КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2(22), 2023
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2023.22.8495
УДК 004.056

Селезньов Віталій Ігорович
асистент кафедри захисту інформації
Вінницький національний технічний університет, Вінниця, Україна
ORCID 0009-0004-0225-9697
seleznov.vitalii@email.com

Лужецький Володимир Андрійович
Доктор технічних наук, професор, завідувач кафедри захисту інформації
Вінницький національний технічний університет, Вінниця, Україна
ORCID 0000-0001-7466-7738
lva.kzi2002@gmail.com

МЕТОД МАЛОРЕСУРСНОГО ГЕШУВАННЯ ТИПУ «ДАНІ — ГЕНЕРАТОР»

Анотація. Створення безпечної та ефективної структури криптографічного алгоритму є однією з ключових криптографічних задач. Останнім часом криптографія для малоресурсних пристроїв привернула значну увагу світових науковців. Велика частина досліджень присвячена дослідженню методів блокового шифрування, і навпаки, існує значно менше публічно оприлюднених пропозицій щодо методів малоресурсного гешування. Багато геш-функцій, рекомендованих для застосування у малоресурсних пристроях відомими організаціями зі стандартизації використовують за основу блокове шифрування, що дозволяє забезпечити достатній рівень безпеки, однак потребує значних обчислювальних ресурсів, що є критичним для використання у подібних пристроях. Актуальність дослідження методів малоресурсного гешування даних полягає у необхідності забезпечення достатнього рівня безпеки геш-функції з мінімальним використанням обчислювальних ресурсів, шляхом внесення модифікацій у процес гешування. В статті виконано огляд відомих підходів до побудови геш-функцій будь-якої складності та аналіз останніх досліджень та публікацій присвячених малоресурсному гешуванню, на основі яких обрано структуру та підхід до побудови методу малоресурсного гешування даних. Запропоновано новий метод малоресурсного гешування, що базується на структурі Меркла-Демґарда та використовує ітеративний байт-орієнтований підхід. Наведено формалізований опис процесу малоресурсного гешування за новим методом. Виконано статистичне тестування запропонованого методу відповідно до NIST SP 800-22. У вигляді узагальненої структурної схеми представлено апаратну реалізацію запропонованого методу малоресурсного гешування. Складність запропонованої апаратної реалізації розраховано в умовних одиницях [GE] для реалізацій обчислення геш-значень розрядності 128, 192 та 256 біт. Виконано порівняння запропонованого методу гешування типу «дані — генератор» з відомими малоресурсними геш-функціями з точки зору апаратних витрат.

Ключові слова: криптографічний алгоритм; малоресурсна криптографія; геш-функція; метод гешування; апаратна складність.

ВСТУП

Постановка проблеми. За останні роки значно зріс попит на використання та розробку Інтернету речей. Складовими Інтернету речей є RFID, смарт-карти, бездротові сенсорні мережі та багато інших систем, що мають обмежені обчислювальні можливості, але підключені до мережі та взаємодіють з іншими пристроями. Системи RFID використовуються для автоматичної ідентифікації об’єктів, таких як транспорт, продукти, люди та тварини. RFID також використовують у системах безпеки, наприклад, пропускного контролю та платіжних системах, що використовують принцип безконтактної оплати. Система RFID складається з мітки, яка прикріплюється до об’єкта,

та зчитувача, що використовується для зчитування даних з мітки, на основі яких виконується ідентифікація об'єкта. Зв'язок між зчитувачем і міткою потребує захисту від несанкціонованого доступу. Бездротова сенсорна мережа складається з ряду сенсорних вузлів, які працюють без втручання людини протягом тривалого часу з невеликим енергоспоживанням. Для бездротових сенсорних мереж необхідно забезпечити безпеку даних, що передаються між сенсорними вузлами з врахуванням обмеженого енергопостачання. Для пристроїв з обмеженими ресурсами Інтернету речей дуже гострою є проблема безпеки даних, що обробляються та передаються подібними пристроями. Традиційні криптографічні алгоритми не можуть бути реалізовані на цих пристроях, оскільки пристрої з обмеженими ресурсами мають обмежений обсяг пам'яті, обмежену обчислювальну потужність та обмежене енергоспоживання. Для забезпечення таких вимог використовують так звану малоресурсну криптографію. Криптографічна спільнота виконала значний обсяг роботи в цій галузі, зокрема розробку, реалізацію та криптоаналіз нових «легких» криптографічних алгоритмів разом із ефективним впровадженням звичайних алгоритмів криптографії в середовищах з обмеженими ресурсами та обчислювальними можливостями [1].

Один з найважливіших криптографічних перетворень є функція гешування. Гешування широко використовується для таких цілей як підтвердження цілісності даних в електронному цифровому підписі та цифрових сертифікатах, електронних валютах, різних протоколах автентифікації користувачів та повідомлень, комп'ютерних системах контролю цілісності тощо.

Аналіз останніх досліджень і публікацій. Дослідженню методів та засобів малоресурсної криптографії присвячені роботи значної кількості закордонних та українських науковців. Незважаючи на те, що більшість наукових робіт присвячені алгоритмам шифрування [1], [2], дослідженням малоресурсного гешування починають займатись все більше науковців [1] – [7]. У статті [3] автори досліджують останні розробки та найсучасніші реалізації легких криптографічних геш-функцій, класифікують тенденції структури процесу малоресурсного гешування. Також представлено порівняльний аналіз різних апаратних і програмних реалізацій геш-функцій на основі дев'яти різних показників, що включають вимоги до безпеки, продуктивності та апаратної складності алгоритмів гешування. У [4] представлено малоресурсну геш-функцію криптографії для великих даних та додатків Інтернету речей. У запропонованій конструкції використовуються функції S-Box, лінійного перетворення та перестановки бітів. За попередніми результатами тестувань представлена структура функції показує чудові результати, однак алгоритм не перевірявся на різних апаратних платформах. Порівняння класичних методів, запропонованих NIST для пристроїв з обмеженими ресурсами висвітлено у роботі [5]. Порівняння здійснювалось на основі ключових метрик, що впливають на ефективність та можливість використання методів для малоресурсної криптографії. Байт — орієнтований підхід до побудови геш-функцій запропоновано у статті [6]. У дослідженні представлено структурні схеми спеціалізованого процесора, що реалізує запропонований підхід для двох функцій гешування. Запропоновані методи гешування забезпечують апаратну реалізацію, складність якої відповідає вимогам міжнародного стандарту ISO/IEC FDIS 29192, що свідчить про доцільність використання подібного підходу для реалізації малоресурсних пристроїв.

Останні дослідження та публікації, присвячені малоресурсному гешуванню, демонструють, що проблема забезпечення оптимального співвідношення між безпекою, продуктивністю та вимогами до апаратних витрат залишається невирішеною в повному обсязі, оскільки досі не представлено єдиний стандарт малоресурсного гешування, що

забезпечить граничну апаратну складність реалізації у 2000 *GE* (gate equivalent) [8]. Серед розглянутих публікацій найменшу складність апаратної реалізації забезпечують геш-функції з байт — орієнтованим підходом та конструкції з використанням функції S-Box, тому один із запропонованих конструкцій доцільно взяти за основу структури нової малоресурсної геш-функції [4], [6].

Метою статті є зменшення апаратних витрат на реалізацію засобу малоресурсного гешування шляхом модифікації структури геш-функції.

Об'єктом дослідження є процес криптографічного гешування даних.

Предметом дослідження є метод та засіб малоресурсного гешування.

ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

Відомі підходи, щодо побудови гаш-функцій малоресурсній криптографії базуються на конструкціях Меркла-Демґарда, Девіса-Меєра, Матіаса-Меєра-Осеаса, Міягучі-Пренеля.

Відповідно до конструкції Меркла-Демґарда вхідні дані геш-функції представляються у вигляді масиву інформації $M = \{m_1, m_2, ..., m_n\}$, що ділиться на блоки рівної довжини. Процес гешування є ітераційним з реалізацією:

$$h_i = f(m_i; h_{i-1}) \quad (1)$$

де h_i — результат гешування *i*-го блоку даних з масиву M;

f — функція ущільнення, що повертає перетворене повідомлення фіксованої довжини.

У такій конструкції результатом гешування повідомлення буде h_l. Якщо ж необхідне виконання гешування з ключем, то в якості ключа зазвичай беруть значення початкового стану h_0. Таку будову використовують відомі геш-функції Lesamnta-LW, ARMADILLO [2], [9].

Також, геш-функції часто будують на основі блокових шифрів. Блокові шифри, подібно до односторонніх функції ущільнення, отримують на вхід ключ і відкритий текст, фіксованої довжини, та повертають зашифровані дані тієї ж довжини. Однак, маючи зашифровані дані та ключ, на основі якого виконувалось зашифрування можна виконати дешифрування та знайти значення вхідних даних. Тому, щоб блоковий шифр використовувати як функцію ущільнення, необхідно виконати додаткові операції. Структури Девіса-Меєра, Матіаса-Меєра-Осеаса, Міягучі-Пренеля базуються на використанні блокових шифрів для формування функції одноблокового ущільнення. Функції одноблокового ущільнення виводять дані довжиною результату зашифрування, тоді як функції ущільнення двоблокової довжини виводять дані довжини, що дорівнює подвоєній довжині виходу блокового шифру [3].

У структурі Девіса-Мейєра повідомлення m_i використовується як ключ, а попереднє геш-значення h_{i-1} — як відкриті дані, що подаються на вхід блокового шифру E_i. Отриманий результат зашифрування додається за модулем два з результатом попередньої ітерації гешування h_{i-1} для отримання наступного геш-значення h_i:

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

Недоліком конструкції Девіса-Мейєра є те, що незважаючи на безпечний блоковий шифр, можна обчислити так звані «нерухомі точки». Тобто для будь-якого значення m можна знайти значення h таке, що $E_m(h) + h = h$ [10].

ISSN 2663 - 4023

Структура Матіса-Мейєра-Осеаса є покращенням структури Девіса-Мейєра. У цій структурі повідомлення m_i використовується як відкриті дані, а попереднє геш-значення h_{i-1} проходить додаткове перетворення g та використовується як ключ, що подаються на входи блокового шифру E_i. Геш-значення обчислюється за формулою:

$$h_i = E_{g(h_{i-1})}(m_{i-1}) + m_{i-1} \quad (3)$$

де g — функція перетворення, що може бути різною залежно від реалізації.

Подібна структура може бути використана, якщо блоки даних і ключ шифрування однакового розміру [2], [3].

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ
МЕТОД ГЕШУВАННЯ

Автори пропонують метод малоресурсного гешування, що базується на конструкції Меркла-Демґарда, суть якого полягає в такому.

Вхідне повідомлення m подається у вигляді послідовності L байт:

$$m = \{m_1, m_2, ..., m_L\}. \quad (4)$$

Початкове геш-значення h_0 є сукупністю псевдовипадкових байтів та представлено у вигляді послідовності:

$$h_0 = \{h_{0,0}, h_{0,1}, ..., h_{0,k-1}\}, \quad (5)$$

де $k = \frac{l}{8}$, l — довжина геш-значення в бітах.

Проміжні геш-значення $h_i = \{h_{i,0}, h_{i,1}, ..., h_{i,k-1}\}$ обчислюються на основі попереднього геш-значення h_{i-1} і поточного значення m_i шляхом виконання функції ущільнення f. Кроки виконання функції перетворень для *i*-го байту даних:

1. Для j від 0 до $k-1$ виконувати:
   1.1. Згенерувати псевдовипадковий біт $g_{i,j}$.
   1.2. Виконати обчислення:

$$h_{i,j} = (h_{i-1,j} + m_i \cdot g_{i,(k-1-j)}) \bmod 256, \quad (6)$$

Схему обрахунку функції ущільнення за методом гешування наведено на рис. 1.

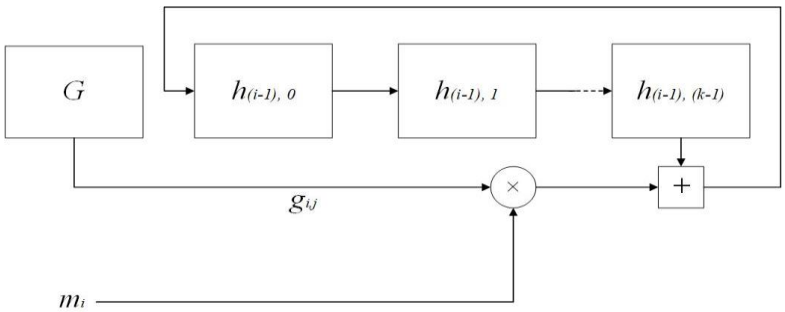

Рис. 1. Схема обчислення геш-значень

Результат *L*-го ущільнення f є остаточним геш-значенням $h = \{h_{L,0}, h_{L,1}, ..., h_{L,k-1}\}$ для вхідного повідомлення m.

У запропонованому методі проміжні та остаточні значення геш-функції обчислюється шляхом додавання байтів даних у позиції, що відповідають $g_{i,j}=1$. Якщо $g_{i,j}=0$, то додавання не відбувається. Враховуючи те, що геш-значення залежать від даних та від бітів $g_{i,j}$, що генеруються псевдовипадковим чином, такий підхід до гешування будемо називати «дані — генератор», а запропонований метод гешування — *Hash* «*Data — Generator*», що скорочено *HDG*.

Формалізований опис геш-функції типу «дані генератор» подано у вигляді теоретико-множинної моделі:

$$\mathbf{HDG}=\{\mathbf{M},\ \mathbf{H},\ \mathbf{H}_0,\ \mathbf{G},\ \mathbf{G}_0,\ \mathbf{O}\}, \quad (7)$$

де $\mathbf{M}=\{m\}$ — множина вхідних даних;

$\mathbf{H}=\{h\}$ — множина вихідних даних;

$\mathbf{H}_0=\{h_0\}$ — початкове геш-значення;

$\mathbf{G}=\{g_i\}$ — множина псевдовипадкових чисел;

$\mathbf{G}_0=\{g_0\}$ — початковий стан генератора псевдовипадкових чисел;

$\mathbf{O}=\{f;g;\&;+\}$ — множина виконуваних операцій.

Початковим геш-значенням h_0 може бути будь-який l — розрядний двійковий код. Початковий стан генератора псевдовипадкових чисел k — розрядний двійковий код. Виконуваними операціями є: операції функції ущільнення f, операції функції генератора g на основі регістра зсуву з лінійним зворотнім зв'язком, логічне множення $\&$ та додавання за модулем 256.

СТАТИСТИЧНЕ ТЕСТУВАННЯ МЕТОДУ

Тестування розробленого методу гешування проведено на основі пакету тестів NIST STS 800-22 [12]. Набір тестів містить 14 статистичних тестів, призначених для перевірки гіпотези випадковості для двійкових послідовностей будь-якої довжини. Всі тести спрямовані на виявлення різних дефектів псевдовипадковості.

Тестування розробленого методу проводилось на основі даних сформованих шляхом обчислення 200000 геш-значень функції *HDG* для різних повідомлень довжини *L = 8, 32, 64* байтів.

Для тестування геш-функції *HDG* використано такі тести:

1. The Frequency (Monobit) Test;
2. Frequency Test within a Block;
3. The Runs Test;
4. Tests for the Longest-Run-of-Ones in a Block;
5. The Binary Matrix Rank Test;
6. The Discrete Fourier Transform (Spectral) Test;
7. The Non-overlapping Template Matching Test;
8. The Overlapping Template Matching Test;
9. The Linear Complexity Test;
10. The Serial Test;
11. The Approximate Entropy Test;
12. The Cumulative Sums (Cusums) Test;
13. Random Excursions Test;

14. Random Excursions Variant Test;

Результати статистичного тестування наведено у таблиці 1.

Таблиця 1

Результати статистичного тестування

Номер тесту	*L = 8*		*L = 32*		*L = 64*	
	Результат тесту	***Тест пройдено***	***Результат тесту***	***Тест пройдено***	***Результат тесту***	***Тест пройдено***
1	97/100	+	99/100	+	100/100	+
2	0/100	-	99/100	+	98/100	+
3	98/100	+	100/100	+	100/100	+
4	43/100	-	99/100	+	99/100	+
5	99/100	+	100/100	+	100/100	+
6	0/100	-	100/100	+	98/100	+
7	95/100	-	95/100	-	98/100	+
8	98/100	+	97/100	+	99/100	+
9	99/100	+	98/100	+	100/100	+
10	97/100	+	98/100	+	99/100	+
11	93/100	-	99/100	+	100/100	+
12	98/100	+	99/100	+	99/100	+
13	9/9	+	7/7	+	13/13	+
14	9/9	+	7/7	+	13/13	+

Відповідно до NIST SP 800-22 тест вважається успішним, якщо його успішно пройдено для понад 96% вхідних даних. Аналіз табл. 1 показує, що при *L = 8* байт, успішними є лише 9 тестів. Для *L = 32* успішними є усі тести, крім *7*-го тесту. Для *L = 64* усі тести є успішними. На основі цього можна зробити висновок про те, що метод *HDG* забезпечує потрібні статистичні характеристики геш-значень лише для повідомлень довжини 64 байти і більше.

Малоресурсна криптографія зазвичай передбачає використання даних невеликої довжини (<64 байт). Для забезпечення потрібних статистичних характеристик геш-значень для даних такої довжини пропонується використання функції ущільнення f над *i*-м байтом повідомлення *n* разів. Результати статистичного тестування для модифікованої геш-функції *HDG* представлено у таблицях 2 та 3.

Аналіз таблиці 2 показує, що для довжини повідомлення *L = 4* реалізація функції ущільнення f не один раз, а чотири рази забезпечує три успішні тести, а реалізація функції вісім разів забезпечує чотири успішні тести. Щоб забезпечити успішне проходження усіх тестів для повідомлення цієї довжини необхідно реалізувати функцію ущільнення 128 разів. Таким чином для виконання вимог щодо статистичних властивостей геш-значень, у разі довжини повідомлень 32 біти потрібно реалізувати функцію ущільнення f не менше 128 разів.

Відповідно до таблиці 3 для довжини повідомлення *L = 8* реалізація функції ущільнення f чотири рази забезпечує усі успішні тести, окрім *7*-го. Реалізації функції ущільнення f шість та вісім разів забезпечують успішне виконання усіх тестів для такої довжини повідомлення. Таким чином потрібні статистичні властивості геш-значень, у разі довжини повідомлень 64 біти досягаються при виконанні функцію ущільнення f не менше 6 разів.

Таблиця 2

ISSN 2663 - 4023

Результати статистичного тестування методу з додаванням ітерацій для *L = 4* байти

Номер тесту	*n = 4*		*n = 8*		*n = 128*	
	Результат тесту	***Тест пройдено***	***Результат тесту***	***Тест пройдено***	***Результат тесту***	***Тест пройдено***
1	1/100	-	0/100	-	98/100	+
2	3/100	-	8/100	-	97/100	+
3	16/100	-	0/100	-	99/100	+
4	18/100	-	10/100	-	99/100	+
5	100/100	+	98/100	+	98/100	+
6	60/100	-	45/100	-	100/100	+
7	96/100	+	97/100	+	99/100	+
8	95/100	-	85/100	-	99/100	+
9	99/100	+	97/100	+	98/100	+
10	94/100	-	98/100	+	100/100	+
11	39/100	-	37/100	-	98/100	+
12	1/100	-	0/100	-	100/100	+
13	0	-	0	-	17/17	+
14	0	-	0	-	17/17	+

Таблиця 3

Результати статистичного тестування методу з додаванням ітерацій для *L = 8* байт

Номер тесту	*n = 4*		*n = 6*		*n = 8*	
	Результат тесту	***Тест пройдено***	***Результат тесту***	***Тест пройдено***	***Результат тесту***	***Тест пройдено***
1	98/100	+	98/100	+	98/100	+
2	96/100	+	99/100	+	97/100	+
3	97/100	+	100/100	+	99/100	+
4	99/100	+	100/100	+	99/100	+
5	99/100	+	98/100	+	100/100	+
6	99/100	+	97/100	+	99/100	+
7	95/100	-	96/100	+	97/100	+
8	100/100	+	99/100	+	98/100	+
9	100/100	+	100/100	+	99/100	+
10	98/100	+	98/100	+	98/100	+
11	100/100	+	100/100	+	99/100	+
12	97/100	+	98/100	+	98/100	+
13	7/7	+	8/8	+	16/16	+
14	7/7	+	7/8	+	17/17	+

АПАРАТНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНОГО МЕТОДУ

Структурну схему пристрою, що реалізує гешування за методом *HDG* представлено на рис. 2. Пристрій складається з k 8-розрядних регістрів Rg, суматора за модулем 256, 8-елементного блоку логічних елементів «І» Block AND та генератора псевдовипадкових чисел G.

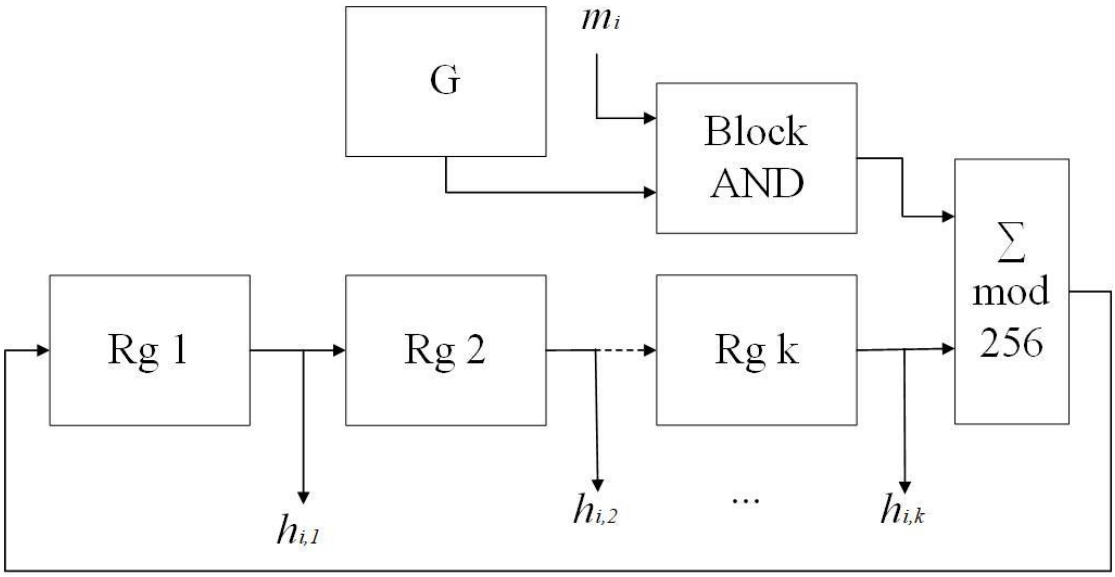

Рис. 2. Узагальнена структурна схема пристрою

Загальна апаратна складність пристрою обчислюється за формулою:

$$S_{\text{ПР}} = S_{\text{G}} + S_{\text{SUM}} + S_{\text{Rg}} + S_{\text{Block}}, \quad (8)$$

де S_{G} — складність генератора G;

S_{SUM} — складність суматора за модулем 256;

S_{Rg} — сумарна складність регістрів Rg 1 — Rg *k*;

S_{Block} — складність блоку елементів «І».

Генератор складається з 32 *D*-тригерів (*D Flip flop*) та 4 елементів *XOR*, тому:

$$S_{\text{G}} = 32D + 4XOR, \quad (9)$$

Суматор за модулем 256 складається з 8-ми однорозрядних суматорів, кожен з яких, у свою чергу, складається з двох елементів *AND*, одного *OR* та двох *XOR*.

$$S_{\text{SUM}} = 8(2AND + 1OR + 2XOR). \quad (10)$$

Кожен регістр Rg складається з восьми *D*-тригерів, тому їх сумарна складність:

$$S_{\text{Rg}} = 8k(D). \quad (11)$$

Складність блоку елементів «І»:

$$S_{\text{Block}} = 8AND. \quad (12)$$

З урахуванням формул (9–12) маємо таку апаратну складність:

$$S_{\text{ПР}} = (8k + 32)D + 20XOR + 24AND + 8OR. \quad (13)$$

У разі реалізації пристрою у вигляді мікросхеми за технологією 0.18 μm з використанням бібліотеки UMCL18G212T3 [13] маємо таку складність в умовних одиницях *GE*:

$$S_{\text{ПР}} = 42{,}64 \cdot k + 266{,}52. \quad (14)$$

У таблиці 4 наведено оцінки апаратної складності реалізації відомих малоресурсних геш-функцій і запропонованої геш-функції *HDG*. Порівняльний аналіз апаратних складностей показує, що для довжин геш-значень 128, 192 та 256 біт запропонована геш-функція *HDG* для своєї реалізації вимагає менших апаратних витрат ніж усі відомі геш-функції. Для довжини геш-значення 128 біт зменшення апаратних витрат складає від 1 до 18%, а для довжини геш-значення 256 біт — 15 до 19,5%.

Таким чином запропонована геш-функція *HDG* задовольняє вимогам щодо апаратної складності засобів малоресурсної криптографії.

Таблиця 4

Апаратна складність геш-функцій

Геш-функція	Структура	Довжина геш-значення, біт	Апаратна складність GE
HVH	На основі блокового шифру VH	160	1385
HVH		128	1145
LHash	П-Губка	128	1028
D-Quark		176	1702
U-Quark		136	1379
PHOTON-224/32/32		224	1736
PHOTON-128/16/16		128	1122
SPONGENT-128/128/8		256	1950
SPONGENT-256/256/16		128	1060
HDG (k = 32)	Меркла-Демґарда	256	1631
HDG (k = 24)		192	1290
HDG (k = 16)		128	949
PRNS		256	1872
PRNS		128	960
SNP		256	1944
SNP		128	984

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Особливість запропонованого методу гешування полягає в тому, що проміжні та остаточні геш-значення обчислюється шляхом додавання байтів даних у позиції, що визначаються символом 1 в коді, згенерованим генератором псевдовипадкових чисел, тому такий підхід до гешування названо «дані — генератор» (*Hash* «*Data — Generator*», що скорочено *HDG*). Тобто, це є байт-орієнтованим підходом до гешування даних, і, як наслідок, процес гешування не потребує доповнення (padding) останнього блоку даних, коли його довжина є меншою, ніж задана довжина блоків.

Результати статистичних тестів показали, що реалізація функції ущільнення лише один раз забезпечує потрібні статистичні властивості геш-значень для повідомлень довжиною більше 64 байтів. Для багатьох задач малоресурсної криптографії довжина повідомлень не повинна перевищувати 8 байт, тому для забезпечення потрібних статистичних властивостей геш-значень необхідно реалізовувати функцію ущільнення не менше шести разів.

Запропонована байт-орієнтована геш-функція *HDG* задовольняє вимогам щодо апаратної складності засобів малоресурсної криптографії. Результати порівняльного аналізу апаратних складностей відомих малоресурсних геш-функцій та запропонованої геш-функції показали, що геш-функція *HDG* для своєї реалізації вимагає менших апаратних витрат ніж усі відомі геш-функції. Для довжини геш-значень 128 біт зменшення апаратних витрат складає від 1 до 18%, а для довжини геш-значень 256 біт — від 15 до 19,5%.

Подальші дослідження будуть спрямовані на зменшення кількості реалізацій функції ущільнення з метою зменшення часу гешування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Buchanan, W., Li, S., Asif, R. (2017). Lightweight cryptography methods. *Journal of Cyber Security Technology*, *1(3–4)*, 187–201. https://doi.org/10.1080/23742917.2017.1384917
2. Mileva, A., et al. (2021). Catalog and Illustrative Examples of Lightweight Cryptographic Primitives. *Security of Ubiquitous Computing Systems*, 21–47. https://doi.org/10.1007/978-3-030-10591-4_2
3. Windarta, S., et al. (2022). Lightweight Cryptographic Hash Functions: Design Trends, Comparative Study, and Future Directions. *IEEE Access*. https://doi.org/ 10.1109/ACCESS.2022.3195572
4. Al-Odat, Z., Al-Qtiemat, E., & Khan, S. (2020) .An Efficient Lightweight Cryptography Hash Function for Big Data and IoT Applications. *2020 IEEE Cloud Summit*. https://doi.org/10.1109/ieeecloudsummit48914.2020.00016
5. Селезньов, В. (2023). Аналіз методів малоресурсного гешування. *LII науково-технічна конференція підрозділів ВНТУ: матеріали наук. конф*. https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/download/18664/15557
6. Лужецький, В., Слободян, С., & Кисюк, Д. (2017). Методи байт-орієнтованого хешування даних низькоресурсної криптографії. *Інформаційні технології та комп'ютерне моделювання: матеріали міжнар. наук.-практ. конф.*, 216–219. https://itcm.comp-sc.if.ua/2017/Luzhetskyi2.pdf
7. Widhiara, B., Kurniawan, Y., Susanti, B. (2023). RM70. A Lightweight Hash Function. IAENG *International Journal of Applied Mathematics, 53(1).* https://www.iaeng.org/IJAM/issues_v53/issue_1/IJAM_53_1_12.pdf
8. *Information technology – Security techniques – Lightweight cryptography – Part 5: Hash-functions.* (29192-5). (2016)
9. Hammad, B., et al. (2017). A survey of Lightweight Cryptographic Hash Function. *International Journal of Scientific & Engineering Research, 8(7),* 806–814. https://www.ijser.org/researchpaper/A-survey-of-Lightweight-Cryptographic-Hash-Function.pdf
10. Permana, O., Susanti, B., & Christine, M. (2023). Fixed-point attack on Davies–Meyer hash function scheme based on SIMON, SPECK, and SIMECK algorithms. *VII International Conference "Safety Problems of Civil Engineering Critical Infrastructures" (Spceci2021).* https://pubs.aip.org/aip/acp/article-abstract/2508/1/020018/2878784/Fixed-point-attack-on-Davies-Meyer-hash-function?redirectedFrom=fulltext
11. Ількевич, Є., & Лужецький, В. (2020). Алгоритм «легкої» геш-функції. *XLIX Науково-технічна конференція підрозділів Вінницького національного технічного університету: тези наук.-практ. конф.*
12. Bassham, L., et al. (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications. *Gaithersburg, MD: National Institute of Standards and Technology*. https://doi.org/10.6028/nist.sp.800-22r1a
13. Poschmann, A. (2009). Lightweight cryptography cryptographic engineering for a pervasive world. https://eprint.iacr.org/2009/516.pdf

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (22), 2023
ISSN 2663 - 4023

Seleznov Vitalii Ihorovych
Post-graduate student of the Information Security Department
Vinnytsia National Technical University, Vinnytsia, Ukraine
ORCID 0009-0004-0225-9697
seleznov.vitalii@email.com

Luzhetskyi Volodymyr Andriiovych
Doctor of Science, professor, Head of Information Security Department
Vinnytsia National Technical University, Vinnytsia, Ukraine
ORCID 0000-0001-7466-7738
lva.kzi2002@gmail.com

METHOD OF LOW-RESOURCE HASHING TYPE "DATA — GENERATOR"

Abstract. Creating a secure and efficient structure of a cryptographic algorithm is one of the key cryptographic tasks. Recently, cryptography for low-resource devices has attracted considerable attention of world scientists. A significant portion of the research is dedicated to the examination of block encryption methods, and conversely, there are notably fewer publicly disclosed proposals for low-resource hashing methods. Many hash functions recommended for use in low-resource devices by well-known standardization organizations are based on block encryption, which offers a sufficient level of security but demands substantial computational resources—something critical for their application in such devices. The urgency of investigating low-resource data hashing methods stems from the need to guarantee an adequate level of hash function security while minimizing computational resource usage through adjustments to the hashing process. This article reviews established approaches to constructing hash functions of varying complexities and examines the latest research and publications focused on low-resource hashing. Based on this, the structure and approach for developing a low-resource data hashing method were chosen. A novel low-resource hashing method, founded on the Merkle-Damgård construction and utilizing an iterative byte-oriented approach, is introduced. The process of low-resource hashing, according to the new method, is formally described. Statistical testing of the proposed method was conducted in accordance with NIST SP 800-22. An overview of the hardware implementation of the proposed low-resource hashing method is presented in the form of a generalized structural diagram. The complexity of the proposed hardware implementation is quantified in conventional units [GE] for hash value calculations of 128, 192, and 256 bits. A comparison of the proposed "data-generator" type hashing method with established low-resource hash functions, in terms of hardware costs, is conducted.

Keywords: cryptographic algorithm; low-resource cryptography; hash function; hashing method; hardware complexity.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Buchanan, W., Li, S., Asif, R. (2017). Lightweight cryptography methods. *Journal of Cyber Security Technology*, *1(3–4)*, 187–201. https://doi.org/10.1080/23742917.2017.1384917
2. Mileva, A., et al. (2021). Catalog and Illustrative Examples of Lightweight Cryptographic Primitives. *Security of Ubiquitous Computing Systems*, 21–47. https://doi.org/10.1007/978-3-030-10591-4_2
3. Windarta, S., et al. (2022). Lightweight Cryptographic Hash Functions: Design Trends, Comparative Study, and Future Directions. *IEEE Access*. https://doi.org/ 10.1109/ACCESS.2022.3195572
4. Al-Odat, Z., Al-Qtiemat, E., & Khan, S. (2020) .An Efficient Lightweight Cryptography Hash Function for Big Data and IoT Applications. *2020 IEEE Cloud Summit*. https://doi.org/10.1109/ieeecloudsummit48914.2020.00016
5. Seleznev, V. (2023). Analysis of low-resource hashing methods. *LII scientific and technical conference of VNTU subdivisions: materials of sciences. conf*. https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/download/18664/15557
6. Luzhetskyi, V., Slobodian, S., & Kisyuk, D. (2017). Methods of byte-oriented data hashing of low-resource cryptography. *Information technologies and computer modeling: materials of the international science and practice conf.*, 216–219. https://itcm.comp-sc.if.ua/2017/Luzhetskyi2.pdf

7. Widhiara, B., Kurniawan, Y., Susanti, B. (2023). RM70. A Lightweight Hash Function. IAENG *International Journal of Applied Mathematics, 53(1).* https://www.iaeng.org/IJAM/issues_v53/issue_1/IJAM_53_1_12.pdf
8. *Information technology – Security techniques – Lightweight cryptography – Part 5: Hash-functions.* (29192-5). (2016)
9. Hammad, B., et al. (2017). A survey of Lightweight Cryptographic Hash Function. *International Journal of Scientific & Engineering Research, 8(7),* 806–814. https://www.ijser.org/researchpaper/A-survey-of-Lightweight-Cryptographic-Hash-Function.pdf
10. Permana, O., Susanti, B., & Christine, M. (2023). Fixed-point attack on Davies–Meyer hash function scheme based on SIMON, SPECK, and SIMECK algorithms. *VII International Conference "Safety Problems of Civil Engineering Critical Infrastructures" (Spceci2021).* https://pubs.aip.org/aip/acp/article-abstract/2508/1/020018/2878784/Fixed-point-attack-on-Davies-Meyer-hash-function?redirectedFrom=fulltext
11. Ilkevich, E., & Luzhetskyi, V. (2020). Algorithm of the "light" hash functionї. *XLIX Scientific and technical conference of subdivisions of the Vinnytsia National Technical University: scientific and practical theses. conf.*
12. Bassham, L., et al. (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications. *Gaithersburg, MD: National Institute of Standards and Technology*. https://doi.org/10.6028/nist.sp.800-22r1a
13. Poschmann, A. (2009). Lightweight cryptography cryptographic engineering for a pervasive world. https://eprint.iacr.org/2009/516.pdf