УДК 003.26 : 004.424.47

В. А. Лужецький, д. т. н., проф.; Ю. В. Баришев

КОНСТРУКЦІЇ ХЕШУВАННЯ СТІЙКІ ДО МУЛЬТИКОЛІЗІЙ

У цій статті представлено аналіз атак, які ґрунтуються на знаходженні мультиколізій та методів протидії їм. Запропоновано узагальнену конструкцію багатоканального хешування. З використанням цієї конструкції узагальнено та удосконалено відомі підходи підвищення стійкості хешування до мультиколізій. Запропоновано новий підхід до побудови паралельного хешування стійкого до мультиколізій, який було формалізовано у вигляді конструкцій. Наведено оцінки часу хешування за допомогою запропонованих у статті конструкцій.

Ключові слова: *конструкції хешування, мультиколізії, атака Жукса, стійкі конструкції багатоканального хешування.*

Вступ

Останнім часом галузь криптографії, яка пов'язана з хешуванням, зіткнулася із суттєвою проблемою – забезпеченням стійкості до мультиколізій, які використовує атака Жукса. Зародження цієї проблеми почалося з атаки "Дня народження", відповіддю на яку стало збільшення вдвічі довжини вихідного хеш-значення. Такий вихід став неадекватним для наявних обчислювальних ресурсів суспільства та обмежень на період виконання хешування. Для скорочення тривалості хешування запропоновано каскадування, описане в роботі [1], тобто розпаралелення обчислень хеш-значень малої розрядності, які обчислювались, використовуючи різні хеш-функції, та їх конкатенація після завершення останньої ітерації. Каскадування вважалося ефективним запобіганням атаці "Дня народження" до 2004 року, коли Жукс опублікував роботу [2]. Знайшовши, так звані, мультиколізії в одній з хеш-функцій, які обчислюються паралельно, Жукс показав, що стійкість хеш-значення, отриманого шляхом каскадування цих функцій, не набагато більша за стійкість однієї з них (за умови однакової стійкості функцій). Отже, проблема, пов'язана зі стійкістю та тривалістю хешування, повернулася.

Підходи, які запропоновано Жуксом у його атаці, узагальнено для довільної кількості хеш-функцій та удосконалено для швидшого знаходження мультиколізій у роботах [3 – 6]. Здійснено низку спроб запропонувати математичні моделі хеш-функцій стійких до таких атак, які відповідно до традиції, що склалася в західній літературі, зазвичай називають конструкціями. Найбільш відомими є конструкція HAIFA [7] та конструкція подвоєного каналу Люкса [8]. Перша використовує ряд специфічних аргументів, але теоретично не доводить збільшення стійкість цієї конструкції до атаки Жукса, а друга – вирішує задачу шляхом збільшення вдвічі обчислень та довжини проміжних хеш-значень.

Метою цього дослідження є розробка конструкцій хешування, які забезпечують розпаралелення обчислень хеш-значень та не дозволяють побудувати мультиколізії .

Для досягнення мети розв'язуються такі задачі:
– аналіз відомих методів побудови мультиколізій;
– аналіз відомих конструкцій, що пропонувались для збільшення стійкості мультиколізіям;
– розробка узагальненої конструкції хешування;
– в узагальненій конструкції хешування визначення властивостей, які не дозволяють побудувати мультиколізію криптоаналітику.

Аналіз методів побудови мультиколізій

Вперше атаку з використанням мультиколізії запропоновано у роботі [2] Жуксом. Цю атаку запропоновано для хеш-функцій, які використовують каскадування [1], та ґрунтуються на підсиленій конструкції Меркля-Дамгаарда. Відповідно до цієї конструкції, інформаційне

повідомлення **M** доповнюється до довжини, яка кратна заданій довжині блока даних, розбивається на l частин та доповнюється блоком m_{l+1}, який містить довжину оригінального повідомлення **M**, а хешування відбувається за формулою:

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

де h_i – проміжне хеш-значення, отримане на i-му кроці; $f(\cdot)$ – функція ущільнення, що забезпечує фіксовану довжину результату.

Розглянемо дві функції ущільнення $f^{(1)}(\cdot)$ та $f^{(2)}(\cdot)$, що реалізують хешування відповідно до формули (1). Вихідне хеш-значення формується шляхом конкатенації хеш-значень малої розрядності $h_{l+1}^{(1)}$ та $h_{l+1}^{(2)}$, обчислених за допомогою $f^{(1)}(\cdot)$ та $f^{(2)}(\cdot)$ відповідно. Такий спосіб хешування дозволяє виконувати паралельне обчислення функцій $f^{(1)}(\cdot)$ та $f^{(2)}(\cdot)$, а тому скоротити час хешування. Проте, як довів Жукс у [2], це зменшує стійкість результуючого хеш-значення, порівняно з хеш-значенням, визначеним за допомогою однієї функції, яка має вдвічі більшу розрядність хеш-значення, ніж $f^{(1)}(\cdot)$ та $f^{(2)}(\cdot)$.

Свою атаку Жукс побудував на знаходженні мультиколізії для однієї функції. Потім серед утворених колізій шукають такі, які спричинять колізію й в іншій функції. Мультиколізію Жукс отримав, знаходячи для кожного i-го блоку даних m_i інший блок даних m_i^* такий, у якому виконується така умова:

$$f^{(1)}(m_i, h_{i-1}) = f^{(1)}(m_i^*, h_{i-1}). \tag{2}$$

На рис. 1 схематично зображено як знаходить мультиколізії Жукс у своїй роботі [2] для хеш-функцій, що використовують підсилену конструкцію Меркля-Дамгаарда або аналогічних до неї.

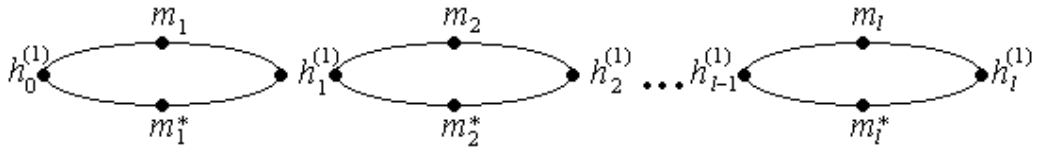

Рис. 1. Вигляд мультиколізії Жукса

Використовуючи l-мультиколізію, яку зображено на рис. 1, можна побудувати 2^l різних повідомлення, які викличуть колізію в першій функції $f^{(1)}(\cdot)$, а потім серед них зі значною ймовірністю можна знайти хоча б одне таке повідомлення, яке призведе до колізії й в іншій функції $f^{(2)}(\cdot)$. Для підсиленої конструкції Меркля-Дамгаарда останній блок даних, які хешуються, m_{l+1} не є об'єктом пошуку колізії, оскільки він містить довжину оригінального повідомлення, а тому його підміна одразу буде помічена. Очевидно, що чим більша кількість блоків даних l, тим більша ймовірність знаходження колізії. Отже, складність пошуку колізії для n-розрядних функцій $f^{(1)}(\cdot)$ та $f^{(2)}(\cdot)$ буде оцінюватись в $O(l \cdot 2^{n/2})$ операцій замість очікуваного значення $O(2^{2n/2})$.

У роботі [2] Жукс запропонував знаходження мультиколізій лише для випадку, коли паралельно обчислюються дві хеш-функції і блоки даних обробляються лише під час однієї ітерації. Робота [3] направлена на пошук мультиколізій за умови, що блоки даних обробляються за допомогою функції ущільнення двічі. У роботі [4] Хоч та Шамір розвинули та узагальнили цей результат. Вони розглядали випадок, коли хеш-функції "розширені" (expanded), тобто вхідні блоки даних для функцій $f^{(1)}(\cdot)$ та $f^{(2)}(\cdot)$ можуть подаватись на канали обчислень, за термінологією Люкса [8], в різній послідовності. Наприклад, за допомогою функції $f^{(1)}(\cdot)$ обчислюється хеш-значення спочатку за парними блоками даних,

а потім за непарними, а за допомогою функції $f^{(2)}(\cdot)$ – навпаки, або використовується будь-який інший спосіб перестановки блоків даних. Крім того, у роботі [4] до поняття "розширені" хеш-функції належать такі, які обробляють кожен блок даних декілька разів, наприклад, рухаючись перший раз від m_1 до m_l, а потім в інверсному напрямку. Ці ідеї набули подальшого розвитку у статті [5], в якій автори з позиції теорії автоматів описують хешування та пошук мультиколізій.

У роботі [6] запропоновано підхід до побудови мультиколізій, які ґрунтуються на знаходженні "повідомлень, що розширюються" (expandable messages). Ці повідомлення використовують аналогічно m_i^* з формули (2), проте на відміну від підходу, запропонованого Жуксом, "повідомлення, що розширюються" можуть складатись з різної кількості блоків даних. У такий спосіб знаходимо декілька повідомлень різної довжини, які спричиняють колізію та мають однакове проміжне хеш-значення як початкове. Крім того, у статті [6] автори показують як можна збільшити ефективність цієї атаки, якщо функція ущільнення, яка лежить в основі хешування, має фіксовані точки, що характерно для конструкції
Девіса-Мейєра. Для довгих повідомлень такий підхід дозволив досягти кращих результатів, ніж у роботі [2].

Аналіз відомих конструкцій хешування

З метою ускладнення атак, що використовують пошук мультиколізій, авторами статті [7] запропоновано конструкцію HAIFA (HAsh Iterative FrAmework). Ця конструкція передбачала розширення аргументів ітеративної функції за рахунок збільшення їх кількості в конструкції (1):

$$h_i = f\left(m_i, h_{i-1}, \#bits_i, r\right), \tag{3}$$

де $\#bits_i$ – кількість вже захешованих бітів повідомлення; r – псевдовипадкове число.

Псевдовипадкове число r в роботі [7] пропонуємо використовувати як ідентифікатор сесії обміну даними або як певне секретне число при використанні хеш-функції для цифрового підпису. Введення додаткових аргументів функції ущільнення дозволило утруднити криптоаналітику задачу заздалегідь підготовленої атаки, але водночас, конструкція (3) не забезпечує нерозв'язуваність цієї задачі. Наявність лічильника вже захешованих даних змушує шукати криптоаналітика колізії з врахуванням відносного місця блока даних у повідомленні, але це ніяк не протидіє атаці Жукса. Основною перевагою конструкції (3) є те, що доти, доки зловмиснику невідоме псевдовипадкове число r та хоча б частина повідомлення, яке хешується, він не може почати атаку.

Найбільш стійкою до мультиколізій є конструкція "подвоєного каналу" (double pipe) Люкса [8], тому вона має найбільший потенціал для розпаралелення порівняно з іншими конструкціями. Конструкція, яку запропоновано Люксом, виникла з очевидної відповіді на атаку Жукса. Якщо при розпаралеленні стійкість n-розрядного хеш-значення $O\left(2^{n/2}\right)$, то необхідно вдвічі збільшити розрядність проміжних хеш-значень [8]. Так з'явилася конструкція "широкого каналу" (wide pipe) [8], проте таке збільшення розрядності очевидно буде негативно вливати на швидкість та обчислювальні ресурси. Для уникнення цього Люксом запропоновано конструкцію "подвоєного каналу", яка передбачає наявність двох обчислювачів, каналів за допомогою яких визначається проміжне хеш-значення розрядності n, при цьому інформаційні дані розбиваються на блоки по k біт ($k \geq n$) [8]:

$$\begin{cases} h_i^{(1)} = f\left(h_{i-1}^{(1)}, h_{i-1}^{(2)} \parallel m_i\right) \\ h_i^{(2)} = f\left(h_{i-1}^{(2)}, h_{i-1}^{(1)} \parallel m_i\right) \end{cases}, \tag{4}$$

де $h_i^{(1)}$ та $h_i^{(2)}$ – проміжні хеш-значення, отримані на першому та другому каналах відповідно.

Після останньої ітерації передбачається раунд хешування, який ущільнює отримані значення до n розрядів [8]:

$$h = f\left(h^{\circ}, h_l^{(1)} \| h_l^{(2)} \| 0^{k-n}\right), \tag{5}$$

де h° – початкове заповнення, аналогічне до $h_0^{(1)}$ та $h_0^{(2)}$; 0^{k-n} – додаток до повного $(k+n)$-розрядного блоку.

Основними недоліками конструкції "подвоєного каналу" є подвійні апаратні витрати для реалізації хешування порівняно з іншими конструкціями; велика кількість векторів ініціалізації – $3n$, що ускладнює розробку ключового варіанту хешування на основі такої конструкції. Варто відзначити, що останній крок негативно впливає на стійкість всієї конструкції загалом та порушує однорідність процесу хешування. Крім того, структура Люкса залишається вразливою до атак, які використовують попередню підготовку криптоаналітика.

Узагальнимо відомі конструкції хешування для визначення методів протидії побудові мультиколізій.

Узагальнена конструкція хешування

Розглянемо узагальнену конструкцію ітеративного хешування. Визначимо основні аргументи, які можуть використовуватися в хеш-функції. Усі відомі конструкції ітеративного хешування як аргументи обов'язково мають блок повідомлення, яке хешується, та проміжне хеш-значення, отримане в результаті попередньої ітерації, як і в конструкції (1). Розширимо це поняття, вважаючи, що проміжне хеш-значення, отримане на i-му кроці, може залежати від: усіх блоків інформаційних даних; i попередніх проміжних хеш-значень (включаючи вектор ініціалізації h_0); додаткових ключових даних. Оскільки в більшості випадків як ключові дані використовують вектор ініціалізації h_0, а використання додаткових ключових даних обумовлюється специфікою конкретних задач і не впливає на пошук мультиколізій, то не будемо їх окремо розглядати в узагальненій конструкції для зменшення її загромадженості.

Отже, проміжне хеш-значення на i-й ітерації може обчислюватись за функцією такого виду:

$$h_i = f\left(h_0, h_1, ..., h_{i-1}, m_1, m_2, ..., m_l, m_{l+1}\right).$$

У загальному випадку можна виконувати підсилення конструкції, враховуючи кількість вже захешованих інформаційних біт замість дописування бітової довжини повідомлення до його кінця, як це зроблено в конструкції (3):

$$h_i = f\left(h_0, h_1, ..., h_{i-1}, m_1, m_2, ..., m_l, length_i\right), \tag{6}$$

де $length_i$ – бітова довжина вже захешованої частини повідомлення або її еквівалент.

У хешуванні можуть використовувати й аргументи, що змінюються псевдовипадково. Якщо розглядати загальний випадок, то числа псевдовипадкової послідовності можуть використовуватись декілька раз на різних ітераціях. Крім того, на одній ітерації може бути використано декілька псевдовипадкових чисел. Враховуючи це, формула (6) набуде такого вигляду:

$$h_i = f\left(h_0, h_1, ..., h_{i-1}, m_1, m_2, ..., m_l, r_1, r_{2,}..., r_z, length_i\right), \tag{7}$$

де z – кількість псевдовипадкових чисел, які використовують під час однієї ітерації; $r_1, r_{2,}..., r_z$ – псевдовипадкові числа.

Псевдовипадкові числа можуть бути опубліковані відкрито, визначатись лише перед початком обміну даними або використовуватись як певний секретний параметр для цифрового підпису, як запропоновано в роботі [7].

З метою розпаралелення обчислень при хешуванні потрібно використовувати q каналів обчислення. Конструкція (7) може бути узагальнена таким чином:

$$\begin{cases} h_i^{(1)} = f\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(1)}, r_2^{(1)}, ..., r_{z_1}^{(1)}, length_i\right) \\ h_i^{(2)} = f\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(2)}, r_2^{(2)}, ..., r_{z_2}^{(2)}, length_i\right) \\ ... \\ h_i^{(q)} = f\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(q)}, r_2^{(q)}, ..., r_{z_q}^{(q)}, length_i\right) \end{cases}, \quad (8)$$

де z_j – кількість псевдовипадкових чисел, які використовують в j-му каналі ($\sum_{j=1}^{q} z_j = z$).

Крім того, хешування може відбуватися з використанням різних функцій ущільнення на кожній ітерації в кожному каналі. Тому формула (8) зміниться так:

$$\begin{cases} h_i^{(1)} = f_i^{(1)}\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(1)}, r_2^{(1)}, ..., r_{z_1}^{(1)}, length_i\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(2)}, r_2^{(2)}, ..., r_{z_2}^{(2)}, length_i\right) \\ ... \\ h_i^{(q)} = f_i^{(q)}\left(h_0^{(1)}, h_1^{(1)}, ..., h_{i-1}^{(1)}, h_0^{(2)}, h_1^{(2)}, ..., h_{i-1}^{(q)}, m_1, m_2, ..., m_l, r_1^{(q)}, r_2^{(q)}, ..., r_{z_q}^{(q)}, length_i\right) \end{cases}. \quad (9)$$

Введемо таке позначення для багатоканального хешування (Multi-Pipe Hashing) – $\mathrm{MPHq(k,d,r,z)}$, де q – кількість каналів, k – кількість каналів, від проміжних хеш-значень яких залежить наступне хеш-значення j-го $(\mathrm{j}=1,2,...,\mathrm{q})$ каналу, d – кількість блоків даних, які беруть участь у формуванні хеш-значення в j-му каналі, g – порядок псевдовипадковості, який характеризує режим застосування псевдовипадкових чисел ($g=0$ – псевдовипадкові числа не використовують, $g=1$ – використовують як додаткові дані, $g=2$ – використовують як індекси блоків даних), z – кількість псевдовипадкових чисел, які використовують під час однієї ітерації.

Отже, було отримано узагальнену конструкцію хешування. Розглянемо її властивості, які зможуть протистояти побудові мультиколізій.

Визначення методів утруднення побудови мультиколізій

Найбільш очевидним методом ускладнена побудови мультиколізій є багатократна обробка блоку інформаційних даних. Проте такий метод може ускладнити лише класичну атаку Жукса, а не пізніші атаки, що походять від неї. До таких належить, наприклад, атака, яку запропоновано в статті [4].

Більш цікавим методом ускладнення побудови мультиколізій є одночасне використання декількох блоків даних на кожній ітерації, наприклад, таке:

$$h_i = f\left(h_{i-1}, m_{i-a}, m_{i-b}\right), \quad (10)$$

де a,b – деякі константи.

Реалізація конструкції (10) можлива декількома способами: послідовним обчисленням двох блоків даних аналогічно конструкції (1) – фактично це дві послідовні ітерації; "об'єднанням" m_{i-a} та m_{i-b} за допомогою певної операції, яка швидко виконується. У першому випадку отримуємо хеш-функцію, вразливість, якої до мультиколізій доведено ще в роботі [3]. Другий випадок є більш цікавим. Нехай операцію, яка перетворює два операнди в

один, позначимо знаком "$\circ$", тоді конструкцію (10) можна записати так:

$$\begin{cases} m'_i = m_{i-a} \circ m_{i-b} \\ h_i = f\left(h_{i-1}, m'_i\right) \end{cases}. \tag{11}$$

Отримано конструкцію, яка більш стійка до мультиколізій, ніж (1), але, очевидно, що її можна зламати аналогічними діями. Так, застосовуючи атаку Жукса, можна знайти такі $m'^{*}_i \neq m'_i$, що відповідатимуть рівності, яка аналогічна з (2), потім знайти мультиколізію. Основною відмінністю між конструкціями (1) та (11) є те, що при використанні конструкції (11), зловмиснику доведеться розв'язувати систему рівнянь для знаходження m^*_{i-a} та m^*_{i-b}. Це не повинно викликати значних проблем у криптоаналітика.

Останнє не свідчить про те, що підхід, використаний у конструкції (10), є неперспективним, але він потребує удосконалення. Для цього змінимо порядок псевдовипадковості та будемо використовувати псевдовипадкові числа як індекси блоків даних, що хешуються. Дійсно, введення псевдовипадковості в перше рівняння в системі (11) не дозволить зловмиснику згодом побудувати систему рівнянь, про яку йшла мова вище. Пропонуємо таке введення псевдовипадковості:

$$\begin{cases} r_i = rand\left(m_i\right) \\ m'_i = m_i \circ m_{i-r_i} \\ h_i = f\left(h_{i-1}, m'_i\right) \end{cases}, \tag{12}$$

де $rand(\cdot)$ – деяка функція, значення якої має рівномірний закон розподілу.

Наприклад, функція $rand(\cdot)$ може бути реалізована у вигляді генератора псевдовипадкових чисел, початковим станом якого є блок даних m_i. Конструкція (12) стійка до мультиколізій, оскільки зловмисник вже не може послідовно замінювати блоки даних для отримання мультиколізії. Дійсно, якщо одну колізію m^*_i знайдено, то вже блок m_{i-r_i} та блок $m_{i-r^*_i}$ ($r^*_i = rand\left(m^*_i\right)$) не можна замінити для побудови мультиколізій. Відповідно, якщо продовжувати пошук колізій, то l колізій не можливо перетворити в l-мультиколізію. Багатоканальна варіант конструкції (12) $\mathrm{MPHq}(1,2,2,1)$ матиме вигляд:

$$\begin{cases} r_i = rand\left(m_i\right) \\ h^{(1)}_i = f^{(1)}_i\left(h^{(1)}_{i-1}, m_i \circ m_{i-r_i}\right) \\ h^{(2)}_i = f^{(2)}_i\left(h^{(2)}_{i-1}, m_i \circ m_{i-r_i}\right). \\ \ldots \\ h^{(q)}_i = f^{(q)}_i\left(h^{(q)}_{i-1}, m_i \circ m_{i-r_i}\right) \end{cases} \tag{13}$$

Для ускладнення задачі криптоаналізу пропонуємо використовувати перемішування блоків даних перед хешуванням для різних каналів. Нехай $\left\{\mathrm{m}^{(\mathrm{j})}_1, \mathrm{m}^{(\mathrm{j})}_2, \ldots, \mathrm{m}^{(\mathrm{j})}_\mathrm{l}\right\}$ є деякою перестановкою боків повідомлення $\mathbf{M} = \left\{\mathrm{m}_1, \mathrm{m}_2, \ldots, \mathrm{m}_\mathrm{l}\right\}$, тоді конструкцію $\mathrm{MPHq}(1,2,2,1)$, що описується формулою (13), можна удосконалити до $\mathrm{MPHq}(1,2,2,\mathrm{q})$:

$$\begin{cases} r_i^{(1)} = rand\left(m_i^{(1)}\right) \\ r_i^{(2)} = rand\left(m_i^{(2)}\right) \\ \ldots \\ r_i^{(q)} = rand\left(m_i^{(q)}\right) \\ h_i^{(1)} = f_i^{(1)}\left(h_{i-1}^{(1)}, m_i^{(1)} \circ m_{i-r_i^{(1)}}^{(1)}\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_{i-1}^{(2)}, m_i^{(2)} \circ m_{i-r_i^{(2)}}^{(2)}\right) \\ \ldots \\ h_i^{(q)} = f_i^{(q)}\left(h_{i-1}^{(q)}, m_i^{(q)} \circ m_{i-r_i^{(q)}}^{(q)}\right) \end{cases}. \quad (14)$$

Крім підходу, що запропонований для збільшення стійкості, можна використовувати підхід, який запропонував Люкс в [8]. Конструкцію $\mathrm{MPH2}(2,1,0,0)$, яку описано формулами (4) та (5), можна узагальнити конструкцією $\mathrm{MPHq}(\mathrm{q},1,0,0)$:

$$\begin{cases} h_i^{(1)} = f_i^{(1)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, \ldots, h_{i-1}^{(q)}, m_i\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, \ldots, h_{i-1}^{(q)}, m_i\right) \\ \ldots \\ h_i^{(q)} = f_i^{(q)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, \ldots, h_{i-1}^{(q)}, m_i\right) \end{cases}. \quad (15)$$

Разом зі збереженням ефекту зв'язку каналів хешування після кожної ітерації, можливий варіант спрощення конструкції $\mathrm{MPHq}(\mathrm{q},1,0,0)$, яка описується формулою (15), до конструкції $\mathrm{MPHq}(2,1,0,0)$ зі збереженням аналогічної стійкості при хешуванні повідомлень, які складаються з кількості блоків не меншої за $(q/2+1)$:

$$\begin{cases} h_i^{(1)} = f_i^{(1)}\left(h_{i-1}^{(q)}, h_{i-1}^{(1)}, m_i\right) \\ h_i^{(2)} = f_i^{(2)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, m_i\right) \\ \ldots \\ h_i^{(q)} = f_i^{(q)}\left(h_{i-1}^{(q-1)}, h_{i-1}^{(q)}, m_i\right) \end{cases}. \quad (16)$$

Перевагою конструкції $\mathrm{MPHq}(1,2,2,\mathrm{q})$, що описується формулою (14), над конструкцією $\mathrm{MPHq}(2,1,0,0)$, описану формулою (16), є те, що обчислення в кожному каналі не залежить від обчислень в інших каналах. Це дає можливість виконати незалежне обчислення хеш-значення. Тобто час обчислення хеш-значень за допомогою конструкції $\mathrm{MPHq}(2,1,0,0)$ становить:

$$t_{MPHq(2,1,0,0)} \approx \sum_{i=1}^{l} \max\left(t_i^{(1)}, t_i^{(2)}, \ldots, t_i^{(q)}\right), \quad (17)$$

де $t_i^{(j)}$ – час виконання i-ї ітерації в j-му каналі.

Оцінка тривалості обчислення хеш-значення за допомогою конструкції $\mathrm{MPHq}(1,2,2,\mathrm{q})$ становить:

$$t_{MPHq(1,2,2,q)} \approx \max\left(\sum_{i=1}^{l}\left(t_i^{(1)}\right), \sum_{i=1}^{l}\left(t_i^{(2)}\right), \ldots \sum_{i=1}^{l}\left(t_i^{(q)}\right)\right). \quad (18)$$

Очевидно, що завжди $t_{MPH(q,2,1,0)} \geq t_{MPH(q,1,2,2)}$.

Висновки

Поява атаки "дня народження" змусила збільшити вдвічі розрядність хеш-значень, що суттєво збільшило і час їх обчислення. Це у свою чергу, унеможливило використання хешування для низки задач. Тому розпаралелення хешування необхідне для того, щоб час обчислення знову відповідав вимогам щодо його тривалості, але на заваді розпаралелення стали мультиколізії, що використовуються в атаці Жукса, в її подальших узагальненнях і удосконаленнях, об'єктом яких є хеш-значення, які отримано за допомогою розпаралелених хеш-функцій.

Проаналізувавши відомі методи розпаралелення, можна узагальнити, що у відомих конструкцій стійкість до мультиколізій досягається збільшенням вдвічі ресурсоємності обчислень без зменшення часу обчислень. Відомий підхід Люкса удосконалено і узагальнено шляхом зменшення його ресурсоємності та можливості обчислень за допомогою q паралельних каналів хешування, замість двох, які розглядали в статті [8]. Це дозволяє зменшити час обчислень. Ці підходи формалізовано конструкціями $\mathrm{MPHq}(\mathrm{q},1,0,0)$ та $\mathrm{MPHq}(2,1,0,0)$.

Запропоновано новий підхід до побудови хешування стійкого до мультиколізій, що був формалізований конструкцією $\mathrm{MPHq}(1,2,2,\mathrm{q})$. Отримані оцінки часу обчислення хеш-значень свідчать, що запропонований підхід, який зображено у вигляді конструкції $\mathrm{MPHq}(1,2,2,\mathrm{q})$, буде швидшим, ніж удосконалений, що зображено у вигляді конструкції $\mathrm{MPHq}(2,1,0,0)$, за умов однакової якості їх реалізації.

СПИСОК ЛІТЕРАТУРИ

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

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. Nandi M. Multicollision Attacks on Some Generalized Sequential Hash Functions / M. Nandi, D. R. Stinton // Scientific Literature Digital Library. – Режим доступу до ресурсу : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.104.5370&rep=rep1&type=pdf

4. Hoch J. J. Breaking the ICE – Finding Multicollisions in Iterative Concatenated and Expanded (ICE) Hash Functions / J. J. Hoch, A. Shamir. – 2006. – Режим доступу до ресурсу : http://www.wisdom.weizmann.ac.il/~yaakovh/papers/hashpaper_submission.pdf

5. Halunen K. An Automata-Theoretic Interpretation of Iterated Hash Functions – Application to Multicollisions / Kimmo Halunen, Juha Kortelainen, Tuomas Kortelainen // Cryptology ePrint Archive. – 2009. – 13 с. – Режим доступу до ресурсу : http://eprint.iacr.org/2009/456.pdf

6. Kelsey J. Second preimages on *n*-bit Hash Function for Less than 2^n Work. / J. Kelsey, B. Schneier // Cryptology ePrint Archive. – 2004. – 15 с. – Режим доступу до ресурсу : http://eprint.iacr.org/2004/304.pdf

7. Biham E. A Framework for Iterative Hash Functions: HAIFA. / Eli Biham, Orr Dunkelman // Second cryptographic hash workshop. – 2006. – 9 с. – Режим доступу до ресурсу : http://csrc.nist.gov/pki/HashWorkshop/2006/Papers/DUNKELMAN_NIST3.pdf

8. 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

Лужецький Володимир Андрійович – д. т. н., професор, завідувач кафедри захисту інформації.

Баришев Юрій Володимирович – магістр з інформаційної безпеки, аспірант кафедри захисту інформації. E-mail: yuriy.baryshev@gmail.com.

Вінницький національний технічний університет.