The Accelerated Data Hashing Method

Volodymyr Luzhetskyi
Department of Information Protection
Vinnytsia National Technical University
Vinnytsia, Ukraine
v.luzhetskyi@vntu.edu.ua

Dmytro Rohachevskyi
Department of Information Protection
Vinnytsia National Technical University
Vinnytsia, Ukraine
dimonrogach@gmail.com

Volodymyr Kozyra
Department of Information Protection
Vinnytsia National Technical University
Vinnytsia, Ukraine
vovakozira@gmail.com

Метод Пришвидшеного Гешування Даних

Володимир Лужецький
Кафедра захисту інформації
Вінницький національний технічний університет
Вінниця, Україна
v.luzhetskyi@vntu.edu.ua

Дмитро Рогачевський
Кафедра захисту інформації
Вінницький національний технічний університет
Вінниця, Україна
dimonrogach@gmail.com

Володимир Козира
Кафедра захисту інформації
Вінницький національний технічний університет
Вінниця, Україна
vovakozira@gmail.com

***Abstract* — The method of accelerated hashing of data on the basis of building a tree of intermediate hash values using parallelized calculations on the microprocessor cores is proposed. The message is divided into blocks of the same length, and the hash value is calculated by constructing a tree of intermediate results, which allows achieving an acceleration of the hashing process compared to the classical iteration method.**

***Анотація* — Запропоновано метод пришвидшеного гешування даних на основі побудови дерева проміжних геш-значень з використанням розпаралелених обчислень на ядрах мікропроцесора. Повідомлення розбивається на блоки однакової довжини, а геш-значення обчислюється шляхом побудови дерева проміжних результатів, що дозволяє досягти пришвидшення процесу гешування порівняно з класичним ітераційним методом.**

Keywords — hashing, hash tree, parallelization of calculations.

Ключові слова — гешування, дерево геш-значень, розпаралелення обчислень.

1. Вступ

Одним із розділів комп'ютерної криптографії є гешування даних та його використання для розв'язання задач захисту інформації. Зокрема, методи гешування використовуються для розв'язання задач автентифікації та авторизації користувачів, автентифікації даних, перевірки цілісності даних, створення послідовностей псевдовипадкових чисел та генерування сеансових ключів.

Авторизація користувачів комп'ютерної системи та розмежування їх прав доступу до ресурсів системи забезпечує виконання політики інформаційної безпеки. Останнє дозволяє зменшити збої у роботі комп'ютерної системи та забезпечити цілісність, конфіденційність та вірогідність даних, що обробляються в системі [1].

Здатність комп'ютерних систем опиратися атакам зловмисників на паролі визначається довжиною використовуваних паролів, їх унікальністю, множиною символів, що можуть використовуватися при введенні пароля [2]. Для того, щоб протидіяти загрозі зі сторони зловмисника, у базі даних замість паролів зберігають їх геш-значення. Завдяки тому, що для відомого геш-значення

практично складно знайти повідомлення, якому відповідає це геш-значення, зловмисник, отримавши доступ до бази даних, не можу порушити прав доступу [3].

Для забезпечення автентичності даних та надання інформації на електронних носіях юридичної значимості використовуються цифрові підписи, елементом яких є геш-значення даних, що підписуються. Відповідно у задачах, коли підписуються великі файли, підписувати безпосередньо дані є недоцільним, натомість замість них підписують їх геш-значення [4]. Крім зменшення часу, що витрачається на підписування, використання геш-значень дозволяє підписувати секретні дані без їх розголошення, оскільки сторона, що підписує, має змогу підписати документ (повідомлення), залишаючи в секретності відомості, що зберігаються у ньому, а натомість можна опублікувати у спеціальній базі даних геш-значення цього документа.

Під час функціонування комп'ютерних систем виникає задача забезпечення цілісності інформації, що зберігається та циркулює у них. Для розв'язання цієї задачі використовують коди автентичності повідомлення – MAC (message authentication code), та функції їх обчислення. Один з підходів до генерування MAC базується на використанні ключових геш-функцій. Гешування відбувається для повідомлення, що створене внаслідок конкатенації ключа та повідомлення, що захищається, відповідно воно може бути $k \parallel M$, $M \parallel k$ або $k \parallel M \parallel k$ [4].

При передаванні даних комп'ютерною мережею часто виникає задача автентифікації користувачів для того, щоб сторони обміну мали змогу переконатися у тому, що зловмисник не видає себе за іншу авторизовану сторону. Для розв'язання таких задач використовують криптографічні протоколи автентифікації, до яких зокрема відносяться і протоколи, що використовують гешування. Такі протоколи автентифікації використовують випадкові (псевдовипадкові) числа або мітки часу для надання унікальності повідомленням, що пересилаються комп'ютерною мережею від одного користувача до іншого [5].

Оскільки результатом гешування є геш-значення, що мають рівномірний закон розподілу, то його часто використовують для генерування псевдовипадкових послідовностей.

Метою роботи є пришвидшення процесу гешування даних.

2. Конструкція Меркля-Дамгарда послідовного процесу гешування

Найпоширенішою схемою процесу криптографічного гешування є конструкція Меркля-Дамгарда, яка є основою таких відомих геш-функцій як MD5, SHA-1, RIPEMD-160 SHA-256 та інших. Основним перетворенням, що реалізується, є функція ущільнення, яка забезпечує формування геш-значень фіксованої довжини [6]. При цьому процес гешування є ітеративним і принципово реалізується як послідовний процес (рис.1).

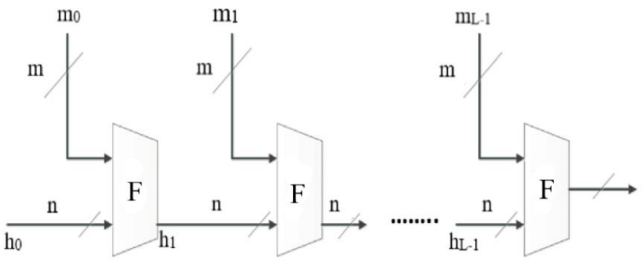

Рис. 1. Конструкція Меркля-Дамгарда ітеративного процесу гешування

Однією з перших математичних моделей гешування стала конструкція Меркля-Дамгарда. Ця конструкція передбачає розбиття повідомлення M, що гешується, на l частин – блоків даних $m_i (i = \overline{1, l})$.

Відповідно до цієї конструкції, всі блоки m_i повинні мати однакову сталу довжину m, а геш-значення – довжину n. У випадку, коли довжина повідомлення M не є кратною довжині блоку, це повідомлення доповнюється бітами таким чином, щоб довжина останнього блоку була m. Гешування реалізується як ітераційний процес:

$$h_i = f(h_{i-1}, m_i),$$

де h_i – проміжне геш-значення, отримане на i-му кроці;

$f(\cdot)$ – функція ущільнення, що забезпечує фіксовану довжину геш-значень та їх рівномірний розподіл.

Як вихідне геш-значення згідно з конструкцією Меркля-Дамгарда, використовується h_l а початкове заповнення, або, як його ще називають, вектор ініціалізації h_0 використовується як ключ, або як його частина.

Незважаючи на широке використання даної конструкції для розв'язання задач захисту інформації, вона має один вагомий недолік у вигляді низької швидкодії через ітераційний процес гешування [7]. Тому для пришвидшення цього процесу запропоновано метод гешування даних, який передбачає розпаралелення обчислень на кількох ядрах.

3. Метод пришвидшеного гешування даних

Особливість сучасних мікропроцесорів полягає в тому, що вони містять 4 або 8 ядер, які можуть обробляти інформацію одночасно, що надає можливості розпаралелювати процес обробки. Саме це спонукає шукати інші конструкції гешування, які передбачають розпаралелення процесів обчислень.

Метод, що пропонується, реалізує процес побудови дерева проміжних геш-значень. Повідомлення M, що підлягає гешуванню, розбивається на блоки однакової довжини, яка дорівнює довжині геш-значення.

Нехай $M = m_1, m_2, \ldots, m_n$. Якщо кількість блоків є непарною, то повідомлення доповнюється одним блоком: $M^* = m_1, m_2, \ldots, m_n, m_{n+1}$. В блоці m_{n+1} вказується код кількості блоків повідомлення M. Коли кількість блоків парна, то повідомлення доповнюється двома блоками: $M^* = m_1, m_2, \ldots, m_n, m_{n+1}, m_{n+2}$. Тут блок m_{n+1} може мати будь-який

код, а блок m_{n+2} містить код кількості блоків повідомлення M.

Елементи дерева проміжних геш-значень обчислюються таким чином.

Перший рівень:

$h_1^{(1)} = f(m_1,m_2)$, $h_2^{(1)} = f(m_3,m_4)$, …, $h_k^{(1)} = f(m_{n-1},m_n)$ для n парного і $h_k^{(1)} = f(m_n,m_{n+1})$ для n непарного.

Другий рівень:

$h_1^{(2)} = f(h_1^{(1)}, h_2^{(1)})$, $h_2^{(2)} = f(h_3^{(1)}, h_4^{(1)})$, …, $h_k^{(2)} = f(h_{k-1}^{(1)}, h_k^{(1)})$.

Рівень $log_2 (n)$ дає остаточне геш-значення.

Кількість кроків обчислення геш-значення повідомлення, що складається з n блоків, з використанням 4 ядер визначається за формулою:

$$C_4 = \left(\sum_{i=1}^{log_2(n)-2} \frac{n}{8i}\right) + 2.$$

Для 8 ядер кількість кроків обчислення визначається за формулою:

$$C_8 = \left(\sum_{i=1}^{log_2(n)-2} \frac{n}{16i}\right) + 2.$$

Оскільки ітераційна конструкція Меркля-Дамгарда принципово реалізується на одному ядрі, то кількість кроків обчислень дорівнює n.

Найбільша ефективність досягається для $n = 8 \times l$ або $n = 16 \times l$ $(l > 1)$.

Приклад побудови дерева для повідомлення, що складається з 8 блоків наведено на рис. 2.

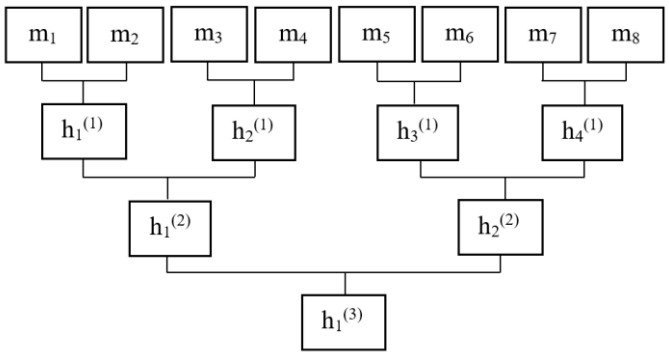

Рис. 2.Приклад дерева для 8 блоків повідомлення

Нехай процесор має 4 ядра, тоді розпаралелення процесу гешування відбувається за схемою, що подана у вигляді таб. 1.

ТАБЛИЦЯ I. ТАБЛИЦЯ I. СХЕМА ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ.

Етап	Ядро 1	Ядро 2	Ядро 3	Ядро 4
1	$h_1^{(1)} = f(m_1,m_2)$	$h_2^{(1)} = f(m_3,m_4)$	$h_3^{(1)} = f(m_5,m_6)$	$h_4^{(1)} = f(m_7,m_8)$
2	$h_1^{(2)} = f(h_1^{(1)}, h_2^{(1)})$	$h_2^{(2)} = f(h_3^{(1)}, h_4^{(1)})$		
3	$h_1^{(3)} = f(h_1^{(2)}, h_2^{(2)})$			

Тобто, за рахунок розпаралелення побудоване дерево матиме швидше виконання на відміну від класичного, ітераційного методу.

Однією з умов реалізації запропонованого методу пришвидшеного гешування є використання блоків однакової довжини, яка дорівнює довжині геш-значення. Цю умову можна задовольнити використовуючи функцію ущільнення, що базується на множенні даних, поданих у вигляді кватерніонів [8].

4. Висновки

Основна відмінність запропонованого методу пришвидшеного гешування від класичного ітераційного методу Меркля-Дамгарда полягає в тому, що процес гешування подається в вигляді дерева проміжних результатів з використанням обчислень на кількох ядрах мікропроцесора.

Пришвидшення процесу гешування тим більше, чим більше використовуваних ядер і блоків повідомлення, що підлягає гешуванню.

Література References

[1] B. Schneier, Secrets & Lies. Digital Security in a Networked World. John Wiley & Sons, 2000.

[2] S. Burnett and S. Paine, “RSA Security's Official Guide To Cryptography”, 2001. [Online]. Available: https://archive.org/details/0355-pdf-burnett-rsa-securitys-official-guide-to-cryptography-s.-burnett-s.-paine-2001-ww/mode/2up [Accessed: 06-May-2024].

[3] B. Schneier, Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C. John Wiley & Sons, 1996.

[4] N. Ferguson and B. Schneier, Practical Cryptography. John Wiley & Sons, 2003.

[5] P. Gauravaram, “Cryptographic Hash Functions: Cryptanalysis, Design and Applications”, Ph.D. dissertation, Queensland University of Technology, 2009. [Online]. Available: http://eprints.qut.edu.au/16372/1/Praveen_Gauravaram_Thesis.pdf [Accessed: 06-May-2024].

[6] “Merkle-Damgard Scheme in Cryptography”, GeeksforGeeks, [Online]. Available: https://www.geeksforgeeks.org/merkle-damgard-scheme-in-cryptography/ [Accessed: 08-May-2024].

[7] Merkle-Damgard Scheme in Cryptography – https://www.geeksforgeeks.org/merkle-damgard-scheme-in-cryptography/ [Accessed: 08-May-2024].

[8] В. А. Лужецький та Ю. В. Баришев, “ПІДХІД ДО ПАРАЛЕЛЬНОГО ГЕШУВАННЯ ДАНИХ НА ОСНОВІ МОДЕЛІ КВАТЕРНІОНА”, у ЗАХИСТ ІНФОРМАЦІЇ І БЕЗПЕКА ІНФОРМ. СИСТЕМ, Львів, Україна, 25–26 трав. 2023. Львів: Вид-во Львів. політехніки, 2023, с. 81–82.