УДК 004.056:004.424.47

Ю. В. Баришев, к. т. н.; А. О. Комаров

МЕТОДИ РОЗПАРАЛЕЛЕНОГО ГЕШУВАННЯ, СТІЙКОГО ДО ЗАГАЛЬНИХ АТАК

У роботі проведено аналіз сучасного стану розвитку геш-функцій та атак на них. На основі цього аналізу визначено, що одними із найнебезпечніших є загальні атаки на основі мультиколізій. Для покращення стійкості конструкцій, що передбачають розпаралелення, запропоновано низку конструкцій гешування та алгоритмів. Визначено функції ущільнення для методів гешування на основі цих конструкцій. Експериментальні дослідження дозволили підтвердити підвищення стійкості запропонованого методу гешування до загальних атак на основі мультиколізій.

Ключові слова: *гешування, розпаралелене гешування, мультиколізія, атака Жу, конструкція гешування, функція ущільнення.*

Вступ

Багато процесорів розробляються з вбудованими двома, чотирма й більше ядрами. Зважаючи на це, програми повинні максимально використовувати цю можливість обчислювальних платформ. Відповідно постає необхідність у розпаралеленні обчислень. На сьогодні всі розпаралелені методи криптографічного гешування є нестійкими до мультиколізій [1 – 4]. При цьому стійкість до мультиколізій не залежить від криптографічних примітивів, що використовують на кожній під час гешування, а залежать від використовуваних конструкцій [3 – 5]. Тому постає актуальна задача розробки методів гешування, які дозволяли б використовувати можливість розпаралелення обчислень зі збереженням стійкості геш-функції до зламу.

Метою цієї роботи є підвищення стійкості методів гешування, які передбачають розпаралелення обчислень, до мультиколізій.

Для досягнення мети виконано такі задачі:

– проаналізовано відомі геш-функції та атаки на них;

– розроблено геш-функції для розпаралеленого гешування, стійкі до мультиколізій;

– розроблено алгоритми гешування;

– розроблено засоби, що реалізують ці алгоритми.

Аналіз відомих конструкцій гешування

Дамгард і Меркль незалежно один від одного запропонували теореми, які показують, якщо існує стійка до колізій функція ущільнення для вхідних даних сталої довжини $f(\cdot)$: $\{0, 1\}^b \times \{0, 1\}^t \rightarrow \{0, 1\}^t$, то можна спроектувати стійку до колізій функцію ущільнення для вхідних даних змінної довжини $h(\cdot)$: $\{0, 1\}^* \rightarrow \{0, 1\}^t$ через ітеративні виклики функції ущільнення $f(\cdot)$. Отже, якщо функція ущільнення $f(\cdot)$ вразлива до певної атаки, то й ітерована геш-функція $h(\cdot)$ також буде вразлива до атак, але загалом протилежний результат не є правильний [1 – 3].

Конструкція гешування, запропонована Дамгардом і Мерклем, має такий вигляд:

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

де h_i – проміжне геш-значення, отримане після обробки i-го блоку даних; m_i – i-й блок даних.

Існує багато подібних між собою конструкцій, які використовують блоковий шифр як функцію ущільнення. Однією з найпоширеніших конструкцій такого виду є конструкція Девіса – Меєра. Для певного блокового шифруна основі ключа $kE_k(\cdot)$ конструкція гешування

має вигляд [1, 2, 6, 7]:

$$h_i = E_{m_i}(h_{i-1}) + h_{i-1} \quad (2)$$

Із конструкції (2) видно, що за суттю цей клас конструкцій аналогічний конструкції (1).

Люкс запропонував конструкцію широкого каналу й подвійного каналу, яка використовує

дві функції ущільнення. Для конструкції широкого каналу при $i \in [1; l]$ обчислюють:

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

$$h(M) = f''(h_{i-1}, h_l),$$

де вихідні значення $f'(\cdot)$ мають вдвічі більшу довжину, ніж $f''(\cdot)$ [5].

Ця конструкція має недолік, пов'язаний із тим, що для обчислень вихідних значень функції ущільнення $f'(\cdot)$ необхідно вдвічі більше ресурсів порівняно з $f(\cdot)$ у конструкції Меркля – Дамгарда.

Атаки на геш-функції можуть бути поділені на дві категорії: атаки методом прямого перебору та криптоаналітичні атаки [1, 2, 4]. До атак, що не залежать від алгоритму (прямий перебір, атака методом «днів народження») вразливі всі алгоритми. Єдина можливість їх уникнути – збільшити довжину геш-значення та ключа.

Відповідно до роботи [2], атаки на геш-функції класифікують так, як зображено на рис. 1.

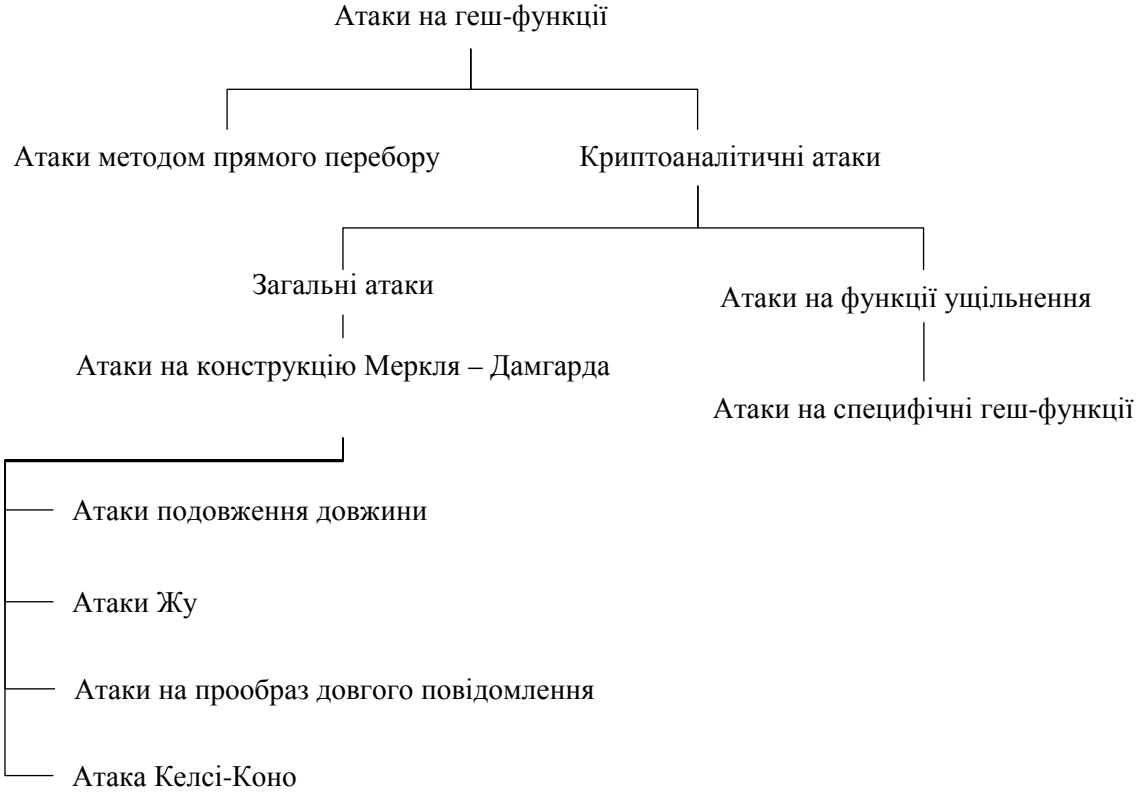

Рис. 1. Класифікація атак за Гауравaрамом

Серед атак цих класів найнебезпечнішими є загальні атаки, оскільки на відміну від атак на функції ущільнення, вони загрожують не одній геш-функції, а всьому класу геш-функцій, які використовують певну конструкцію, попри більшу кількість ресурсів, необхідних для їх реалізації. Так, після появи атаки Жу переважна більшість геш-функцій, побудованих на основі конструкції Меркля – Дамгарда, перестали вважати стійкою. Особливо це стосувалося геш-функцій, де використовували ідею Преніла [6] щодо «каскадування» геш-значень – паралельного обчислення геш-функцій, що забезпечують вихідні значення невеликої розрядності (16, 32, 64 біта – залежно від обчислювальних платформ), та конкатенації їх вихідних значень для отримання вихідного геш-значення. Водночас, на відміну від атак повного перебору, загальні атаки для своєї реалізації потребують суттєво меншої кількості ресурсів, і в певних випадках ця кількість ресурсів може бути доступною для зловмисників, на відміну від кількості ресурсів, необхідних для реалізації атак повного перебору для геш-функцій, які мають сучасні довжини вихідних геш-значень (256, 512, 1024, 2048 біт).

Атака «прямим перебором» [1, 7] може бути виконана для знаходження прообразу за заданим геш-значенням чи для знаходження прообразу, що дає задане геш-значення. Суть атаки полягає в послідовному або випадковому переборі вхідних повідомлень і порівнянні результату обчислення геш-функції для цих повідомленьіз заданим. Складність такої атаки оцінюють 2^{n-1} операцій обчислення функції ущільнення, де n – довжина значень у бітах.

Жу описав загальну атаку з використанням мультиколізій на геш-функцію Меркля – Дамгарда, де показав, що на побудову 2^l колізій потрібно витратити менше часу, ніж2^lреалізацій атак «дня народження» [1, 3, 7]. Такого результату досягнули завдяки оригінальному підходу до побудови колізій. На рис. 2 зображена схема колізій на геш-функцію Меркля – Дамгарда.

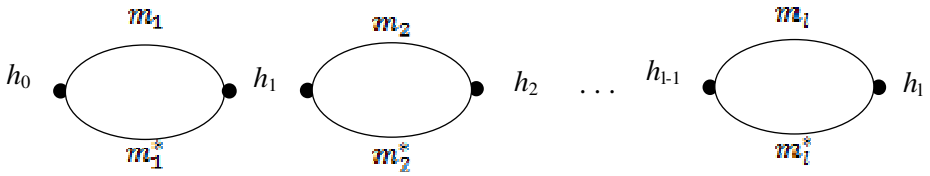

Рис. 2. Граф процесу гешування повідомлень під час атаки Жу на конструкцію Меркля – Дамгарда

Із рис. 2 видно, що атака передбачає l реалізацій атак «дня народження», тобто має складність $O(l \cdot 2^{0.5 \cdot n})$ обчислень функції ущільнення $f(\cdot)$для конструкції Меркля – Дамгарда. Характерною особливістю цього методу є те, що всі повідомлення, які утворюють колізії, мають однакову довжину [3].

Конструкції підвищеної стійкості

Оскільки загальні атаки можуть бути застосовані для різних геш-функцій, які використовують одну конструкцію, тобто вразливість до цих атак криється в конструкціях, то методи протидії необхідно впроваджувати саме на рівні конструкцій.

Для покращення стійкості до мультиколізій пропонуємо таку конструкцію:

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

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

Нехай для цієї конструкції відбувається атака Жу. Тоді зловмисник, відповідно до парадоксу «дня народження» за $2^{0.5 \cdot n}$ обчислень, 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 такого вигляду, як на рис. 3.

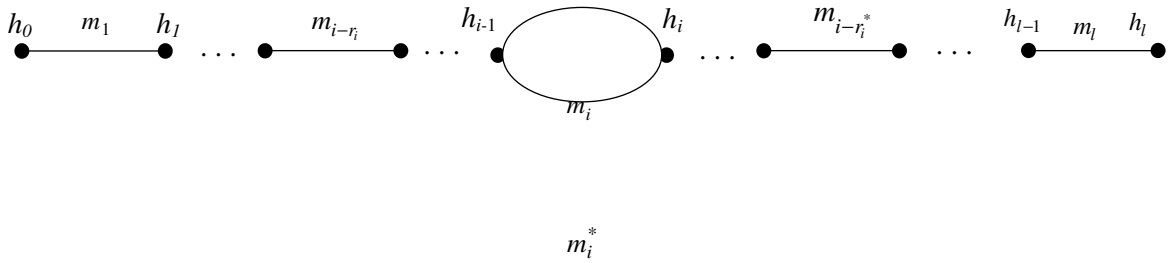

Рис. 3. Граф процесу гешування за реалізації атаки Жу на запропоновану конструкцію

Якщо значення функції $rand(\cdot)$ розподілені відповідно до рівномірного закону і $l > 2$, то така атака буде малоймовірною, оскільки необхідно, щоб виконувалась така умова: $\forall j \in N, j \in [1; l]$ і при цьому $j - r_i \neq i$.

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

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

Оскільки ця конструкція не дозволяє побудову мультиколізії, то її можна використовувати для побудови геш-функцій, що передбачають розпаралелення:

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

За функції ущільнення, результат якої обчислюють залежно від значення блоку даних m_{i-r_i}, де r_i залежить від більшої кількості блоків (m_i та m_i+1), пропонуємо таку конструкцію гешування:

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

Максимальною кількістю аргументів функції *rand*(·) може бути l – 1 блок даних, де l – кількість блоків даних, на які розбивають вхідне повідомлення M. Ця конструкція має такий вигляд для алгоритму із послідовною обробкою:

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

У цьому випадку викидається m_i блок даних. Для паралельного обчислення конструкція матиме такий вигляд:

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

Отже, було розроблено конструкції з використанням третього аргументу у функції ущільнення. Використання запропонованих конструкцій дозволить протидіяти мультиколізіям і пришвидшити обчислення за рахунок розпаралелення виконуваних операцій за умови, що буде обрано відповідну безпечну функцію ущільнення.

Функції ущільнення

Для побудови геш-функцій на основі конструкцій гешування, наведених вище, необхідно реалізувати функції ущільнення. Наразі відома велика кількість способів їх реалізації, однак з погляду параметрів стійкість/швидкість одними з найпоширеніших є такі [1, 7 – 10]:

– на основі піднесення до степеня за модулем простого числа (дискретне логарифмування) [1, 7]:

$$f(a,b,c) = g^{a+b+c} \bmod p.$$

– на основі еліптичних кривих (дискретне логарифмування): обирають точку на еліптичній кривій x і для неї розраховують координату y. Криву задають формулою, наприклад $y^2 = x^3 + ax^2 + bx + c$.

– на основі нелінійних логічних функцій (SHA2, SEAL) [10, 11]:

$$f(a,b,c) = (a \wedge b) \vee (a \wedge c);$$

$$f(a,b,c) = a \oplus b \oplus c;$$

$$f(a,b,c) = (a \wedge b) \vee (a \wedge c) \vee (b \wedge c);$$

$$f(a,b,c) = (a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c);$$

$$f(a,b,c) = (a \wedge b) \oplus (\bar{a} \wedge c);$$

Функція ущільнення на основі піднесення до степеня за модулем простого числа та на основі еліптичних кривих є теоретично стійкою до атак [7]. Хоча оцінки її складності менші за оцінки реалізації атак повного перебору, вони достатньо високі для забезпечення гешування, злам якого з практичного погляду неможливий.

Останні п'ять перетворень дозволяють швидко об'єднувати вхідні значення за рахунок швидкості обчислення логічних операцій на мікропроцесорах, однак їх стійкість теоретично не доведена, і не можна гарантувати, що вони не будуть зламані в подальшому. Попри це, вони були детально досліджені в роботах різних учених (унаслідок їх використання в міжнародних стандартах гешування, які були чинними близько 20 років), і в цих роботах не

показана їх уразливість [10, 11].

Для тестування було згенеровано п’ять псевдовипадкових послідовностей довжиною 6400 байт. Із додаванням розміру повідомлення й розбиванням на 256 біт отримано 1000 блоків даних. У табл. 1 зображено частоту вибору певного блоку на кожній ітерації гешування для кожної з п’яти послідовностей. Використовували генерацію номеру блоку даних відносно одного блоку даних (m_i).

Таблиця 1

Частота вибору блоку в якості аргументу функції ущільнення

	Експе-римент №1	Експе-римент №2	Експе-римент №3	Експе-римент №4	Експе-римент №5
Блоки не були зав’язані	351/1000	40/1000	36/1000	33/1000	40/1000
Блоки були зав’язані на 1 ітерації	400/1000	359/1000	402/1000	389/1000	310/1000
Блоки були зав’язані на 2 ітераціях	170/1000	150/1000	170/1000	250/1000	210/1000
Блоки були зав’язані на 3 і більше ітераціях	81/1000	110/1000	82/1000	40/1000	91/1000

Із табл. 1 видно, що близько третини блоків не беруть участі в зав’язуванні блоків, близько 40% використовуються на певній ітерації 1 раз, і менше третини блоків використовуються у функції ущільнення більше одного разу. За зміни методу генерування псевдовипадкових чисел отримані результати експериментального дослідження зміняться.

Цей 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}). У цьому 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) (табл. 2).

Таблиця 2

Частота вибору блоку в 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	361/1000	432/1000	371/1000	389/1000	380/1000
Блоки були зав’язані на 3 і більше ітераціях	271/1000	240/1000	200/1000	271/1000	262/1000

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

Висновки

Із проведеного аналізу атак, що використовують мультиколізії, випливає, 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 визначають за певним псевдовипадковим законом.

Аналіз цього підходу дозволив формально обґрунтувати підвищення стійкості, зокрема до атак Жу та Келсі – Коно. За результатами експериментального дослідження, було виявлено, що за використання певного типу зав'язування блоків даних отримуване підвищення стійкості варіюється залежно від вибору генератора псевдовипадкових чисел. Тому саме цій особливості будуть присвячені подальші дослідження.

СПИСОК ЛІТЕРАТУРИ

1. Баришев Ю. В. Методи та засоби швидкого багатоканального гешування даних в комп'ютерних системах : монографія / Ю. В. Баришев, В. А. Лужецький; за заг. ред. В. А. Лужецького. – Вінниця : ВНТУ, 2016. – 144 с.

2. Gauravaram P. Cryptographic Hash Functions: Cryptanalysis, Design and Applications [Електронний ресурс] / Praveen Gauravaram. – 2009. – 298 с. – Режим доступу до ресурсу: http://eprints.qut.edu.au/16372/1/Praveen_Gauravaram_Thesis.pdf. – Назва з екрану.

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

4. Kelsey J. Second preimages on n-bit hash functions for much less than 2n work / J. Kelsey, B. Schneier // EUROCRYPT. – 2005. – P. 474 – 490.

5. Lucks S. Design Principles for Iterated Hash Functions [Електронний ресурс] / S. Lucks // Cryptology ePrint Archive. – 2004. – 22 с. – Режим доступу до ресурсу: http://eprint.iacr.org/2004/253.pdf. – Назва з екрану.

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

7. Schneier B. Applied Cryptography, Second edition: Protocols, Algorithms, and Source Code in C / Bruce Schneier. – New-York: Wiley Computer Publishing, John Wiley & Sons, Inc, 1996. – 1027 с.

8. Schneier B. The Skein Hash Function Family [Електронний ресурс] / [B. Schneier, N. Ferguson, S. Lucks and others]. – Режим доступу: URL https://www.schneier.com/skein1.3.pdf. – Назва з екрану.

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

10. Secure Hash Signature Standard (SHS): FIPS PUB 180-2. – [Чинний від 2002-08-01]. Processing Standards Publication. – Gaithersburg 2002. – 76 с. (NIST).

11. Implementation of SHA-256 in C [Електронний ресурс] / Режим доступу: URL http://bradconte.com/sha256_c. – Назва з екрану.

Баришев Юрій Володимирович – к. т. н.. доцент кафедри захисту інформації, e-mail:yuriy.baryshev@gmail.com.

Комаров Андрій Олегович – магістрант кафедри захисту інформації, e-mail: komand9@gmail.com.
Вінницький національний технічний університет.