УДК 004.056: 004.424.47

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

МЕТОДИ ТА ЗАСОБИ ПАРАЛЕЛЬНОГО КЕРОВАНОГО ХЕШУВАННЯ

У цій статті представлено аналіз підходів до розробки керованих криптографічних примітивів. Розроблені конструкції керованого багатоканального хешування. Визначені криптографічні примітиви для реалізації функцій ущільнення в хешуванні. Запропоновані підходи до синтезу цих функцій та методи хешування, які використовують такі підходи. Наведені оцінки швидкості програмної реалізації цих методів.

Ключові слова: *хешування, криптографічні примітиви, керованість, багатоканальність, функції ущільнення.*

Вступ

Виникнення нових атак на функції хешування [1] спричинило потребу у перегляді підходів не лише до проектування функцій ущільнення, але й до переосмислення суті, хешування. Це обумовило актуальність розробки нових концепцій хешування. Одним із напрямків удосконалення хешування є концепція керованого хешування [2], яка передбачає зміну параметрів перетворень у функціях ущільнення від ітерації до ітерації. Оскільки відомі методи криптоаналізу як об'єкт дослідження використовують функції ущільнення, параметри яких не змінюється від ітерації до ітерації, то від реалізації керованого хешування очікується підвищення стійкості.

Для криптографічних методів і методів хешування, зокрема, актуальним є забезпечення високої швидкості обробки даних, що досягається шляхом розпаралелення обчислень [3]. Концепція керованого хешування дозволяє розпаралелити процес хешування, а, отже, зменшити його тривалість [4]. Саме тому дослідження у напрямку реалізації концепції керованого хешування є актуальними.

Мета цього дослідження є підвищення швидкості хешування за рахунок реалізації концепції керованого хешування.

Для досягнення мети необхідно вирішити такі завдання:

– аналіз відомих методів керованого хешування;

– розробка конструкцій керованого хешування;

– обґрунтування вибору криптографічних примітивів та синтез функцій ущільнення;

– розробка методів хешування та їхня програмна реалізація.

Аналіз відомих методів керованого хешування

У функціях хешування Dynamic SHA та Dynamic SHA-2, які запропоновані для участі у конкурсі на новий стандарт хешування SHA-3 [5, 6], реалізовано один із підходів до побудови керованого хешування, що базується на використанні керованих зсувів. Для створення множини функцій ущільнення у роботі [5] пропонується використовувати логічні функції аналогічні до функцій, що використовуються у стандарті хешування SHA-2 [7]. Як керований параметр перетворення використовується кількість бітів, на яку відбувається зсув. При цьому цей параметр пропонувалось визначати залежно від кількох модифікованих вхідних блоків даних. Саме це стало причиною недостатньої стійкості Dynamic SHA та Dynamic SHA-2. Автори роботи [6] запропонували атаку на ці хеш-функції, яка використовує здатність криптоаналітика впливати на керування, нав'язуючи певні блоки даних.

Відомий підхід до побудови керованих функцій хешування на основі підстановочно-перестановочних мереж (ППМ), запропонований у роботі [8] та розвинений у роботах [9, 10]. Зокрема, у роботі [8] приводяться класи криптографічних примітивів, приклади шифрів та

хеш-функцій. Приклад примітивів класу $F_{2/1}$ (два вхідних / вихідних біти даних, один керуючий) в алгебраїчній нормальній формі наведено у таблиці 1 [8].

Таблиця 1

Приклад керованих криптографічних примітивів класу $F_{2/1}$

№	Примітив	№	Примітив	№	Примітив
1	$x_1v \oplus x_2v \oplus x_1$	9	$x_1v \oplus x_2 \oplus v$	17	$x_1v \oplus x_2v \oplus x_1 \oplus v \oplus 1$
2	$x_2v \oplus x_1$	10	$x_1v \oplus x_2 \oplus x_1 \oplus v$	18	$x_2v \oplus x_1 \oplus v \oplus 1$
3	$x_1v \oplus x_2v \oplus x_2$	11	$x_1v \oplus x_2v \oplus x_2 \oplus v$	19	$x_1v \oplus x_2 \oplus x_1 \oplus 1$
4	$x_2v \oplus x_1 \oplus x_2$	12	$x_2v \oplus x_2 \oplus x_1 \oplus v$	20	$x_1v \oplus x_2 \oplus 1$
5	$x_1v \oplus x_2$	13	$x_2v \oplus x_2 \oplus x_1 \oplus v \oplus 1$	21	$x_2v \oplus x_1 \oplus x_2 \oplus 1$
6	$x_1v \oplus x_2 \oplus x_1$	14	$x_1v \oplus x_2v \oplus x_2 \oplus v \oplus 1$	22	$x_1v \oplus x_2v \oplus x_2 \oplus 1$
7	$x_2v \oplus x_1 \oplus v$	15	$x_1v \oplus x_2 \oplus x_1 \oplus v \oplus 1$	23	$x_2v \oplus x_1 \oplus 1$
8	$x_1v \oplus x_2v \oplus x_1 \oplus v$	16	$x_1v \oplus x_2 \oplus v \oplus 1$	24	$x_1v \oplus x_2v \oplus x_1 \oplus 1$

Попарно комбінуючи такі примітиви автори, роботи [8] визначають найкращі комбінації, використовуючи критерії збалансованості вихідних значень та нелінійності перетворень, отримуючи ППМ класу $F_{2/1}$. Аналогічно у роботі [8] отримують ППМ класів $F_{2/2}$ та $F_{3/1}$. У роботі [9] наведені керовані функції, що мають два, три та чотири керуючих біти для двох вхідних / вихідних бітів даних, а у роботі [10] – обернені перетворення для таких примітивів.

Недоліком підходу ППМ є те, що вони першочергово були орієнтовані на розробку шифрів, а тому криптографічні примітиви не враховують особливості хешування. Наприклад, для хешування не обов'язкова наявність криптографічного примітива, що здійснює обернене перетворення, на відміну від шифрування, де ця вимога є обов'язковою. Крім того, функції, запропоновані в роботах [8 – 10], першочергово орієнтовані на апаратну реалізацію, де перестановка двох сусідніх бітів блока даних виконується шляхом розгалуження провідників. Водночас програмна реалізація цих примітивів ускладнюється тим, що універсальні мікропроцесори не пристосовані до такого роду перетворень. Саме тому їх необхідно реалізовувати, використовуючи резервне копіювання змінної, її зсув, накладання маски, що зменшує швидкість обчислення логічних функцій.

Отже, відомі функції ущільнення для методів керованого хешування мають недоліки, пов'язані з недостатніми стійкістю або швидкістю програмної реалізації. Саме тому необхідно розробити нові функції ущільнення, для чого спочатку необхідно розробити моделі ітерацій хешування, які враховуватимуть розпаралелення обчислень.

Розробка конструкцій керованого хешування

Оскільки процес хешування повинен забезпечувати хешування даних довільної довжини, він повинен бути ітеративним [8]. Ітеративність обумовлює обробку даних блоками та їх зчеплення з проміжним хеш-значенням, отриманим на попередній ітерації. Для розпаралелення пропонується використовувати декілька каналів хешування (обчислювачів, що реалізують функцію ущільнення). Для забезпечення стійкого зав'язування каналів один з одним необхідно, щоб обчислення у кожному з каналів на кожній ітерації враховували проміжні хеш-значення, отримані у інших каналах [4]. Однак кероване хешування дозволяє зав'язувати канали один з одним як шляхом використання канальних проміжних хеш-значень як аргументів, так і за допомогою вектора керування:

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

де $h_i^{(j)}$ – проміжне хеш-значення, отримане у j-му каналі ($j = \overline{1, q}$) на i-ій ітерації ($i = \overline{1, l}$); m_i – i-ий блок даних; $f_{v_i^{(j)}}(\cdot)$ – функція ущільнення, що забезпечує сталу довжину вихідного значення; $v_i^{(j)}$ – вектор керування, який визначає параметри перетворення функції ущільнення $f_{v_i^{(j)}}(\cdot)$, у j-му каналі на i-ій ітерації ($i = \overline{1, l}$); $g(\cdot)$ – функція формування вектора керування.

Узагальнимо конструкцію (1) для k проміжних хеш-значень, що використовуються як аргумент функції $f_{v_i^{(j)}}(\cdot)$, та ϕ проміжних хеш-значень – як аргумент функції $g(\cdot)$:

$$\begin{cases} h_i^{(1)} = f_{v_i^{(1)}}\left(h_{i-1}^{(1)}, h_{i-1}^{(2)}, \ldots, h_{i-1}^{(k)}, m_i\right) \\ h_i^{(2)} = f_{v_i^{(2)}}\left(h_{i-1}^{(2)}, h_{i-1}^{(3)}, \ldots, h_{i-1}^{(k+1)}, m_i\right) \\ \ldots \\ h_i^{(q)} = f_{v_i^{(q)}}\left(h_{i-1}^{(q)}, h_{i-1}^{(1)}, \ldots, h_{i-1}^{(k-1)}, m_i\right) \\ v_i^{(1)} = g\left(h_{i-1}^{(q)}, h_{i-1}^{(q-1)}, \ldots, h_{i-1}^{(q-\phi+1)}\right) \\ v_i^{(2)} = g\left(h_{i-1}^{(1)}, h_{i-1}^{(q)}, h_{i-1}^{(q-1)}, \ldots, h_{i-1}^{(q-\phi+2)}\right) \\ \ldots \\ v_i^{(q)} = g\left(h_{i-1}^{(q-1)}, h_{i-1}^{(q-2)}, \ldots, h_{i-1}^{(q-\phi)}\right) \end{cases}, \quad (2)$$

де $k < (q - \phi)$.

Для реалізації конструкції багатоканального керованого хешування (2) визначимо криптографічні примітиви, за допомогою яких синтезуємо множину функцій ущільнення.

Множина функцій ущільнення для керованого хешування

Для того, щоб методи керованого хешування дозволяли реалізувати швидке виконання цього процесу, необхідно, щоб операції, які використовуються як криптографічні примітиви, були "зручними" для універсальних мікропроцесорів. До таких операцій належать: додавання за модулем 2^n (+); виключне або ($\oplus$); інвертування (~); логічне множення ($\wedge$); логічне додавання ($\vee$); циклічний зсув на u біт праворуч (>>> u); простий зсув на u біт праворуч / ліворуч (>> u/ << u). Водночас криптографічні операції повинні забезпечувати однаковий вплив кожного біта вхідних даних на результат перетворення, а операція простого зсуву (>> u/ << u) не дозволяє виконати цю умову. Саме тому операції >> u та << u не можуть бути використані для побудови функцій ущільнення в керованому хешуванні. Параметром керованих перетворень пропонується обрати кількість бітів u, на яку циклічно зсувається

змінна праворуч, оскільки його зміна є простою і швидко виконується, при програмній реалізації. Крім того, кожен керований параметр пропонується використовувати як частину вектора керування.

Як основу для розробки функцій ущільнення пропонується використати логічні функції, які застосовані у стандарті SHA-2 [7], та модифікувати їх для врахування саме керованого хешування. Так кожен канал буде реалізовувати перетворення ($k = 2$):

$$\begin{aligned} h_i^{(j)} = &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j)} >>> u_i^{(j)(h1)}\right)\oplus \\ \oplus &\left(\sim m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j+1)} >>> u_i^{(j)(h2)}\right) \end{aligned}, \tag{3}$$

де $u_i^{(j)(xk)}$ – кількість бітів, на яку зсувається змінна x у к -ій позиції у функції ущільнення на i-ій ітерації в j-му каналі хешування.

У функції ущільнення (3) використовується три керовані параметри, відповідно вектор керування є конкатенацією цих параметрів $v_i^{(j)} = u_i^{(j)(m1)} \,\|\, u_i^{(j)(h1)} \,\|\, u_i^{(j)(h2)}$.

Для забезпечення більшого впливу аргументів функції ущільнення на вихідне значення пропонується модифікувати ще одну функцію, що використовується у стандарті [7]:

$$\begin{aligned} h_i^{(j)} = &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j)} >>> u_i^{(j)(h1)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j+1)} >>> u_i^{(j)(h2)}\right)\oplus \\ \oplus &\left(h_{i-1}^{(j)} >>> u_i^{(j)(h1)} \wedge h_{i-1}^{(j+1)} >>> u_i^{(j)(h2)}\right) \end{aligned}. \tag{4}$$

Для функції (4) вектор керування не зміниться порівняно з функцією (3). При збільшенні кількості аргументів у функціях ущільнення крім операції $\wedge$ пропонується використовувати $\vee$. Наприклад, для випадку $k = 4$ функція ущільнення (3) набуде вигляду:

$$\begin{aligned} h_i^{(j)} = &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j)} >>> u_i^{(j)(h1)}\right)\oplus \\ \oplus &\left(\sim m_i >>> u_i^{(j)(m1)} \wedge h_i^{(j+1)} >>> u_i^{(j)(h2)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m2)} \vee h_{i-1}^{(j+2)} >>> u_i^{(j)(h3)}\right)\oplus \\ \oplus &\left(\sim m_i >>> u_i^{(j)(m2)} \vee h_i^{(j+3)} >>> u_i^{(j)(h4)}\right) \end{aligned}. \tag{5}$$

Функція ущільнення (4) при значенні параметра $k = 4$ зміниться так:

$$\begin{aligned} h_i^{(j)} = &\left(m_i >>> u_i^{(j)(m1)} \wedge h_{i-1}^{(j)} >>> u_i^{(j)(h1)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m1)} \wedge h_i^{(j+1)} >>> u_i^{(j)(h2)}\right)\oplus \\ \oplus &\left(h_{i-1}^{(j)} >>> u_i^{(j)(h1)} \wedge h_i^{(j+1)} >>> u_i^{(j)(h2)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m2)} \vee h_{i-1}^{(j+2)} >>> u_i^{(j)(h3)}\right)\oplus \\ \oplus &\left(m_i >>> u_i^{(j)(m2)} \vee h_i^{(j+3)} >>> u_i^{(j)(h4)}\right)\oplus \\ \oplus &\left(h_{i-1}^{(j+2)} >>> u_i^{(j)(h3)} \vee h_i^{(j+3)} >>> u_i^{(j)(h4)}\right) \end{aligned}. \tag{6}$$

При збільшенні параметра k пропонується застосовувати підхід аналогічний до використаного при побудові функцій ущільнення (5) та (6) з функцій (3) та (4) відповідно.

Функції ущільнення (3 – 6) були використані для програмної реалізації керованого хешування мовою С та скомпільовані за допомогою середовища програмування Visual Studio 2005. Отримані програми передбачали використання однакових процедур формування вектора керування та ініціалізації. Тестування програм відбувалося на комп'ютері з процесором Intel Pentium 4 з частотою 3 ГГц та обсягом оперативної пам'яті 2 Гб. Для оцінювання швидкості хешування використано стандартну бібліотеку time.h. Оцінки тривалості хешування даних цими методами керованого хешування для 16 каналів, що

оперують даними довжиною 16 біт (вихідне хеш-значення – 256 біт), наведено в таблиці 2.

Таблиця 2

Оцінки тривалості хешування

Функція ущільнення	Тривалість хешування, cycles			
	10 Кб	100 Кб	1 Мб	10 Мб
(3)	78	781	8219	82328
(4)	141	1516	15297	151813
(5)	94	812	8547	84718
(6)	171	1719	17406	171719

Розроблені програми передбачають обробку даних довільної довжини та формування для них вихідного хеш-значення, довжина якого кратна розрядності каналу.

Висновок

Потреба в досягненні більшої швидкості хешування та водночас необхідність забезпечення стійкості цього процесу до атак породжують необхідність пошуку нових концепцій хешування, до яких належить концепція керованого хешування. Побудова методів керованого хешування за допомогою криптографічних примітивів, виконання яких природне для універсальних мікропроцесорів, дозволило досягти високих показників швидкості, що збільшуються лінійно при зростанні обсягу хешованих даних, у той час як відомі методи досягають найкращих показників швидкості при хешуванні тільки великих масивів вхідних даних. Саме тому найбільш доречним використанням запропонованих у цій статті методів є хешування малих обсягів даних, зокрема у протоколах автентифікації.

СПИСОК ЛІТЕРАТУРИ

1. Second Preimage Attacks on Dithered Hash Functions [Електронний ресурс] / E. Andreeva, C. Bouillaguet, P.-A. Fouque and others. – 19 с. – Режим доступу: http://homes.esat.kuleuven.be/~eandreev/eurocrypt08.pdf.

2. Баришев Ю. В. Підхід до хешування, що стійке до аналізу зловмисника / Ю. В. Баришев // Системи обробки інформації. – № 3. – 2010. – С. 99 – 100.

3. Корченко А. Г. Построение систем защиты информации на нечетких множествах. Теория и практические решения / А. Г. Корченко. – К.: "МК-Пресс", 2006. – 320 с.

4. Баришев Ю. В. Методи побудови швидкого хешування / Ю. В. Баришев // Методи та засоби кодування, захисту й ущільнення інформації. Тези другої Міжнародної науково-практичної конференції, м. Вінниця, 22 – 24 квітня 2009 року. – Вінниця: ВНТУ, 2009. – С. 138 – 139.

5. Zijie Xu. Dynamic SHA [Електронний ресурс] / Zijie Xu // Cryptology ePrint Archive. – 2007. – 34 с. – Режим доступу: http://eprint.iacr.org/2007/476.pdf.

6. Aumasson J.-P. Cryptanalysis of Dynamic SHA(2) [Електронний ресурс] / J.-P. Aumasson, O. Dunkelman, S. Indesteege and B. Preneel // COSIC publications. – 2009. – 18 с. – Режим доступу: https://www.cosic.esat.kuleuven.be/publications/article-1277.pdf.

7. Secure Hash Standard: Federal Information Processing Publication Standard Publication 180-3 [Електронний ресурс] – Gaithersburg, 2008. – 27 с. – Режим доступу: http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.

8. Молдовян Н. А. Криптография: от примитивов к синтезу алгоритмов. / Н. А. Молдовян, А. А. Молдовян, М. А. Еремеев. – СПб.: БХВ-Петербург, 2004. – 448 с.

9. Рудницький В. М. Модель уніфікованого пристрою криптографічного перетворення інформації / В. М. Рудницький, В. Г. Бабенко // Системи обробки інформації. – № 3. – 2009. – С. 91 – 95.

10. Рудницький В. М. Синтез математичних моделей пристроїв декодування інформації для криптографічних систем / В. М. Рудницький, В. Г. Бабенко // Системи обробки інформації. – № 2. – 2009. – С. 124 – 128.

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.