Метод байт-орієнтованого хешування

Лужецький В.А.[1], Кисюк Д.В.[2], Савицька Л. А.[3]

[1]Проф., д.т.н., завідувач кафедри захисту інформації, Вінницький національний технічний університет
вул. Хмельницьке шосе 95, м. Вінниця, Україна, lva_zi@mail.ru
[2]Асист. кафедри обчислювальної техніки, Вінницький національний технічний університет
вул. Хмельницьке шосе 95, м. Вінниця, Україна, kneimad@gmail.com
[3]К.т.н., ст. викл. кафедри обчислювальної техніки, Вінницький національний технічний університет
вул. Хмельницьке шосе 95, м. Вінниця, Україна, lyudik0304@gmail.com

***Анотація*—Розглянуто основні принципи побайтової обробки вхідного повідомлення з виділенням його характеристичних ознак. Запропоновано принципово новий підхід до побудови хеш-функцій без використання ітеративної процедури на основі побайтових операцій. Наведено математичну модель запропонованого методу.**

Ключові слова: хешування, гешування, хеш-функція, неітеративне хешування, побайтова обробка даних.

Method of byte-oriented hashing

Luzhetskyi V.A.[1], Kysiuk D.V.[2], Savyutska L. A.[3]

[1]Prof., Head of Department ofInformation Protection,Vinnytsia National Technical University
Khmelnytske shose str., 95, Vinnytsia, Ukraine,lva_zi@mail.ru
[2]Asist., Department of Computer Science, Vinnytsia National Technical University
Khmelnytske shose str.,95, Vinnytsia, Ukraine, kneimad@gmail.com
[3]Ph.D., Senior Lecturer , Department of Computer Science, Vinnytsia National Technical University,
Khmelnytske shose str., 95, Vinnytsia, Ukraine, lyudik0304@gmail.com

***Abstract*—Basic methods of byte-oriented data carrying wasconsidered. Theadvantages and disadvantages of these methods also was analyzed.Authors offered the new byte-oriented method of non-iteration hash-function creation. The mathematical and schematic models of this method were presented.**

Keywords: hashing,hash - function, noniterrate hashing, byte-oriented hashing.

Вступ

Криптографічні хеш-функції є одними з найважливіших видів криптографічних перетворень. Вони широко застосовуються у задачах криптографічного захисту інформації. Натепер існує велика кількість різноманітних хеш-функцій. Проте, зростаючі вимоги, що висуваються до швидкості хешування даних, а також необхідність реалізації у пристроях з невеликими обчислювальними можливостями, приводять до необхідності розробки нових методів хешування, з можливою їх спеціалізацією для певних пристроїв чи повідомлень особливого виду [1, 2].

Відомі хеш-функції передбачають реалізацію у вигляді ітераційних процедур. Проте, у зв'язку з тим, що питання про лавиноподібний ефект з початковим заповненням при великій кількості ітерацій недостатньо досліджений, і, відповідно, використання цих функцій є недостатньо обгрунтованим. Також, до недоліків слід віднести такі їх особливості [3, 6]:

1) різний вплив блоків даних на остаточний результат хешування (значення першого блоку бере участь у формуванні усіх проміжних хеш-значень через ітеративність процедури, а значення останнього блоку враховується лише на останній ітерації);

2) існує потенційна можливість за результатами кожної ітерації відновити блок даних і попереднє хеш-значення, тому зазвичай в даних методах намагаються ускладнити процедуру такого відновлення за рахунок ускладнення обчислень на кожній ітерації [4, 5].

Для усунення вказаних недоліків автори пропонують принципово новий підхід до побудови хеш-функцій.

Метод хешування на основі характеристчних ознак байтової стурктури даних

Вхідне повідомлення M розбивається на послідовність байтів:

$$M = \{ m_1, m_2, \ldots, m_L \}.$$

Кожен байт розглядається як число n, що відповідає ASCII – коду символу представленого байтом m_l ($l = 1 \div L$), тобто $n = f(m_l)$.

Повідомлення характеризується кількістю елементів k_n, що мають числовий еквівалент n ($n = 0 \div 255$) та номерами позицій у яких розташовані ці елементи.

На основі цих характеристик утворюється два масиви $\boldsymbol{K}$ та $\boldsymbol{S}$:

$$\boldsymbol{K} = (k_0, k_1, \dots, k_{255}),$$
$$\boldsymbol{S} = (s_0, s_1, \dots, s_{255}),$$

де: k_n – кількість байтів, що мають числовий еквівалент n;

s_n – сума номерів позицій, на яких розташований байт з числовий еквівалент n;

$$s_n = (\sum_{j=1}^{k_n} l_j^n)\ mod\ 2^8 . s_n = \left(\sum_{j=1}^{k_n} l_j^n\right) mod 2^8$$

Узагальнена схема процесу хешування наведена на рис. 1.

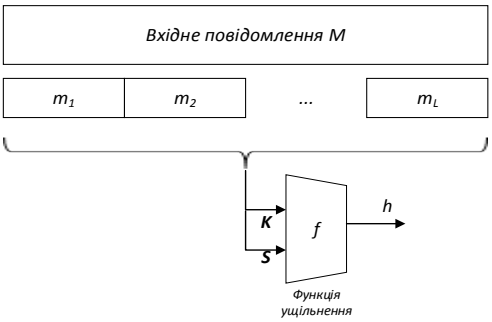

Рис. 1 – Узагальнена схема процесу хешування

Для підрахунку хеш-коду, необхідно виконати такі операцій над масивами***K*** та ***S***:

1) додати побайтно елементи масивів ***K***та ***S***:

$$q_i = (k_i + s_i)\ mod\ 2^8, i = 0 \div 255;$$

2) записати 256 елементів результуючого масиву у вигляді матриці 8×32 елементи:

$$Q = \begin{pmatrix} q_0 & \dots & q_{31} \\ q_{32} & \dots & q_{63} \\ \dots & & \dots \\ q_{224} & \dots & q_{255} \end{pmatrix};$$

3) порахувати суми для кожного з восьми утворених рядків у межах байта:

$$str_0 = (q_0 + q_1 + \dots + q_{31})\ mod\ 2^8\ ;$$
$$str_1 = (q_{32} + q_{33} + \dots + q_{63})\ mod\ 2^8;$$
$$\dots$$
$$str_7 = (q_{224} + q_{225} + \dots + q_{255})\ mod\ 2^8;$$

4) утворити нову матрицю Q^*, збільшивши кожен елемент матириці Q на відповідну величину str_i:

$$Q^* = \begin{pmatrix} q_0^* & \dots & q_{31}^* \\ q_{32}^* & \dots & q_{63}^* \\ \dots & & \dots \\ q_{224}^* & \dots & q_{255}^* \end{pmatrix},$$

де $q_i^* = (q_i + str_j)\ mod\ 2^8, j = \left]\frac{i}{32}\right[$;

5) підрахувати хеш-значення як суми кожного з 32-х рядків матриці Q^*:

$$H = (h_0, h_1, \dots, h_{31}),$$

де $h_n = (\sum_{m=0}^{7} q_{32m}^* + n)\ mod\ 2^8, n = 0 \div 31.$

Таким чином, функція ущільнення f описується послідовністю операцій, що показана на рис. 2.

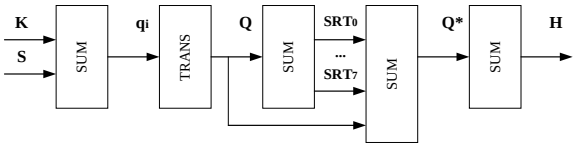

Рис. 2 – Схема реалізації функції ущільнення

Для реалізації запропонованого методу необхідно виконати таку кількість операцій. Вважаємо, що L - довжина вхідного повідомлення. Для формування масивів ***K*** та ***S*** потрібно виконати L зчитувань, L додавань та L записів.

Обчислення хеш-значення H вимагає виконання 256 зчитувань, додавань та записів для формування масиву ***Q***. Для формування значень сум $str_0 - str_7$ потрібно 256 зчитувань і додавань та 8 записів. Формування масиву ***Q**** вимагає виконання 264 зчитувань, 256 додавань та 32 записи. Тобто, всього 2608 операцій.

Загальна кількість операцій дорівнює $3L$+2608. І чим більша довжина повідомлення, тим ближче до 3 оцінка кількості операцій потрібних для обробки 1 байта.

Для порівняння метод хешування BLAKE вимагає близько 65 операцій на обробку 1 байта.

Для апаратної реалізації запропонованого методу хешування потрібно 3 восьмирозрядних лічильники, 2 восьмирозрядних суматори та 2 оперативних запам'ятовувальних пристрої обсягом 256 байт кожен.

ВИСНОВКИ

Запропонований метод хешування не передбачає ітеративної процедури обчислення хеш-значення і,як наслідок, до нього не можуть бути застосовані відомі атаки на ітеративні хеш-функції.

Використання характеристичних ознак вхідних даних та їх побайтна обробка значно прискорюють процес обчислення хеш-значення, а також зменшують апаратні витрати на реалізацію такого методу.

ЛІТЕРАТУРА REFERENCES

[1] Лужецький В. А. Новий підхід до побудови криптографічних хеш-функцій / В. А. Лужецький, Д. В. Кисюк // «Інформаційні технології та комп'ютерна інженерія»; матеріали статей п'ятої міжнародної науково-практичної конференції, м. Івано-Франківськ, 27-29 травня 2015 року. – Івано-Франківськ: Супрун В. П., 2015 р. – с. 206-208..

[2] Лужецький В. А. Узагальнений метод хешування байтової форми представлення інформації / В. А. Лужецький, Д. В. Кисюк // IV міжнародна науково-практична конференція «Інформаційні технології та комп'ютерна інженерія». – Вінниця: ВНТУ, 2014., -275с.

[3] Aumasson J. P. SHA-3 proposal BLAKE / Henzen L., Meier W., Phan R. - 2010.

[4] Bos J. W. Performance analysis of the SHA-3 candidates on exotic multi-core architectures / J. W. Bos, D. Stefan. - 2010.

[5] Neves S. ChaCha implementation. - 2009. – Режим доступу до статті: http://eden.dei.uc.pt/sneves/chacha/chacha.html.

[6] Knezevic M. Fair and consistent hardware evaluation of fourteen round two SHA-3 candidates / M. Knezevic, K. Kobayashi, J. Ikegami, S. Matsuo, A. Satoh, U. Kocabas, J. Fan // April 2011