ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

УДК 004.056 : 004.424.47

В. А. ЛУЖЕЦЬКИЙ, Ю. В. БАРИШЕВ

МЕТОДИ БАГАТОКАНАЛЬНОГО КЕРОВАНОГО ХЕШУВАННЯ ДЛЯ КОМП’ЮТЕРНОЇ КРИПТОГРАФІЇ

Вступ

Повторюваність перетворень, що використовуються в хешуванні, дозволяє зловмиснику виконувати попередню підготовку до атак [1] та будувати мультиколізії [2], що віддаляє всі ітеративні хеш-функції від ідеальної за показниками стійкості до пошуку колізій та прообразів. Саме тому для протидії таким атакам необхідні принципові зміни – зміни у самій концепції хешування, а не в перетвореннях, за допомогою яких це хешування реалізується. Вирішення цієї проблеми може бути здійснено шляхом реалізації концепції керованого хешування [3].

Водночас тенденція до розпаралелення обчислень в комп’ютерних системах знайшла своє відображення й у комп’ютерній криптографії, тому сучасне хешування повинно бути природним для платформ, що використовують паралельні обчислювальні ядра, тобто бути багатоканальним. При цьому багатоканальність у хешуванні підвищує вразливість хеш-функцій до спеціалізованих атак, що використовують мультиколізії [2], тому хеш-функції, що будуть отримані в результаті реалізації концепції керованого хешування, повинні бути багатоканальними та водночас стійкими до мультиколізій для того, щоб відповідати сучасним вимогам до процесу хешування.

У зв’язку з вищевикладеним метою даного дослідження є підвищення стійкості багатоканального керованого хешування за рахунок розробки нових методів та конструкцій хешування.

Для досягнення мети необхідно розв’язати такі задачі:

- аналіз відомих атак на багатоканальне хешування та методів протидії ним;
- аналіз методів формування вектора керування у керованому хешуванні та його використання при розпаралеленні;
- розробка методів стійкого багатоканального керованого хешування;
- наведення прикладів конструкцій, розроблених відповідно до запропонованих методів.

Аналіз атак, що використовують властивості конструкцій хешування, та методів протидії ним

Першою атакою на багатоканальне хешування стала атака Жукса. Об’єктом цієї атаки є конструкція "каскадування" хеш-функцій [4], яка передбачала паралельне і незалежне обчислення в кожному з каналів вихідного хеш-значення за допомогою конструкції Меркля-Дамгаарда. Тобто для двох каналів ця конструкція мала такий вигляд:

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

де $h_i^{(j)}$ – проміжне хеш-значення, визначене після i-ї ($i = \overline{1,l}$) ітерації у j-му каналі ($j = \overline{1,q}$); $f^{(j)}$ – функція ущільнення, що реалізується j-м каналом; m_i – i-й блок даних.

Вихідне хеш-значення пропонувалось отримувати шляхом конкатенації вихідних хеш-значень кожного каналу [4], тобто для прикладу (1) вихідним хеш-значенням буде $h_l^{(1)} \,||\, h_l^{(2)}$ (де "||" – позначення конкатенації). Атака Жукса передбачала знаходження мультиколізій для таких конструкцій, тобто для кожного блоку m_i Жукс знаходив інший m_i^*, такий що виконувалась рівність:

$$h_i^{(1)} = f^{(1)}\left(h_{i-1}^{(1)}, m_i\right) = f^{(1)}\left(h_{i-1}^{(1)}, m_i^*\right). \quad (2)$$

Таким чином, для повідомлення довжиною l блоків даних побудувалося 2^l таких колізій за $l \cdot 2^{n/2}$ ітерацій хешування. Цю сукупність колізій Жукс назвав мультиколізією. Існує ймовірність, якою не можна знехтувати і яка зростає зі збільшенням l того, що серед знайдених колізій для першого каналу знайдеться хоча б одна, яка породжує колізію й в іншому каналі [2].

Після появи атаки Жукса, закладені ним підходи були узагальнені та розвинуті, зокрема, в роботі [2] наводиться приклад для побудови мультиколізій для деревоподібних конструкцій хешування та для

хеш-функцій, які передбачають перестановку блоків даних для кожного з каналів хешування. Отже, з'явився клас атак, загальних для більшості ітеративних хеш-функцій, який використовував саме цю властивість ітеративності.

Однією з перших спроб протидії мультиколізіям стала конструкція подвійного каналу, запропонована Штефаном Люксом у роботі [5]. Ця конструкція передбачала взаємозв'язок каналів хешування після кожної ітерації, реалізований у такий спосіб [5]:

$$\begin{cases} h_i^{(1)} = f\left(h_{i-1}^{(1)}, h_{i-1}^{(2)} \,||\, m_i\right) \\ h_i^{(2)} = f\left(h_{i-1}^{(2)}, h_{i-1}^{(1)} \,||\, m_i\right) \end{cases}. \tag{3}$$

Після останньої ітерації передбачався додатковий раунд для отримання вихідного хеш-значення, що реалізує таке перетворення [5]:

$$h = f\left(h', h_l^{(1)} \,||\, h_l^{(2)} \,||\, \overline{0}\right), \tag{4}$$

де h' – частина вектора ініціалізації; $\overline{0}$ – рядок нулів, довжиною $\left\|m_i\right\| - n$ бітів ($\left\|m_i\right\|$ – довжина блока даних в бітах).

Недоліком методу Люкса, який описується формулами (3) та (4), є подвійні витрати обчислювальних ресурсів порівняно з одноканальним хешуванням, оскільки останнє перетворення (4) фактично зменшує вихідне хеш-значення, отримане після завершення l-го раунду. Відповідно, не зважаючи на стійкість конструкції подвійного каналу Люкса до мультиколізій, у цьому підході фактично нівелюється сенс розпаралелення процесу хешування – тобто маніпулювання із даними меншої розрядності та збільшення швидкості процесу.

Ще одним суттєвим недоліком реалізації ідеї Люкса є його бажання пристосувати запропоновану ним конструкцію до відомих хеш-функцій замість того, щоб розробляти нову авторську хеш-функцію на базі запропонованого ним підходу. Саме тому формула (3) передбачає "домішування" проміжного хеш-значення з іншого каналу до вхідних даних за допомогою конкатенації, що, в свою чергу, зменшує кількість бітів даних, що хешуються за один раунд, а тому збільшує тривалість процесу хешування. Крім того, перетворення (4) передбачає використання ще одного початкового заповнення, відповідно для двох каналів з вихідним значенням довжиною n бітів необхідно $3n$ бітів для вектора ініціалізації ($h_0^{(1)} \,||\, h_0^{(2)} \,||\, h'$), тобто вектор ініціалізації повинен бути втричі довшим, ніж у класичних конструкціях одноканального хешування з вихідним значенням такої самої довжини.

У роботі [6] було запропоновано методи, що узагальнюють та надають подальшого розвитку підходам Люкса, суть яких описується такою формулою:

$$\begin{cases} h_i^{(1)} = f^{(1)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, \ldots, h_{i-1}^{(q)}, m_i\right) \\ h_i^{(2)} = f^{(2)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, \ldots, h_{i-1}^{(q)}, m_i\right) \\ \ldots \\ h_i^{(q)} = f^{(q)}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, \ldots, h_{i-1}^{(q)}, m_i\right) \end{cases}. \tag{5}$$

Як результуюче хеш-значення хеш-функції конструкції (5) пропонується використовувати конкатенацію хеш-значень, отриманих у кожному з каналів $h_l^{(1)} \,||\, h_l^{(2)} \,||\, \ldots \,||\, h_l^{(q)}$ [6]. Оскільки за допомогою кожного каналу визначається частина хеш-значення довжиною n/q бітів, то довжина вектора ініціалізації становить n бітів, як і в одноканальному хешуванні. Водночас менша розрядність операндів у каналах хешування дозволяє швидше виконувати неприродні для обчислювальних платформ перетворення такі, як піднесення до степеня за модулем.

Крім узагальнення та удосконалення методу Люкса у роботі [6] пропонується новий підхід, який передбачає перестановки блоків даних в межах каналів та зчеплення кожного i-го блока даних з іншим, індекс якого залежить від i-го, що формалізується такою конструкцією:

$$\begin{cases} r_i^{(1)} = rand\left(m_i^{(1)}\right) \\ r_i^{(2)} = rand\left(m_i^{(2)}\right) \\ \ldots \\ r_i^{(q)} = rand\left(m_i^{(q)}\right) \\ h_i^{(1)} = f^{(1)}\left(h_{i-1}^{(1)}, m_i^{(1)} * m_{i-r_i^{(1)}}^{(1)}\right) \\ h_i^{(2)} = f^{(2)}\left(h_{i-1}^{(2)}, m_i^{(2)} * m_{i-r_i^{(2)}}^{(2)}\right) \\ \ldots \\ h_i^{(q)} = f^{(q)}\left(h_{i-1}^{(q)}, m_i^{(q)} * m_{i-r_i^{(q)}}^{(q)}\right) \end{cases}, \quad (6)$$

де $r_i^{(j)}$ – псевдовипадкове відхилення від номера поточного блока даних, що обробляється, яке отримують у j-му каналі; $rand(\cdot)$ – деяка функція генерування псевдовипадкових чисел; "*" – операція об’єднання двох операндів в один.

Основною перевагою підходу, що описується конструкцією (6), є незалежність каналів один від одного, а тому більша швидкість порівняно з конструкцією (5) [6].

Попри існування підходів стійкого багатоканального хешування в межах концепції ітеративного хешування, у науково-технічній літературі невідомі підходи стійкого розпаралелення в межах концепції керованого хешування, особливістю якого є наявність додаткового параметра (вектора керування), який необхідно враховувати при розпаралеленні. Перед тим як враховувати цей параметр необхідно проаналізувати способи його формування, оскільки від його зміни від ітерації до ітерації залежить вибір тієї чи іншою функції ущільнення, тобто сама реалізація концепції керованості. Якщо ж зловмисник зможе передбачити правило зміни вектора керування, то для нього задача зламу матиме складність аналогічну ітеративному хешуванню.

Аналіз методів формування вектора керування

У методах хешування, що будуть розроблятися, кожен канал повинен реалізовувати кероване хешування, яке передбачає невизначеність для зловмисника у раундовому перетворенні, відповідно, щоб це було реалізовано в повній мірі, кожен вектор керування повинен обиратися з однаковою ймовірністю. Крім того, якщо кожен канал буде реалізовувати своє раундове перетворення, то це збільшить невизначеність для зловмисника у перетворенні, що здійснюється, оскільки йому необхідно буде розглядати всі варіанти комбінування перетворень у різних каналах. Якщо для кожного повідомлення будуть створюватися свої параметри перетворення, тоді зловмиснику буде складніше виконати атаку з попередньою підготовкою, адже, змінивши блок повідомлення, він змінить також дії, які в подальшому будуть виконуватись над наступними блоками даних. При ключовому варіанті хешування вектор керування повинен бути закритим від зловмисника, однак правила його формування можуть бути відкритими.

Отже, вектори керування повинні задовольняти таким вимогам.

1. Значення векторів керування обираються з однаковою ймовірністю.
2. Вектори керування мають бути різними для різних каналів.
3. Вектори керування мають бути різними для різних повідомлень.
4. Бути невідомим зловмиснику (для ключового варіанта хешування).

Вектори керування можуть формуватися на основі:

- псевдовипадкових чисел;
- блоків даних;
- проміжних хеш-значень.

Використання різних для кожного каналу послідовностей псевдовипадкових чисел як векторів керування дозволить забезпечити вимоги 1, 2. Забезпечення вимоги 4 можна досягти, якщо початковий стан генераторів буде частиною ключової інформації (або всім ключем). Однак псевдовипадкові числа не залежать від даних, що хешуються, а тому послідовність векторів керування для кожного каналу буде однаковою для різних повідомлень, якщо ключ лишатиметься незмінним, отже цей варіант не відповідає вимозі 3.

Розподіл значень блоків даних, у загальному випадку, прагне до рівномірного закону, тому теоретично, при їх використанні як векторів керування, вони відповідають вимозі 1, хоча на практиці це

не обов'язково виконується. Використання значень блоків даних для формування векторів керування забезпечує виконання вимоги 3. Вимога 2 може бути забезпечена, якщо виконувати перестановки блоків даних для кожного каналу. Однак ключове хешування може бути реалізовано лише в певних специфічних задачах, коли дані, що хешуються, невідомі зловмиснику, що суттєво обмежує галузь застосування хешування, в якому блоки даних використовуються як вектори керування.

Розподіл проміжних хеш-значень, за визначенням, повинен прагнути до рівномірного закону, чим забезпечується виконання вимоги 1. Якщо кожен канал буде керуватися різними наборами проміжних хеш-значень, то й послідовність дій буде унікальною для кожного каналу, а отже можна забезпечити виконання вимоги 2. Відповідність вимозі 3 забезпечується тим, що проміжні хеш-значення залежать від блоків даних. Якщо вектор керування буде невідомим зловмиснику, що природно для хешування, то відповідність вимозі 4 буде забезпечена. Таким чином, проміжні хеш-значення задовольняють усім висунутим вище вимогам до векторів керування.

Методи багатоканального керованого хешування

Використаємо позначення, запропоноване в статті [6], для багатоканального хешування (Multi-Pipe Hashing) з деякими змінами – $MPHq(k,d,g,z)$, де q - кількість каналів, k – кількість каналів від проміжних хеш-значень яких залежить наступне хеш-значення j-го $(j=1,2,...,q)$ каналу, d – кількість блоків даних, які беруть участь у формуванні хеш-значення у j-му каналі, g – режим хешування, ($g=1$ – ітеративне некероване хешування, $g=2$ – використовуються псевдовипадкові операнди, $g=3$ – використовуються операції, що залежать від блоків даних, $g=4$ – кероване хешування, $g=0$ – можливий будь-який з режимів, $g=x_1/x_2$ – комбінація режимів x_1 та x_2, де x_1, $x_2\in\{2;3;4\}$, $x_1\neq x_2$), z – кількість псевдовипадкових чисел, що використовуються під час однієї ітерації.

Для підвищення стійкості методів хешування до мультиколізій пропонується використовувати залежність одного з каналів хешування від інших, однак на відміну від метода, що використовує проміжні хеш-значення з інших каналів як операнди (5), методи керованого хешування можуть використовувати їх як вектори керування. Для двох каналів метод хешування $MPH2(1,1,4,0)$ передбачатиме обробку кожного блоку даних відповідно такої формули:

$$\begin{cases} h_i^{(1)} = f_{h_{i-1}^{(2)}}\left(h_{i-1}^{(1)}, m_i\right) \\ h_i^{(2)} = f_{h_{i-1}^{(1)}}\left(h_{i-1}^{(2)}, m_i\right) \end{cases}. \tag{7}$$

Для узагальненого методу керованого хешування, що передбачає довільне хешування даних, можливі різні варіанти реалізації керування. Кожен з q каналів може використовувати $q-1$ проміжне хеш-значення з інших каналів як вектор керування, оскільки не варто використовувати проміжні хеш-значення, отримані в даному каналі, для його керування. Останнє пояснюється тим, що коли проміжне хеш-значення використовується як вектор керування та операнд одночасно, то кількість значень, що можуть бути отримані в результаті хешування, обмежується, оскільки при певному проміжному хеш-значенні завжди будуть виконуватись лише певні операції.

Перевагою керованого хешування над ітеративним при багатоканальній реалізації є те, що зв'язок між каналами можна виконати "опосередковано", тобто метод, що параметрично описується $MPHq(1,1,4,0)$, реалізує хешування за такою конструкцією:

$$\begin{cases} h_i^{(1)} = f_{v_i^{(1)}}\left(h_{i-1}^{(1)}, m_i\right) \\ h_i^{(2)} = f_{v_i^{(2)}}\left(h_{i-1}^{(2)}, m_i\right) \\ \ldots \\ h_i^{(q)} = f_{v_i^{(q)}}\left(h_{i-1}^{(q)}, m_i\right) \end{cases}, \tag{8}$$

де $v_i^{(j)}$ – вектор керування, що використовується у j-му каналі під час i-ї ітерації та визначається за формулою:

$$v_i^{(j)} = \sum_{u=1,u\neq j}^{q} h_{i-1}^{(u)} . \quad (9)$$

Причому, замість додавання у формулі (9) може бути використана інша операція, що забезпечує рівномірний вплив кожного операнда на вихідне значення, наприклад побітове додавання.

Можливості багатоканального керованого хешування дозволяють виконувати керування за допомогою всіх $(q-1)$ каналів хешування. Однак для пришвидшення хешування можливий метод хешування, який би використовував як вектор керування проміжні хеш-значення з меншої кількості каналів за умови забезпечення "опосередкованого" впливу кожного каналу на інші. Це може бути досягнуто за рахунок того, що вплив одного каналу на інші відбувається певною затримкою на кілька ітерацій, що для задач, де відбувається хешування великих масивів даних, допустиме. Так, наприклад, для керування кожного каналу за допомогою одного каналу формулу (9) замінимо такою:

$$v_i^{(j)} = h_{i-1}^{(j-1)} . \quad (10)$$

Крім зв'язку проміжних хеш-значень за допомогою вектора керування, методи керованого хешування можуть використовувати підходи до підвищення стійкості, аналогічні запропонованим у роботі [6]. Так проміжні хеш-значення з декількох каналів можуть бути використані як операнди, якщо вони не використовуються в даному каналі для формування вектора керування. Наприклад, метод хешування $MPHq(2,1,4,0)$ може мати вигляд:

$$\begin{cases} h_i^{(1)} = f_{h_{i-1}^{(q)}}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, m_i\right) \\ h_i^{(2)} = f_{h_{i-1}^{(1)}}\left(h_{i-1}^{(2)}, h_{i-1}^{(3)}, m_i\right) \\ \ldots \\ h_i^{(q)} = f_{h_{i-1}^{(q-1)}}\left(h_{i-1}^{(q)}, h_{i-1}^{(1)}, m_i\right) \end{cases} . \quad (11)$$

Очевидно, що до конструкції (11) можна і надалі додавати проміжні хеш-значення і як операнди, і як складові вектора керування. Однак додаткові операнди потребують додаткової обробки, що негативно впливає на швидкість хешування. Відповідно в низці випадків краще забезпечувати підвищення стійкості методу хешування до мультиколізій за допомогою зв'язку блоків даних. Метод хешування $MPHq(1,2q,3/4,q)$ буде мати вигляд аналогічний методу підвищення стійкості некерованого хешування (6):

$$\begin{cases} r_i^{(1)} = rand\left(m_i^{(1)}\right) \\ r_i^{(2)} = rand\left(m_i^{(2)}\right) \\ \ldots \\ r_i^{(q)} = rand\left(m_i^{(q)}\right) \\ h_i^{(1)} = f_{h_{i-1}^{(q)}}^{(1)}\left(h_{i-1}^{(1)}, m_i^{(1)} * m_{i-r_i^{(1)}}^{(1)}\right) \\ h_i^{(2)} = f_{h_{i-1}^{(1)}}^{(2)}\left(h_{i-1}^{(2)}, m_i^{(2)} * m_{i-r_i^{(2)}}^{(2)}\right) \\ \ldots \\ h_i^{(q)} = f_{h_{i-1}^{(q-1)}}^{(q)}\left(h_{i-1}^{(q)}, m_i^{(q)} * m_{i-r_i^{(q)}}^{(q)}\right) \end{cases} . \quad (12)$$

Існує низка задач, коли необхідно обчислювати хеш-значення для масиву даних, що динамічно генерується, тобто сторона, яка виконує хешування не має всіх вхідних даних в момент, коли необхідно починати процес. Щоб зробити використання хешування можливим для таких задач пропонується застосовувати для кожного каналу свою функцію генерування псевдовипадкових чисел та обирати псевдовипадкові відхилення в межах певного ковзного "вікна" даних фіксованої довжини. З урахуванням цього метод хешування $MPHq(1,2q,3/4,q)$, що використовує конструкцію (12), зміниться на $MPHq(1,1+q,3/4,q)$, а ітерація хешування матиме такий вигляд:

$$\begin{cases} r_i^{(1)} = rand^{(1)}(m_i) \bmod w \\ r_i^{(2)} = rand^{(2)}(m_i) \bmod w \\ \ldots \\ r_i^{(q)} = rand^{(q)}(m_i) \bmod w \\ h_i^{(1)} = f_{h_{i-1}^{(q)}}\left(h_{i-1}^{(1)}, m_i * m_{i-r_i^{(1)}}\right) \\ h_i^{(2)} = f_{h_{i-1}^{(1)}}\left(h_{i-1}^{(2)}, m_i * m_{i-r_i^{(2)}}\right) \\ \ldots \\ h_i^{(q)} = f_{h_{i-1}^{(q-1)}}\left(h_{i-1}^{(q)}, m_i * m_{i-r_i^{(q)}}\right) \end{cases}, \quad (13)$$

де w – кількість блоків даних, що потрапляють до "вікна" даних.

У конструкції (13) псевдовипадкові відхилення можуть приймати значення в меншому діапазоні чисел, однак за рахунок того, що "вікно" даних є ковзним, тобто після ітерації в його межі потрапляє новий блок даних, то всі блоки даних поєднуються у вигляді кільця. Обробка даних при реалізацій конструкції буде починатися з $(w+1)$го блока даних, а завершуватиметься w-м.

Метод зв’язку блоків даних може бути використаний декілька разів, оскільки його ефект залежить від кількості пов’язаних однин з одним блоків даних. Однак для останнього випадку необхідно генерувати значну кількість псевдовипадкових чисел. Для зменшення часу на це пропонується використовувати комбінування підходів до генерування псевдовипадкових чисел, що використовувались у (12) та (13):

$$\begin{cases} r_i^{(j)(1)} = rand^{(1)}\left(m_i^{(j)}\right) \\ r_i^{(j)(2)} = rand^{(2)}\left(m_i^{(j)}\right) \\ \ldots \\ r_i^{(j)(z)} = rand^{(z)}\left(m_i^{(j)}\right) \end{cases}. \quad (14)$$

Відповідно хешування $MPHq(1, z+1, 3/4, zq)$ відбуватиметься за таким ітеративним правилом:

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

Не зважаючи на те, що метод хешування, який використовує конструкцію (15), вимагає великої кількості генераторів псевдовипадкових чисел та повинен мати меншу швидкість порівняно з методами, що передбачатимуть обробку даних відповідно формул (12) та (13), що може зменшити його прикладне застосування. Тому для випадків, коли неможливе використання декількох різних генераторів псевдовипадкових чисел пропонується використовувати такий підхід до генерування псевдовипадкових чисел:

$$\begin{cases} r_i^{(j)(1)} = rand\left(m_i^{(j)}\right) \\ r_i^{(j)(2)} = rand\left(m_{i+1}^{(j)}\right) \\ \ldots \\ r_i^{(j)(z)} = rand\left(m_{i+z-1}^{(j)}\right) \end{cases}. \quad (16)$$

Крім переваги у зменшені кількості генераторів метод хешування, що використовуватиме генерацію псевдовипадкових чисел за формулою (16), дозволить зменшити кількість ітерацій, якщо кожна наступна ітерація передбачатиме обробку блоків даних з $(i+z)$-го до $(i+2\cdot z-1)$-го.

Висновки

Поява нових атак, що використовують властивість ітеративності операцій у конструкціях, спричинила необхідність у пошуку нових методів хешування, які передбачатимуть використання конструкцій з керованими параметрами операцій. Водночас сьогодення вимагає від комп’ютерних систем розпаралелення обчислень, від хешування – багатоканальності.

Аналіз відомих методів хешування виявив методи стійкого багатоканального хешування, які, проте, не враховують специфіки керованого хешування – наявності вектора керування. З проведеного аналізу методів формування векторів керування випливає, що найкращим є метод формування векторів керування на основі проміжних хеш-значень, оскільки розподіл їх значень є рівномірним, вони є різними для різних каналів та різних повідомлень, а також цей метод є зручним при ключовому хешуванні.

З урахуванням обраного методу формування вектора керування розроблені методи багатоканального керованого хешування, що дозволяють виконувати зв’язок між каналами хешування за допомогою вектора керування. Останнє дозволяє побудувати конструкції, які "опосередковано" пов’язують канали, тобто без участі проміжного хеш-значення як операнда, чого не може бути при класичному ітеративному хешуванні. Водночас дані методи багатоканального керованого хешування дозволяють, як і методи багатоканального ітеративного хешування, здійснювати "безпосередній" зв’язок між каналами, що підтверджується наведеними прикладами конструкцій для реалізації запропонованих методів. Крім цього, в межах концепції керованого хешування можливе використання зав’язування блоків даних з іншими блоками даних, номер яких залежить від даних, що виявив себе швидшим, ніж метод зв’язку каналів за допомогою проміжних хеш-значень. Автори бачать перспективу подальших досліджень у розробці та реалізації функцій ущільнення для запропонованих методів.

Література

1. Kelsey J. Herding hash functions and the Nostradamus attack / John Kelsey, Tadayoshi Kohno.– 2005. – 18 с. – Режим доступу до ресурсу : http://archives.scovetta.com/pub/crypto/Nostradamus%20Attack.pdf
2. Hoch J. J. Breaking the ICE – Finding Multicollisions in Iterative Concatenated and Expanded (ICE) Hash Functions / Jonathan J. Hoch, Adi Shamir. – 2006. – 13. – Режим доступу до ресурсу : http://www.wisdom.weizmann.ac.il/~yaakovh/papers/hashpaper_submission.pdf
3. Баришев Ю. В. Підхід до хешування, що стійке до аналізу зловмисника. / Ю. В. Баришев // Системи обробки інформації. – №3(84). – 2010. – С. 99-100
4. Preneel B. Analysis and Design of Cryptographic Hash Functions. PhD thesis / Bart Preneel. – Katholieke Universiteit Leuven, 1993. – 338 с. – Режим доступу до ресурсу : http://homes.esat.kuleuven.be/~preneel/phd_preneel_feb1993.pdf
5. Lucks S. Design Principles for Iterated Hash Functions / Stefan Lucks // Cryptology ePrint Archive. – 2004. – 22 с. Режим доступу до ресурсу : http://eprint.iacr.org/2004/253.pdf
6. Лужецький В. А. Конструкції хешування стійкі до мультиколізій / В. А. Лужецький, Ю. В. Баришев // Наукові праці ВНТУ. – №1. – 2010. – 8 с. – Режим доступу до статті : http://www.nbuv.gov.ua/e-journals/vntu/2010_1/2010-1.files/uk/10valsam_ua.pdf

Стаття надійшла до редакції 11.10.2010.

Відомості про авторів

Лужецький Володимир Андрійович – д.т.н, професор, завідувач кафедри захисту інформації;
Баришев Юрій Володимирович – аспірант кафедри захисту інформації.
Вінницький національний технічний університет