відношенню до інших найвищій коефіцієнт прискорення;
3) згідно з вимогою 3) оптимальна архітектура повинна мати найвище значення ефективності використання обчислювальних ядер;
4) також, алгоритм, що задовольняє вимозі 4), повинен мати найнижче значення питомих витрат пам'яті.

Необхідно відмітити, що критерій 1) є безумовним, тобто він повинен завжди справджуватись, тоді як інші критерії є умовними, з їх використанням можна робити порівняння архітектури обчислення функції гешування.

2. Опис архітектури основних функцій гешування

Аналіз робіт [10,12,13], дозволяє зробити висновок, що паралельні обчислення можуть бути виконаними при застосуванні архітектури Белларе або деревовидної архітектури.

Також будемо вважати, що критерій 2) має найвищій пріоритет серед трьох умовних критеріїв. Тому порівняння за критеріями «3» і «4» має сенс лише за умови рівнозначності архітектур, що порівнюються, по критерію «2».

Проведений аналіз архітектури Белларе дозволяє зробити висновок, що відносно неї немає суворих доведень стійкості, оскільки в основу доведення покладено твердження про складність вирішення проблеми дискретного логарифма [12], тому не відповідає критерію «1» і не може бути використана для побудування геш-функції.

Відносно деревовидної архітектура необхідно відмітити таке. Стійкість деревовидної архітектури обчислення геш значень доведена в роботі [13]. Деревовидна архітектура характеризується двома основними параметрами k - висота дерева і N - арність дерева.

В роботі [13] показано, що коефіцієнт прискорення K для такої архітектури становить

$$K \approx N^k, \tag{4}$$

але за умови, що кількість обчислювальних ядер достатня. В указаній роботі запропонований алгоритм симулювання (N,k) - дерева на $N^{k'}, k' < k$ обчислювальних ядрах з коефіцієнтом прискорення

$$K' \approx N^{k'}. \tag{5}$$

Також у вказаній роботі показано, що ефективно можуть бути використані лише $N^{\lfloor \log_N p \rfloor}$ обчислювальних ядер, що утворюють $(N, \lfloor \log_N p \rfloor)$-піддерево

(N,k)-дерева. Всі інші будуть простоювати. Тому функція коефіцієнту прискорення від кількості обчислювальних ядер має вигляд

$$K(p) = N^{\min(\lfloor \log_N p \rfloor, k)}. \tag{6}$$

З урахуванням (2) та (6) одержимо формулу для обчислення ефективності використання паралельних обчислень

$$E(p) = \frac{K(p)}{p} = \frac{N^{\min(\lfloor \log_N p \rfloor, k)}}{p}. \tag{7}$$

Для визначення обсягу пам'яті, необхідної для реалізації деревовидної архітектури, будемо враховувати таке. Нехай на один виклик функції стискання необхідно виділити один блок пам'яті певного фіксованого розміру. Рекурсивний алгоритм обходу дерева на одному обчислювальному ядрі дозволяє обійтися k блоками для зберігання проміжних значень з розрахунку один блок на один виклик функції. Приклад показано на рисунку 1.

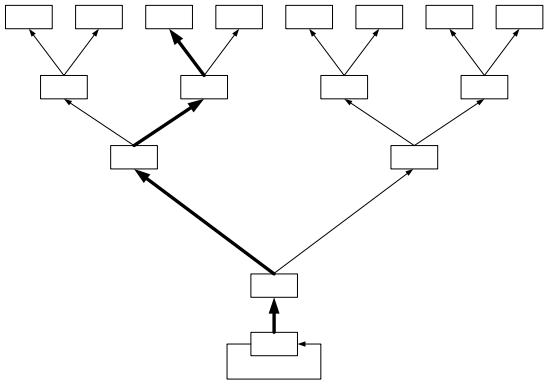

Рис. 1. Схема рекурсивного обходу дерева

Якщо (N,k)-дерево реалізується за допомогою (N,k')-піддерева, то рекурсивний алгоритм виконає $\lceil k/k' \rceil$ викликів, причому на кожен з них витратить $N^{k'}$ блоків пам'яті. Приклад показано на рисунку 2. Пунктирний прямокутник на рисунку означає паралельне виконання обох функцій стискання. Тому загальна кількість використаної пам'яті дорівнюватиме

$$M(k') = \begin{cases} k, k' = 0 \\ \left\lceil \frac{k}{k'} \right\rceil N^{k'}, k' > 0 \end{cases} \tag{8}$$

блоків, де k - висота дерева, k' - висота піддерева і N - арність дерева.

В свою чергу k' визначається з кількості обчислювальних ядер (для $p > 1$) як

$$k' = \lfloor \log_N p \rfloor, \tag{9}$$

де p - кількість наявних обчислювальних ядер.

Отже загальна формула для обчислення витрат пам'яті (в блоках) на збереження проміжних даних має вигляд

$$M(p)=\begin{cases}k, p=1\\ \left\lceil \dfrac{k}{\min\left(k,\lfloor \log_N p \rfloor\right)} \right\rceil N^{\min\left(k,\lfloor \log_N p \rfloor\right)}, p>1\end{cases}, \tag{10}$$

Використовуючи (10), отримаємо формулу для обчислення питомої витрата пам'яті

$$RM(p)=\frac{M(p)}{p}=\begin{cases}k, p=1\\ \dfrac{1}{p}\left\lceil \dfrac{k}{\min\left(k,\lfloor \log_N p \rfloor\right)} \right\rceil N^{\min\left(k,\lfloor \log_N p \rfloor\right)}, p>1\end{cases}, \tag{11}$$

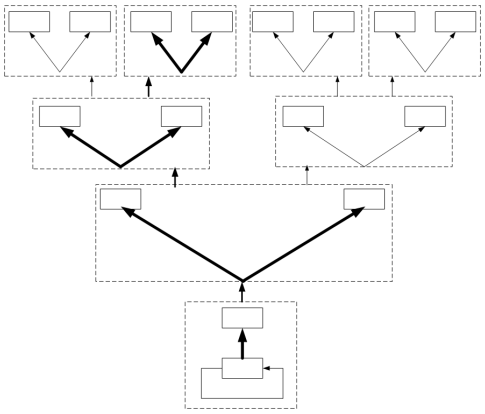

Рис. 2. Симуляція дерева за допомогою під дерева меншої висоти

При дослідженні деревовидної архітектури слід звернути особливу увагу на один частковий випадок, який можна описати як $(N,1)$-дерево або як $(1,k)$-дерево. Схема для цього випадку наведена на рисунку 3.

Опису властивостей архітектури з такими параметрами у доступній літературі не знайдено. Приймаючи висоту дерева $k=1$, виначимо, що коефіцієнт прискорення $K(p)$ відповідно до (6)

$$K(p)\approx p\,. \tag{12}$$

Тому для цього випадку ефективність обчислень $E(p)$ визначається як

$$E(p)=\frac{K(p)}{p}\approx\frac{p}{p}=1\,. \tag{13}$$

Необхідний обсяг пам'яті можна визначимо таким чином. Нехай на один виклик функції стискання виділяється один блок пам'яті. З рисунку 4 видно, що для кожного моменту часу використовується $M(p)=p$ блоків пам'яті – по одному на кожне обчислювальне ядро. Тому також

$$RM(p)=p/p=1\,. \tag{14}$$

Цей результат повністю відповідає результату, обчисленому по формулі (11), у випадку, якщо розглядати $(N,1)$ - дерево на одному обчислювальному ядрі або як $(1,k)$ - дерево на p ядрах (якщо умовно прийняти, що $\log_1 p=\infty$).

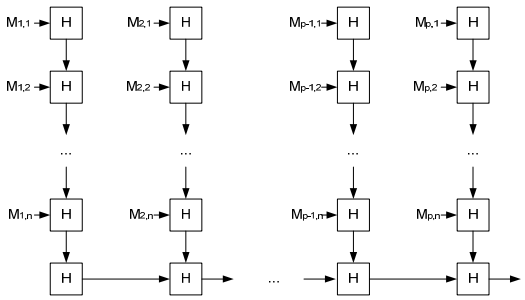

Рис. 3. Схема часткового випадку деревовидної архітектури

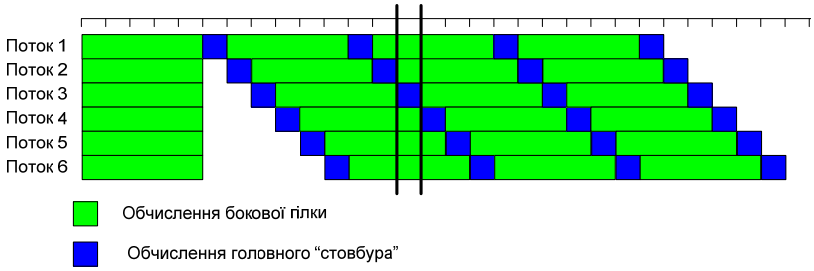

Рис. 4. Діаграма часу виконання

3. Порівняння

Зробимо порівняння використовуючи дані, що наведені на рис. 5, 6, 7, що ілюструють поведінку обраних показників при зміні кількості обчислювальних ядер. По коефіцієнту прискорення (рисунок 5) частковий випадок повністю завантажує обчислювальні ядра, тоді як в загальному випадку частина ядер буде простоювати. За рахунок цього ефективність використання паралельних обчислень у випадку використання часткового випадку дерева завжди найвища (рисунок 6) і дорівнює одиниці, тоді як в загальному випадку вона значно менша одиниці. Питома витрата пам'яті для часткового випадку завжди однакова і порівняно невелика, в той час як для реалізації загального випадку необхідна найбільша кількість пам'яті саме при реалізації на одному обчислювальному ядрі. Як видно, частковий випадок (1,32)-дерева має найкращі властивості.

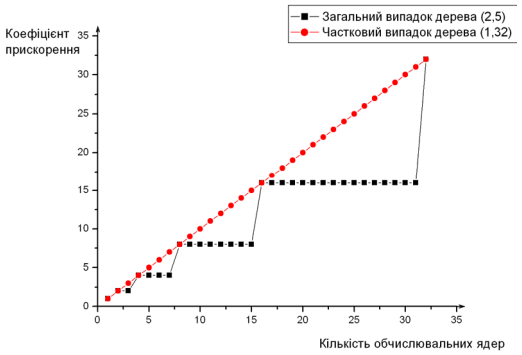

Рис. 5. Графік залежності коефіцієнту прискорення від кількості обчислювальних ядер

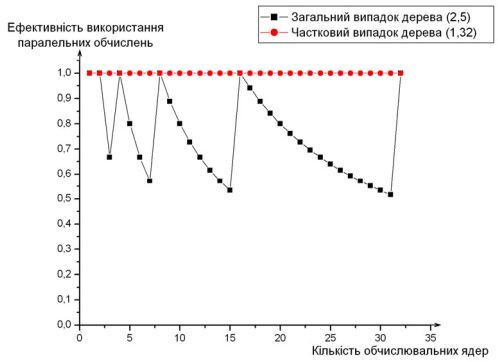

Рис. 6. Графік залежності ефективності використання паралельних обчислень від кількості обчислювальних ядер

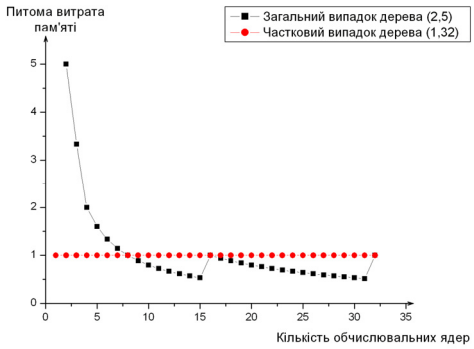

Рис. 7. Графік залежності питомої витрати пам'яті від кількості обчислювальних ядер

Висновки

Порівнюючи отримані результати, можливо зробити висновок, що частковий випадок деревовидної архітектури, зображений на рисунку 3 за умови незмінності коефіцієнту прискорення, забезпечує найкращі показники ефективності використання паралельних обчислень і питомих витрат пам'яті. Отже він забезпечує найбільш ефективну реалізацію на різноманітніших типах обчислювальних пристроїв та рекомендується до застосування.

ЛІТЕРАТУРА

1. ДСТУ 4145-2002. Державний стандарт України. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка. – Введ. 2002 р.– К. – Держстандарт України, 2003 – 33 с.
2. ISO/IEC 9796-3 Information technology — Security techniques — Digital signature schemes giving message recovery — Part 3: Discrete logarithm based mechanisms . – Введ. 2006 – ISO/IEC
3. FIPS 186-3 Digital Signature Standard (DSS). – Введ. 2009 – NIST
4. X9- 63 Public Key Cryptography for the Financial Services Industry, Key Agreement and Key Transport Using Elliptic Curve Cryptography Введ. 2001 – ANSI

5. ISO/IEC 18031 Information technology — Security techniques — Random bit generation. – Введ. 2005 – ISO/IEC
6. Горбенко І. Д., Бойко А. О. Сучасні підходи до побудування геш-функцій з підвищеною стійкістю. Праці міжнародного симпозіуму «Питання оптимізації обчислень (ПОО-XXXV)», присвяченого 40-річчю І Симпозіуму та літньої математичної школи з питань точності й ефективності обчислювальних алгоритмів (м. Київ, м. Одеса; 1969 рік). – Київ: Інститут кібернетики імені В. М. Глушкова НАН України, 2009. Т. 1.–447 с.; ст.ст. 159-163
7. Florian Mendel, Norbert Pramstaller, Christian Rechberger, Marcin Kontak, and Janusz Szmidt. Cryptanalysis of the GOST Hash Function. – Режим доступу: http://wiki.uni.lu/esc/docs/mendel_gost.pdf. – Назва з екрану.
8. Praveen Gauravaram, John Kelsey - Linear-XOR and Additive Checksums Don't Protect Damgaard-Merkle Hashes from Generic Attacks In Proceedings of CT-RSA, LNCS 4964, pp. 36-51, Springer, 2008
9. Ewan Fleischmann and Christian Forler and Michael Gorski. Classification of the SHA-3 Candidates – Режим доступу:http://eprint.iacr.org/2008/511. – Назва з екрану.
10. NISTIR 7620: Status Report on the First Round of the SHA-3 Cryptographic Hash Algorithm Competition – Режим доступу: http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/sha3_NISTIR7620.pdf . – Назва з екрану.
11. Введение в алгоритмы параллельных вычислений / Молчанов И. Н.; Отв. ред. Яковлев М. Ф.; АН УССР. Ин-т кибернетики им. В. М. Глушкова. – Киев: Наук. думка, 1990, стр. 7
12. Mihir Bellare, Daniele Micciancio A New Paradigm for Collision-free Hashing: Incrementality at Reduced Cost – Режим доступу: http://groups.csail.mit.edu/cis/incremental/inchash.pdf. – Назва з екрану.
13. Palash Sarkar, Paul J. Schelenberg A Parallelizable Design Principles for Cryptographic Hash Functions – Режим доступу: http://eprint.iacr.org/2002/031.ps. – Назва з екрану.

Надійшла 25.03.2010.