УДК 004.056 : 004.424.47

МЕТОДИ ЗАВ'ЯЗУВАННЯ ДАНИХ ДЛЯ ПІДВИЩЕННЯ СТІЙКОСТІ ГЕШУВАННЯ ДО ЗАГАЛЬНИХ АТАК

Баришев Ю.В.,
к.т.н., доцент кафедри захисту інформації,
Вінницький національний технічний університет,
yuriy.baryshev@gmail.com

Комаров А.О.,
магістрант кафедри захисту інформації,
Вінницький національний технічний університет,
komand9@gmail.com

Анотація. *У роботі наведено аналіз сучасного стану розвитку геш-функцій та атак на них. Виділено загальні атаки як основні загрози гешуванню, що передбачає розпаралелення обчислень. Для покращення стійкості запропоновано методи зав'язування даних, реалізовані у конструкціях гешування. Наведено теоретичне обгрунтування та результати експериментального підтвердження підвищення стійкості до атак, які використовують мультиколізії.*

Багато процесорів розробляються з вбудованими двома, чотирма і більше ядер, як наслідок програми повинні максимально використовувати цю можливість обчислювальних платформ, відповідно постають передумови для розпаралелення обчислень. Особливо це набуває значущості для криптографічних обчислень, які зазвичай є неприродними для арифметико-логічних пристроїв універсальних процесорів, а відтак потребують значних витрат часу для свого виконання. До таких криптографічних обчислень зокрема належить гешування.

Водночас відома низка загальних атак, що використовують мультиколізії, яка особливо значуща для методів розпаралеленого гешування [1]. При цьому стійкість до мультиколізій не залежить від криптографічних примітивів, що використовуються під час гешування, а залежать від використовуваних конструкцій [2-5]. Відповідно постає актуальна задача розробки методів підвищення стійкості гешування до атак на основі мультиколізій, які б зокрема дозволяли використовувати можливість розпаралелення обчислень зі збереженням стійкості геш-функції до зламу.

Метою даної роботи є підвищення стійкості методів гешування шляхом зав'язування даних, що гешуються.

Для досягнення мети необхідно розв'язати такі задачі:

- проаналізувати відомі геш-функції та атаки на них;
- розробити методи підвищення стійкості гешування до мультиколізій;
- розробити конструкції гешування на основі запропонованих методів;
- розробити засоби, що реалізують ці конструкції.

Дамгард і Меркль незалежно одне від одного запропонували теореми, які показують, якщо існує стійка до колізій функція ущільнення для вхідних даних сталої довжини $f(\cdot)$: $\{0, 1\}^b \times \{0, 1\}^t \to \{0, 1\}^t$, то можна спроектувати стійку до колізій функцію ущільнення для вхідних даних змінної довжини $h(\cdot)$: $\{0, 1\}^* \to \{0, 1\}^t$ через ітеративні виклики функції ущільнення $f(\cdot)$ [6-7]. Таким чином, якщо функція ущільнення $f(\cdot)$ вразлива до певної атаки, то й ітерована геш-функція $h(\cdot)$ також буде вразлива до атак, однак в загальному випадку протилежний результат не є коректним [7].

Відома низка методів підвищення стійкості до мультиколізій[1, 2, 7-9]. Дані методи лише ускладнюють зловмиснику задачу побудови мультиколізії, при цьому залишаючи саму можливість її побудови.

В роботі [1] показано, що вразливість до цього класу атак криється в конструкціях, тому методи протидії повинні впроваджуватись саме на рівні конструкцій. Для покращення стійкості до мультиколізій пропонується така конструкція:

$$\begin{cases} h_i = f(h_{i-1}, m_i, m_{i-r_i}) \\ r_i = rand(m_i) \end{cases};$$

де $rand(\cdot)$ – функція, що забезпечує рівномірний розподіл вихідних значень r_i на кожній ітерації (якщо $i - r_i < 0$, то обирається блок $i - r_i + l$).

Нехай для даної конструкції відбувається атака Жу. Тоді зловмисник відповідно парадоксу «дня народження» за $2^{0.5 \cdot n}$ обчислень функції ущільнення $f(\cdot)$ знаходить колізію для блоку даних m_i:

$$\begin{cases} h_i = f(h_{i-1}, m_i, m_{i-r_i}) = f(h_{i-1}, m_i^*, m_{i-r_i^*}) \\ r_i = rand(m_i) \\ r_i^* = rand(m_i^*) \end{cases};$$

де $r_i \in [1; l-1]$, $r_i \in N$.

Тоді граф процесу гешування набуває вигляду, наведеного на рис. 1.

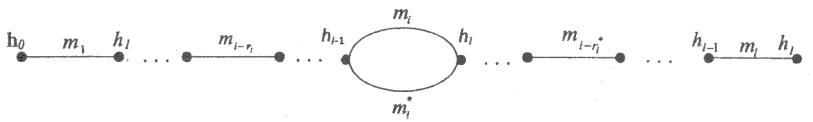

Рис. 1. Граф процесу гешування при реалізації атаки Жу на запропоновану конструкцію

Така атака стане можливою лише за умови, що для $\forall j \in N, j \in [1; l], j - r_i \neq i$, що малоймовірно у випадку, коли вихідні значення функції $rand(\cdot)$ підкорюються рівномірному закону розподілу при $l > 2$, де l – кількість блоків даних в повідомленні, що гешується.

Для покращення стійкості такого методу, з послідовним обчисленням, пропонується визначати номер другого блоку даних залежно від більшої кількості аргументів функції $rand(\cdot)$:

$$\begin{cases} h_i = f(h_{i-1}, m_i, m_{i-r_i}) \\ r_i = rand(m_i, m_{i+1}) \end{cases};$$

Індуктивним шляхом отримується конструкція, що має $(l\text{-}1)$ аргументів у функції $rand(\cdot)$. Для розпаралеленого гешування ця конструкція має такий вигляд:

$$\begin{cases} h_i^{(1)} = f^{(1)}(h_{i-1}^{(1)}, m_i, m_{i-r_i^{(1)}}); \\ h_i^{(2)} = f^{(2)}(h_{i-1}^{(2)}, m_i, m_{i-r_i^{(1)}}); \\ \ldots \\ h_i^{(q)} = f^{(q)}(h_{i-1}^{(q)}, m_i, m_{i-r_i^{(q)}}); \\ r_i^{(1)} = rand^{(1)}(m_1, m_2, \ldots, m_i, m_{i+2}, m_{i+3} \ldots, m_l); \\ r_i^{(2)} = rand^{(2)}(m_1, m_2, \ldots, m_i, m_{i+2}, m_{i+3} \ldots, m_l); \\ \ldots \\ r_i^{(q)} = rand^{(q)}(m_1, m_2, \ldots, m_i, m_{i+2}, m_{i+3} \ldots, m_l). \end{cases}$$

Таке гешування складно апаратно реалізовувати у вигляді спеціалізованого процесора, оскільки це потребуватиме q функцій (а як наслідок і блоків) генерування псевдовипадкових чисел. Цю задачу автори пропонують розв'язувати шляхом перестановки блоків даних в кожному з каналів гешування, тобто для j-го каналу вхідними даними пропонується використовувати послідовність блоків даних:

$$\left\{m_1^{(j)}, m_2^{(j)}, ..., m_l^{(j)}\right\} = p^{(j)}\left(\{m_1, m_2, ..., m_l\}\right),$$

де $p^{(j)}(\cdot)$ – j-та перестановка.

Отже, використовуючи даний метод, конструкція має такий вигляд:

$$\begin{cases} h_i^{(1)} = f^{(1)}(h_{i-1}^{(1)}, m_i^{(1)}, m_{i-r_i^{(1)}}^{(1)}); \\ h_i^{(2)} = f^{(2)}(h_{i-1}^{(2)}, m_i^{(2)}, m_{i-r_i^{(2)}}^{(2)}); \\ ... \\ h_i^{(q)} = f^{(q)}(h_{i-1}^{(q)}, m_i^{(q)}, m_{i-r_i^{(q)}}^{(q)}); \\ r_i^{(1)} = rand^{(1)}(m_1^{(1)}, m_2^{(1)}, ..., m_i^{(1)}, m_{i+2}^{(1)}, ..., m_l^{(1)}); \\ r_i^{(2)} = rand^{(2)}(m_1^{(2)}, m_2^{(2)}, ..., m_i^{(2)}, m_{i+2}^{(2)}, ..., m_l^{(2)}); \\ ... \\ r_i^{(q)} = rand^{(q)}(m_1^{(q)}, m_2^{(q)}, ..., m_i^{(q)}, m_{i+2}^{(q)}, ..., m_l^{(q)}). \end{cases}$$

Таким чином на основі методів зав'язування даних було розроблено конструкції з використанням додаткового аргументу у функції ущільнення. Використання запропонованих конструкцій дозволить унеможливити побудову мультиколізій та пришвидшити обчислення, за рахунок розпаралелення виконуваних операцій, за умови що буде обрано відповідну безпечну функцію ущільнення.

Для тестування було згенеровано 5 псевдовипадкових послідовностей довжиною 63992 байт. З додаванням розміру повідомлення і розбиванням на 256 біт, отримано 1000 блоків даних. В таблиці 1 зображено частоту вибору певного блоку на кожні ітерації гешування для кожної з п'яти послідовностей. Використовувалась генерація номеру блоку даних відносно одного блоку даних m_i.

Таблиця 1

Частота вибору блоку у якості аргументу функції ущільнення

	Експе-римент №1	Експе-римент №2	Експе-римент №3	Експе-римент №4	Експе-римент №5
Блоки не були зав'язані	361/1000	385/1000	367/1000	367/1000	364/1000
Блоки були зав'язані на 1 ітерації	381/1000	354/1000	478/1000	363/1000	381/1000
Блоки були зав'язані на 2 ітераціях	175/1000	172/1000	171/1000	197/1000	173/1000
Блоки були зав'язані на 3 і більше ітераціях	83/1000	89/1000	84/1000	73/1000	82/1000

З таблиці 1 видно, що близько третини блоків не беруть участі у зав'язуванні блоків, близько 40% використовуються на певній ітерації 1 раз, і менше третини блоків використовуються у функції ущільнення більше одного разу.

Даний експеримент був проведений для реалізації, у якій згенерований номер блоку даних залежить від значення двох блоків на кожній ітерації (m_i та m_{i+1}). Як наведено в табл. 2, в даному випадку кожен блок даних став аргументом функції

ущільнення як мінімум один раз (не враховуючи те, що він є аргументом m_i на i-ій ітерації).

Порівнюючи дані у таблицях, можна зробити висновок, що кількість пов'язаних блоків даних збільшилась від 60 до 100%, а зав'язаність блоків в цілому(кількість використань блоку на інших ітераціях) збільшилась більше ніж в 2 рази.

Таблиця 2

Частота вибору блоку у якості аргументу функції ущільнення

	Експе-римент №1	Експе-римент №2	Експе-римент №3	Експе-римент №4	Експе-римент №5
Блоки не були зав'язані	-	-	-	-	-
Блоки були зав'язані на 1 ітерації	1000/1000	1000/1000	1000/1000	1000/1000	1000/1000
Блоки були зав'язані на 2 ітераціях	375/1000	368/1000	371/1000	389/1000	380/1000
Блоки були зав'язані на 3 і більше ітераціях	258/1000	264/1000	255/1000	262/1000	261/1000

З проведеного аналізу атак, що використовують мультиколізії виплило, що основним методом протидії ним є порушення ітеративності процесу гешування. Для цього було запропоновано низку конструкцій гешування повідомлення, які передбачають використання аргументу у функції ущільнення, що визначається за певним псевдовипадковим законом. Аналіз цього підходу дозволив формально обгрунтувати підвищення стійкості, зокрема до атак Жу та Келсі-Коно.

Література:

1. Лужецький В. А. Конструкції хешування стійкі до мультиколізій / В. А. Лужецький, Ю. В. Баришев // Наукові праці ВНТУ. – 2010. – №1. – С 1-8

2. Joux A. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions / Antoine Joux // Lecture Notes in Computer Science. – 2004. – № 3152. – С. 306-316

3. Kelsey J. Herding hash functions and the Nostradamus attack / John Kelsey, Tadayoshi Kohno. – 2005. – 18 с. – Режим доступу до статті: http://archives.scovetta.com/pub/crypto/Nostradamus%20Attack.pdf

4. Lucks S. Design Principles for Iterated Hash Functions / S. Lucks // Cryptology ePrint Archive. – 2004. – 22 с. – Режим доступу до ресурсу: http://eprint.iacr.org/2004/253.pdf

5. Hoch J. Breaking the ICE - Finding Multicollisions in Iterated Concatenated and Expanded (ICE) / Jonathan J. Hoch, Adi Shamir. – 2006. – 16 с. – Режим доступу до статті: http://link.springer.com/chapter/10.1007%2F11799313_12

6. Merkle-Damgaard Revisited: How to Construct a Hash Function / J. S. Coron [and others] // Advances in Cryptology – CRYPTO 2005

7. Gauravaram P. Cryptographic Hash Functions: Cryptanalysis, Design and Applications. / Praveen Gauravaram. – 2009. – 298 с. – Режим доступу до ресурсу: http://eprints.qut.edu.au/16372/1/Praveen_Gauravaram _Thesis.pdf

8. Preneel B. Analysis and Design of Cryptographic Hash Functions: PhD thesis / Bart Preneel. – Leuven: Katholieke Universiteit Leuven, 1993. – 323 с. – [Електронний ресурс]. – Режим доступу до ресурсу: http://homes.esat.kuleuven.be/~preneel/phd_preneel_feb1993.pdf

9. Bertoni G. The Keccak sponge function family. / G. Bertoni, J. Daemen, M. Peeters, G. V. Assche. – [Електронний ресурс]. – Режим доступу: URL http://keccak.noekeon.org /specs_summary.html – Назва з екрану.