https://doi.org/10.31649/1997-9266-2025-183-6-99-112

УДК 004.056

В. А. Лужецький[1]
В. І. Селезньов[1]

ОГЛЯД ПІДХОДІВ ТА МЕТОДІВ МАЛОРЕСУРСНОГО ГЕШУВАННЯ ДАНИХ

[1]Вінницький національний технічний університет

Представлено огляд сучасних підходів до побудови геш-функцій, що є основним елементом криптографічного захисту в системах з обмеженими апаратними можливостями, зокрема в пристроях Інтернету речей (IoT), вбудованих мікроконтролерах та сенсорних мережах. Актуальність дослідження зумовлена інтенсивним розвитком цих технологій та необхідністю забезпечення високого рівня безпеки за мінімальних витрат ресурсів. Основною метою роботи є аналіз структурних особливостей, криптографічних властивостей та технічних характеристик засобів, що реалізують малоресурсні геш-функції. Розглядається два принципово різні підходи до побудови геш-функцій: ітеративна конструкція Меркла–Дамгарда та конструкцію «губка».

Дослідження охоплює аналіз 12 геш-функцій та їхніх сімейств, що використовують одну із зазначених конструкцій. Оцінка кожної з геш-функцій проводилася за основними характеристиками, такими як апаратна складність, довжина геш-значення та криптографічна стійкість, а також, пропускна здатність та енергоспоживання. Таким чином, отримано порівняльні дані, що дозволили виявити сильні та слабкі сторони кожної геш-функції в контексті їх реалізації для малоресурсних систем. Для узагальнення результатів запропоновано інтегральний коефіцієнт ефективності, який враховує вплив чотирьох ключових параметрів з відповідними вагами.

Порівняльний аналіз показав, що геш-функції на основі конструкції Меркла–Дамґарда, хоча й мають прийнятний рівень криптографічної стійкості, вимагають значних апаратних витрат, обмежуючи їхнє застосування у малоресурсних системах. Натомість, алгоритми, розроблені на основі конструкції «губка», продемонстрували найвищі значення інтегрального коефіцієнта ефективності, що підтверджує найкращий компроміс між безпекою, продуктивністю та апаратними витратами. Зокрема, SPONGENT-88 вирізняється мінімальними вимогами до апаратних ресурсів, а HVH забезпечує високу пропускну здатність та криптографічну стійкість за невеликих витрат. Геш-функції DeeR-Hash та Hash-One, які використовують комбіновані схеми, вимагають найменші апаратні витрати для 160-бітних геш-значень з високим рівнем криптографічної стійкості. Проведений аналіз засвідчує, що подальші дослідження у напрямі конструкцій «губка» та комбінованих схем з регістрами зсуву є найперспективнішими, оскільки саме ці підходи забезпечують найкраще співвідношення між криптографічною стійкістю, продуктивністю та апаратною ефективністю.

Ключові слова: малоресурсна криптографія, геш-функція, конструкція Меркла–Дамґарда, конструкція «губка», засіб гешування, криптографічні характеристики, технічні характеристики.

Вступ

Інтернет речей (Internet of Things, IoT) є одним з ключових напрямів сучасних досліджень у галузі комп’ютерних наук та інформаційних технологій. Широке впровадження IoT-рішень у таких сферах, як промисловість, охорона здоров’я, енергетика та інфраструктура, зумовило зростання інтересу до проблем безпеки цих систем. За прогнозами компанії IoT Analytics [1], до 2030 року кількість підключених до мережі IoT-пристроїв перевищить 40 мільярдів.

Серед таких пристроїв частина має апаратні характеристики, зіставні зі звичайними комп’ютерами, проте значна кількість IoT-пристроїв мають суттєві обмеження, серед яких можуть бути відносно малий обсяг пам’яті, обмежені обчислювальні можливості процесора та низьке енергоспоживання. Забезпечення безпеки в умовах таких обмежень становить серйозний виклик. Для пристроїв з достатніми ресурсами можуть застосовуватись традиційні криптографічні механізми, тоді як для пристроїв з обмеженнями потрібні спеціалізовані, малоресурсні рішення.

Одним із базових криптографічних примітивів, що використовується у більшості систем безпеки, є криптографічна геш-функція. Криптографічна геш-функція перетворює вхідні дані довільної довжини на вихід фіксованої довжини, генеруючи геш-значення, що використовується для контролю цілісності даних, у протоколах автентифікації, для формування цифрових підписів, генерування псевдовипадкових чисел, похідних ключів та в інших криптографічних механізмах [2]—[5]. У цій статті розглянуто особливості відомих малоресурсних криптографічних геш-функцій та подано порівняльні оцінки їх складності реалізації та криптографічної стійкості, оскільки вони відіграють важливу роль у забезпеченні безпеки пристроїв з обмеженими ресурсами.

Метою дослідження є отримання оцінок порівняльного аналізу сучасних методів малоресурсного криптографічного гешування.

За останні роки опубліковано значну кількість наукових робіт з оглядом і дослідженням криптографічних алгоритмів для пристроїв з обмеженими обчислювальними можливостями [6]—[13]. У роботі [6] подано порівняльний аналіз малоресурсних геш-функцій, побудованих на основі sponge-конструкцій. Оцінювання здійснено за шістьма критеріями: рівень безпеки, апаратна складність реалізації, розмір геш-значення, пропускна здатність, енергоспоживання та кількість тактів, необхідних для обчислення. Автори підкреслюють відсутність універсального рішення, яке б одночасно поєднувало високу безпеку та мінімальні апаратні витрати, що обґрунтовує необхідність подальшої розробки нових геш-функцій для застосування в умовах обмежених ресурсів.

У дослідженнях [7], [8] виконано аналіз можливостей застосування як універсальних криптографічних алгоритмів, так і спеціалізованих малоресурсних рішень у середовищі Інтернету речей. У [7] подано класифікацію малоресурсних шифрів за принципом побудови та подано їх порівняння за характеристиками, подібними до використаних у [6], зокрема з урахуванням апаратних і програмних витрат. У свою чергу, робота [7] узагальнює сучасний стан розвитку малоресурсних протоколів для IoT-мереж. Автори також проаналізували три основні примітиви малоресурсної криптографії: блокові шифри, потокові шифри та криптографію на еліптичних кривих, розглядаючи їхню придатність для використання в IoT-мережах з обмеженими ресурсами.

У роботі [10] розглядають застосування малоресурсних геш-функцій для автентифікації пристроїв у мережах Інтернету речей та бездротових сенсорних мереж (WSN). У дослідженні також проведено порівняння відомих малоресурсних геш-функцій за трьома характеристиками: апаратною складністю, пропускною здатністю та споживаною потужністю. Автори зосередились на поліпшенні характеристик продуктивності та споживаної потужності для апаратної реалізації геш-функції та пропонують вдосконалений алгоритм SPONGENT-88 з власною реалізацією на FPGA, при цьому реалізація забезпечує зменшене використання логічних елементів і тригерів, підвищену пропускну здатність на блок та оптимізоване співвідношення між послідовною та комбінаційною логікою. Також у роботі проведено масштабування архітектури для варіантів SPONGENT-128 та SPONGENT-224, що дозволяє підвищити рівень безпеки без значного зростання апаратних витрат.

Авторами публікації [11] проведено комплексний огляд сучасних «легких» криптографічних геш-функцій для пристроїв з обмеженими ресурсами. Автори класифікують тенденції проектування таких алгоритмів, аналізують їхні криптографічні властивості та стійкість до атак, а також порівнюють апаратні та програмні реалізації за різними характеристиками. У роботі проаналізовано та порівняно більшість відомих «легких» геш-функцій, представлених у літературі до 2022 року, зокрема за показниками продуктивності, споживаної потужності, апаратних витрат і криптографічної стійкості. У дослідженні наявні неточності та помилкові значення метрик для окремих геш-функцій, проте автори висвітлюють основні проблеми стандартизації показників безпеки й продуктивності, окремо розглядають підходи до оптимізації S-box та раундових функцій, а також пропонують напрямки майбутніх досліджень у галузі «легких» геш-функцій для ресурсозалежних пристроїв.

Питання аналізу малоресурсних геш-функцій частково розглянуто в роботі [12], де здійснено порівняння відомих малоресурсних методів, зокрема Quark, PHOTON, SPONGENT та Lesamnta-LW, за основними апаратними та програмними характеристиками. Результати показали, що навіть спеціально розроблені малоресурсні геш-функції не завжди є однаково доцільними для застосування на пристроях із різними обмеженнями, оскільки мають відмінності у швидкодії, апаратній складності та вимогах до пам’яті. Обмеженість розглянутого кола методів у [12] не дозволяє сформувати універсальні висновки, проте підкреслює актуальність подальших досліджень і необхідність розширеного аналізу сучасних малоресурсних геш-функцій, який має охоплювати як їхні структурні особливості та криптографічну стійкість, так і ефективність апаратної реалізації в умо-

вах обмежених ресурсів, що дозволить сформувати обґрунтованіші рекомендації для практичного застосування й сприятиме розвитку малоресурсної криптографії.

Матеріали та методи

У цьому дослідженні використано класифікаційний підхід до аналізу малоресурсних криптографічних геш-функцій, в основу якого покладено тип побудови внутрішньої конструкції геш-функції. Такий підхід дозволяє систематизовано дослідити алгоритми відповідно до їх архітектури, коректність порівняння характеристик і виявити конструкції, найпридатніші до реалізації в умовах обмежених ресурсів. Розглядаються дві основні конструкції геш-функції:

– ітераційна конструкція Меркла–Дамґарда;

– конструкція «губка».

Аналіз виконано на основі наукових публікацій із провідних міжнародних баз даних (IEEE Xplore, SpringerLink, ScienceDirect, ACM Digital Library, Google Scholar), а також офіційних звітів авторів алгоритмів та технічної документації [14]—[27], пов’язаної із сучасними криптографічними геш-функціями, орієнтованими на використання у пристроях з обмеженими ресурсами, зокрема в IoT-системах, вбудованих мікроконтролерах та сенсорних мережах.

Дослідження охоплює геш-функції, що реалізують зазначені конструктивні підходи, та спрямоване на оцінку їхньої придатності до використання в умовах обмежених обчислювальних можливостей. Особливий акцент зроблено на апаратних реалізаціях малоресурсних геш-функцій, оскільки саме вони визначають практичну доцільність застосування в IoT-пристроях. Для оцінювання враховано такі показники:

– *апаратна складність*, як універсальна міра розміру апаратної реалізації криптографічного примітиву, що визначається площею схеми у вентильних еквівалентах. Менші значення GE відповідають компактнішим і дешевшим реалізаціям; для реалізації на пристроях з дуже низькими ресурсами (наприклад, 4-бітових мікроконтролерах) вважається прийнятним забезпечення апаратної складності до 1000 GE, а для загальніших IoT-платформ допустимими є реалізації до 2000–3000 GE;

– *довжина геш-значення*, яка визначає рівень криптографічної стійкості та впливає на апаратні витрати;

– *криптографічна стійкість*, що визначається стосовно колізій та атак на перший і другий прообрази;

– *пропускна здатність*, яка визначає кількість даних, що можуть бути оброблені за одиницю часу (у бітах або байтах за секунду). Визначається за формулою

$$T = \frac{B \cdot F}{N}, \tag{1}$$

де B — розмір блока (у бітах), F — тактова частота (у герцах), N — кількість тактів на обробку блока;

– *енергоспоживання*, яке визначається за формулою

$$P = \frac{B \cdot E_{per\ bit}}{Lat}, \tag{2}$$

де B — розмір блока (у бітах), Lat — час за який мікросхема видає вихідні дані після отримання вхідного повідомлення, $E_{per\ bit}$ — енергія необхідна для зберігання біта даних.

На завершальному етапі результати аналізу систематизовано та узагальнено у вигляді порівняльних таблиць, де відображено основні характеристики геш-функцій відповідно до типу їх побудови з метою виявлення найефективніших рішень та визначення балансу між апаратною складністю та рівнем забезпеченої криптографічної стійкості.

Підходи до побудови малоресурсних геш-функцій

Конструкція Меркла–Дамґарда (Merkle–Damgård) є класичним та широко застосовуваним підходом. Теоретично вона базується на результаті Меркла та Дамґарда: якщо використати коректну схему доповнення (Padding) та стійку до колізій функцію ущільнення, то утворена на її основі геш-функція також зберігатиме стійкість до колізій [28]. Відповідно до цього підходу вхідне повідомлення M довільної довжини перед обробкою розбивається на блоки однакової довжини m_1, m_2,…, m_n, і доповнюються (Padding) згідно з визначеним правилом. Гешування здійснюється ітеративно,

де кожен блок обробляється функцією ущільнення f, що приймає на вхід попереднє геш-значення h_{i-1} та поточний блок m_i, і обчислює нове геш-значення h_i за формулою:

$$h_i = f\left(h_{i-1}, m_i\right),\ i = 1,..., n, \tag{3}$$

де h_0 — значення ініціалізації геш-функції, а h_n — остаточне геш-значення.

На жаль, конструкція Меркла–Дамґарда має низку властивостей, що впливають на її безпеку. Зокрема, у цій схемі можлива атака довжинного доповнення, яка дозволяє, отримавши одну колізію, відносно просто знайти додаткові. Для повідомлень значної довжини складність атаки на другий прообраз є суттєво нижчою за повний перебір, а побудова множинних колізій (мультиколізій) вимагає лише незначного додаткового обчислювального ресурсу. Існує також, так звана, атака доповнення, коли, маючи геш від невідомого повідомлення X, можна обчислити геш-значення вхідних повідомлень, пов'язаних з X, хоча саме повідомлення X залишається невідомим. Через ці структурні слабкості, особливо проблеми довжинного доповнення та мультиколізійних атак, в реалізації використовують низку модифікацій класичної побудови. Серед них, наприклад, ширококанальна адаптація (Wide-Pipe Construction) Стефана Лакса [29], яка дозволяє збільшити внутрішній розмір стану геш-функції, що підвищує стійкість до атак на другий прообраз та атак мультиколізії. Інші підходи включають використання додаткових схем доповнення та модифікацій функції ущільнення для зменшення ефективності атак доповнення, що підвищує криптографічну стійкість алгоритму і дозволяє його застосовувати у малоресурсних середовищах.

Поширеним підходом щодо реалізації функції ущільнення в конструкції Меркла–Дамґарда є використання блокових шифрів. Найпоширенішими є схеми Девіса–Меєра (Davies–Meyer), Матіаса–Меєра–Оseaca (Matyas–Meyer–Oseas) [30] та Міягучі–Пренеля (Miyaguchi–Preneel [31], які визначають, як комбінувати вихід шифру та блок вхідних даних для отримання нового стану геш-функції.

У схемі Девіса–Меєра (Davies–Meyer) функція ущільнення реалізується шляхом зашифрування попереднього геш-значення h_{i-1} заданим блоковим шифром E_i, що як ключ використовує блок повідомлення m_i, та подальшого виконання операції додавання за модулем два між отриманим результатом і значенням h_{i-1} для отримання наступного геш-значення h_i

$$h_i = E_{m_i}\left(h_{i-1}\right) \oplus h_{i-1}. \tag{4}$$

Недоліком схеми Девіса–Мейєра є те, що незважаючи на безпечний блоковий шифр, є можливість появи так званих «нерухомих точок», коли для деякого блоку повідомлення m існує значення $h = E_m^{-1}(0)$, що задовольняє умову $E_m(h) \oplus h = h$, тобто функція ущільнення повертає те саме геш-значення [32].

У схемі Матіаса–Меєра–Оseaca (Matyas–Meyer–Oseas) враховано недолік конструкції Девіса–Мейєра. Тут блок повідомлення m_i використовується як вхідні дані для функції шифрування, а попереднє геш-значення h_{i-1} після додаткового перетворення g застосовується як ключ для блокового шифру E. Наступне геш-значення обчислюється шляхом поєднання результату зашифрування з самим блоком повідомлення операцією додавання за модулем два:

$$h_i = E_{g(h_{i-1})}\left(m_i\right) \oplus m_i, \tag{5}$$

де $g(h_{i-1})$ — функція перетворення, що змінюється залежно від реалізації. Блоковий шифр E може використовувати ключ, довжина якого відрізняється від дожини блока, тобто не збігається з розміром геш-значення. Цю проблему усуває функція g, яка перетворює h_{i-1} у ключ відповідного розміру.

Схема Міягучі–Пренеля (Miyaguchi–Preneel) є подальшим розвитком конструкції Матіаса–Меєра–Оseaca. Відмінність від попередньої схеми полягає в тому, що до зашифрованого блоку повідомлення додається не лише блок повідомлення, а й попереднє геш-значення

$$h_i = E_{g(h_{i-1})}\left(m_i\right) \oplus h_{i-1} \oplus m_i. \tag{6}$$

Таке поєднання гарантує, що кожне наступне геш-значення залежить одночасно від всіх трьох компонентів — ключа, блоку повідомлення та попереднього геш-значення, що підвищує дифузію та стійкість функції ущільнення до криптографічних атак, зокрема атак на «нерухомі точки» та селекційні колізії.

Конструкція «губка» (sponge-конструкція) є однією з найрозповсюдженіших сучасних схем для побудови геш-функцій. Існує дві класичні sponge-конструкції: P-sponge, де в ролі базової функції використовується відома перестановка, та T-sponge, що ґрунтується на випадковій функції й пере-

важно застосовується як теоретична модель. «Губка» є ітеративною конструкцією, яка на відміну від класичної схеми Меркла–Дамґарда, дозволяє обробляти повідомлення довільної довжини та формувати вихід довільної довжини на основі перетворень f [33]. Перетворення f оперує внутрішнім станом фіксованого розміру біт $b = r + c$, для якого r називають бітовою швидкістю, а c — ємністю.

Робота «губки» відбувається у дві фази: «вбирання» (absorb) та «вичавлювання» (squeeze). Фаза «вбирання» призначена для поступового інтегрування вхідного повідомлення у внутрішній стан «губки», а фаза «вичавлювання» формує вихідну послідовність довільної довжини на основі цього стану. Вхідне повідомлення M за необхідності доповнюється (Padding), після чого розбивається на рівні блоки довжиною по r біт. Внутрішній стан «губки» ініціалізується нульовими бітами. Геш-значення обчислюється шляхом послідовного «вбирання» доповненого повідомлення та «вичавлювання» внутрішнього стану.

Фаза «вбирання» включає кроки:

1) перші r біт внутрішнього стану замінюються результатом операції їх додавання за модулем два з відповідним блоком доповненого повідомлення;

2) виконання перетворення f для внутрішнього стану.

Процес «вбирання» виконується для всіх блоків вхідного повідомлення M. У фазі «вичавлювання» для внутрішнього стану здійснюється перетворення f в результаті якого перші r біт подаються на вихід. Процес «вичавлювання» продовжується протягом L/r ітерацій, де L — довжина геш-значення. Геш-значення формується шляхом конкатенації отриманих r-бітових блоків. Останні c біт внутрішнього стану залежать від вхідних блоків лише опосередковано й не виводяться в ході фази «вичавлювання», що підвищує криптографічну стійкість конструкції. Зокрема, складність знаходження колізій у конструкції «губка» не менша ніж $2^{c/2}$, тоді як атаки на перший та другий прообраз вимагають не менше 2^c операцій. Крім того, ймовірність відрізнити випадкову «губку» від випадкового оракула обмежується величиною $N(N+1)/2^{c+1}$, де N — кількість виконаних перетворень f. Це означає, що у разі практично доцільного вибору параметрів, коли $c \geq 2L$ для вихідного геш-значення довжиною L, конструкція «губки» досягає криптографічної стійкості, еквівалентної ідеальній моделі випадкового оракула [34].

Аналіз сучасних малоресурсних методів гешування

Розглянемо три геш-функції, побудовані на основі конструкції Меркла–Дамґарда.

Алгоритм ARMADILLO, за твердженням авторів [14], запропоновано як універсальну криптографічну конструкцію загального призначення, придатну для використання у протоколах автентифікації, гешування даних, формування цифрових підписів, генерування псевдовипадкових чисел та інших криптографічних застосуваннях. Подальша модифікація ARMADILLO2 усунула недоліки першої версії, підвищивши криптостійкість і продуктивність алгоритму. Обидві версії підтримують кілька варіантів довжини геш-значення: 80, 128, 160, 192 та 256 біт.

Представниками геш-функцій, побудованих з використанням блокових шифрів є DM-PRESENT, H-PRESENT, C-PRESENT та Lesamnta-LW.

Геш-функція DM-PRESENT [15] реалізована на основі схеми Девіса–Маєра, де 64-бітний внутрішній стан оновлюється за допомогою одного циклу зашифрування PRESENT-80 або PRESENT-128 та подальшої операції додавання за модулем два з попереднім станом. Такий підхід забезпечує односторонність функції та 64-бітний рівень стійкості. Апаратна реалізація DM-PRESENT-80 включає 64-бітний регістр, модуль керування на основі автомата станів, XOR-елементи та покрокову реалізацію алгоритму PRESENT. Повний цикл обчислення вимагає 31 такт для виконання зашифрування. Авторами також наведено порівняння серійної та паралельної реалізацій на FPGA, що дозволяє оцінити компроміс між площею апаратних витрат та пропускною здатністю.

H-PRESENT-128 та C-PRESENT-192 також побудовані за схемою Девіса–Маєра з використанням блокового шифру PRESENT [15], [16]. У H-PRESENT-128 внутрішній стан має довжину 128 біт, поділений на два 64-бітні слова, які обробляються разом з 64-бітним блоком повідомлення та незалежними константами для запобігання симетріям. У C-PRESENT-192 внутрішній стан складається з трьох 64-бітних слів, що обробляються аналогічним чином, що дозволяє формувати геш-значення довжиною 128 та 192 біт.

Геш-функція Lesamnta-LW використовує конструкцію wide-pipe конструкцію Меркла–Дамґарда, у поєднанні з AES-подібним блоковим шифром, реалізованим через 4-гілкову мережу Фейстеля (GFN). Основними операціями є SubBytes та MixColumns, що забезпечують високу нелінійність

і дифузію. Така конструкція робить Lesamnta-LW стійкою до відомих атак на геш-функції, що підтверджено авторами у роботі [17]. Lesamnta-LW обчислює геш-значення довжиною 256 біт.

Переважна більшість сучасних геш-функцій в основі використовують конструкцію «губка».

У дослідженні [18] автори пропонують геш-функцію SPONGENT, що використовує конструкцію «губка» у поєднанні з перестановками подібними до блокового шифру PRESENT. Основними перетвореннями є 4-бітні S-блоки, побітові перестановки та побітова операція XOR. SPONGENT реалізована у п’яти варіантах з вихідним геш-значенням довжиною: 80, 128, 160, 224 та 256 біт. Перший варіант призначений для застосувань з низькими вимогами до безпеки, другий і третій забезпечують помірний рівень безпеки, а четвертий і п’ятий — високий рівень.

Геш-функція PHOTON [19] реалізована за розширеною схемою «губка» з AES-подібною внутрішньою перестановкою. Вона представлена п’ятьма варіантами (PHOTON-80/20/16, PHOTON-128/16/16, PHOTON-160/36/36, PHOTON-224/32/32 та PHOTON-256/32/32), які відрізняються розмірами внутрішніх станів і параметрами бітової швидкості. Внутрішня перестановка складається з N раундів і чотирьох перетворень: AddConstants, SubCells, ShiftRows і MixColumnsSerial, що забезпечує сильну нелінійність і дифузію. Найбільший варіант PHOTON потребує близько 4362 GE у паралельній реалізації, що забезпечує максимальну пропускну здатність, при цьому внутрішня перестановка дозволяє економити близько 8 % ресурсів порівняно з AES. Послідовна реалізація для визначення 256-бітного геш-значення потребує лише 2177 GE, проте пропускна здатність при цьому суттєво нижча.

На відміну від SPONGENT та PHOTON, де функція ущільнення реалізована через S-блокові перетворення, автори Quark [20], [21], базуючись на потоковому шифрі Grain та полегшеному блоковому шифрі KATAN, застосували інший підхід — використання регістрів зсуву. Функція ущільнення складається з двох регістрів зсуву з нелінійним зв’язком (NFSR) та регістра зсуву з лінійним зв’язком (LFSR), що поєднані логічними функціями. Розробники представили три варіанти алгоритму: U-Quark, що забезпечує 64-бітний рівень безпеки, відзначається найменшою апаратною складністю (близько 1379 GE); D-Quark, який досягає 80-бітної безпеки за більшої складності апаратної реалізації (близько 1700 GE); S-Quark, що надає рівень безпеки до 112 біт і вимагає для своєї реалізації 2296 GE.

Геш-функція GLUON, запропонована в [22], побудована на використанні регістрів зсуву з переносом (FCSR), які раніше застосовувалися у потокових шифрах F-FCSR-v3 та X-FCSR-v2. Автори пропонують три варіанти GLUON з довжиною геш-значення: 128, 160 та 224 біт. Ключовою особливістю реалізації є паралельне завантаження даних у FCSR, що забезпечує значне зростання пропускної здатності, проте вимагає більших апаратних витрат: 2071 GE для 128-бітної реалізації, 2799 GE – для 160-бітної та 4724 GE – для 224-бітної реалізації.

У [23] запропоновано геш-функцію SPN-Hash, в якій функція ущільнення реалізується мережею підстановок-перестановок (SPN), що складається з чергування нелінійних S-блоків та лінійних операцій перестановки, забезпечуючи високі дифузійні властивості. На відміну від класичної «губки», SPN-Hash використовує так званий режим JH, де ущільнення повідомлення відбувається через спеціально сконструйовану перестановку з урахуванням ключових констант, що дозволяє уникати симетрій і підвищує стійкість до колізій. Завдяки цьому SPN-Hash формально захищена від диференційного криптоаналізу та забезпечує необхідну стійкість до атак на перший та другий прообрази. Алгоритм підтримує довжини геш-значень: 128, 256 та 512 біт.

У дослідженні [24] описано сімейство легких геш-функцій HVH, які базуються на конструкції «губка» з використанням блокового шифру VH. Ці геш-функції підтримують п’ять варіантів довжини геш-значення від 88 до 256 біт, що дозволяє адаптувати їх до різних вимог щодо безпеки та обмежень ресурсів. Автори зосередились на підвищенні швидкості гешування для 8-бітних мікроконтролерів, що застосовуються у RFID та заявляють про досягнення пропускної здатності 1,47 Мбіт/с. Апаратна складність HVH з довжиною геш-значення 256 біт становить 2009 GE.

Геш-функція THF (Tiny Hash Function) [25] побудована на конструкції «губка», де функція ущільнення містить дві фази розбиття блоку повідомлення на секції, у кожній з яких перетворення відбуваються окремо протягом заданої кількості раундів. В якості перетворень використовуються операції зсуву вліво та додавання за модулем два. Надалі автори пропонують шляхом «вичавлювання» формувати геш-значення довжиною 64, 128 та 256 біт з використанням реалізації з фіксованим внутрішнім станом довжиною 64 біти. У дослідженні наведено оцінки продуктивності та апаратної складності геш-функції, проте оцінка криптографічної стійкості виконана лише на основі дослідження Гемінгової відстані, тоді як стійкість до колізій, атак на перший та другий прообраз

та інших атак на геш-функції не досліджувалася.

Hash-One [26] реалізує конструкцію «губка» з двома нелінійними регістрами зсуву зі зворотним зв'язком (NFSR). Функція ущільнення використовує булеві перетворення (Pf, Qf, Lf), реалізовані виключно за допомогою NAND-вентилів, що забезпечує апаратну складність 1006 GE для обчислення геш-значень довжиною 160 біт. Внутрішній стан функції має нестандартний розмір 161 біт, а параметр ємності дорівнює 160 біт, що дозволяє Hash-One забезпечувати стійкість до колізій на рівні 2^{80}, а стійкість до атак на перший та другий прообрази на рівні 2^{160} та 2^{80} відповідно. У свою чергу, DeeR-Hash [27] подібно до Hash-One використовує два NFSR та додатково вводить один лінійний регістр зсуву (LFSR) для покращення дифузії. Автори запропонували використовувати у функції ущільнення динамічну схему вибору змінних з масивів, що дозволяє оптимізувати логіку. Для вибору позицій використано систему мультиплексорів та конфігураційних масивів, що формує оновлення стану з мінімальним набором логічних елементів. DeeR-Hash дозволяє обчислювати геш-значення довжиною 80 та 160 біт. Для довжини геш-значення 160 біт використання ємності 158 біт забезпечує криптографічну стійкість співставну з Hash-One, з апаратною складіостю 984 GE.

Для порівняння рівня криптографічної стійкості розглянутих малоресурсних геш-функцій у табл. 1 подано їх основні криптографічні характеристики, зокрема довжину геш-значення, оцінки стійкості до колізій, атак на перший і другий прообрази, а також інші характеристики: тип внутрішньої структури та мінімальна апаратна складність. Ці дані дозволяють простежити баланс між складністю апаратної реалізації та забезпеченням мінімальних вимог до безпеки, які відповідно до стандартів повинні складати не менше ніж $2^{L/2}$ для стійкості до колізій і 2^{L} для стійкості до атак на перший прообраз, де L — довжина геш-значення. Довжина геш-значення суттєво впливає як на криптографічну стійкість, так і на апаратну складність. Алгоритми з довжиною геш-значення 128 біт в більшості випадків демонструють апаратні витрати на рівні від 1000 до 2000 GE, тоді як для довших геш-значень (160…256 біт) складність часто зростає до 2000…4000 GE. Варто зазначити, що для завдань, які вимагають надзвичайно високих вимог до безпеки та криптографічної стійкості, використання геш-функцій з довжиною вихідного значення менше 256 біт може бути недостатнім. Міжнародні стандарти, зокрема NIST SP 800-232, рекомендують для довгострокового захисту використовувати геші не менше 256 біт, оскільки це забезпечує стійкість до колізій на рівні 2^{128}, що вважається достатнім для протистояння сучасним обчислювальним можливостям [35].

Таблиця 1

Криптографічні характеристик геш-функцій

Геш-функція	Довжина геш-значення (біт)	Структура	Мінімальна апаратна складність (GE)	Стійкість до		
				колізій	атак на перший прообраз	атак на другий прообраз
Геш-функції на основі конструкції Меркла–Дамгарда						
ARMADILLO [14]	80	Бітові перетворення на основі даних	2923	2^{40}	2^{80}	2^{80}
	128		4353	2^{64}	2^{128}	2^{128}
	160		5406	2^{80}	2^{160}	2^{160}
	192		6554	2^{96}	2^{192}	2^{192}
	256		8653	2^{128}	2^{256}	2^{256}
DM-PRESENT [15]	64	Девіса–Мейєра	1600	2^{40}	2^{64}	—
H-PRESENT [15]	128		2330	2^{64}	2^{128}	—
C-PRESENT [15]	192		4600	2^{96}	2^{192}	—
Lesamnta-LW [17]	256	Блоковий шифр	8240	2^{120}	2^{256}	2^{256}
Геш-функції на основі конструкції «губка»						
U-Quark [21]	136	P-губка	1379	2^{64}	2^{128}	2^{64}
D-Quark [21]	176	P-губка	1702	2^{80}	2^{160}	2^{80}
S-Quark [21]	256	P-губка	2296	2^{112}	2^{224}	2^{112}

Продовження таблиці 1

Геш-функція	Довжина геш-значення (біт)	Структура	Мінімальна апаратна складність (GE)	Стійкість до		
				колізій	атак на перший прообраз	атак на другий прообраз
Геш-функції на основі конструкції «губка»						
SPONGENT [18]	88	P-губка	738	2^{40}	2^{80}	2^{40}
	128		1060	2^{64}	2^{120}	2^{64}
	160		1329	2^{80}	2^{144}	2^{80}
	224		1728	2^{112}	2^{208}	2^{112}
	256		1950	2^{128}	2^{240}	2^{128}
PHOTON [19]	80	P-губка	865	2^{40}	2^{64}	2^{40}
	128		1122	2^{64}	2^{112}	2^{60}
	160		1396	2^{80}	2^{124}	2^{80}
	224		1736	2^{112}	2^{192}	2^{112}
	256		2177	2^{128}	2^{224}	2^{128}
GLUON [22]	128	T-губка	2071	2^{64}	2^{128}	2^{64}
	160		2800	2^{80}	2^{160}	2^{80}
	224		4724	2^{112}	2^{224}	2^{112}
SPN-Hash [23]	128	JH-губка	2777	2^{64}	2^{128}	—
	256		4625	2^{64}	2^{128}	—
HVH [11]	88	P-губка	857	2^{40}	2^{72}	2^{40}
	128		1145	2^{64}	2^{120}	2^{64}
	160		1385	2^{80}	2^{144}	2^{80}
	224		1769	2^{112}	2^{208}	2^{112}
	256		2009	2^{128}	2^{224}	2^{128}
THF [25]	128/256	P-губка	1296	?	?	—
Hash-One [26]	160	P-губка	1006	2^{80}	2^{160}	2^{80}
DeeR-Hash [27]	160	P-губка	984	2^{79}	2^{158}	2^{79}

Для більшості застосунків Інтернету речей критично важливим є забезпечення балансу між безпекою та апаратними витратами. Оптимальним є забезпечення високого рівня криптографічної стійкості за найменшою кількістю вентильних еквівалентів (GE). У цьому контексті помітною є перевага конструкції «губка», над ітеративними структурами. Наприклад, у категорії 128-бітних геш-функцій SPONGENT-128 демонструє стійкість до колізій на рівні 2^{64} з апаратною складністю всього 1060 GE, що значно ефективніше, порівняно з GLUON-128 (2071 GE) або H-PRESENT (2330 GE), які забезпечують такий самий рівень безпеки, але вимагають для своєї реалізації майже вдвічі більше апаратних витрат. Алгоритми DeeR-Hash та Hash-One у категорії обчислення геш-значень довжиною 160…192 біт демонструють найкращий компроміс між безпекою та апаратною складністю. Вони забезпечують стійкість до колізій на рівні 2^{79} та 2^{80} відповідно з апаратною складністю 984 GE і 1006 GE, суттєво перевершуючи такі функції, як SPONGENT-160 (1329 GE), HVH-160 (1385 GE), PHOTON (1396 GE), D-Quark (1702 GE) та інші. Для забезпечення високого рівня безпеки варто застосовувати геш-значення довжиною 256 біт. Серед розглянутих геш-функцій SPONGENT-256 забезпечує найвищий рівень безпеки з криптографічною стійкістю до колізій 2^{128} за мінімальних витрат 1950 GE. Інші 256-бітні геш-функції, як ARMADILLO-256 (8653 GE) або Lesamnta-LW (8240 GE), хоч і забезпечують подібний рівень стійкості, є надзвичайно неефективними і непридатними для більшості сучасних малоресурсних пристроїв через надмірні апаратні витрати. Варто зауважити, що геш-функція HVH-256 для обчислення геш-значень довжиною 256 біт також забезпечує високий рівень стійкості до колізій (2^{128}), атак на перший та другий прообрази (2^{224} та 2^{128} відповідно) з порівняно невеликими апаратними витратами (2009 GE), а алгоритм THF-256 надає можливість обчислення таких геш-значень з найменшими витратами (всього 1296 GE). У той же час рівень безпеки THF викликає сумніви, оскільки відсутні дані щодо оцінки криптографічної стійкості цієї геш-функції.

Для пристроїв, де критично важливим є мінімізація апаратних витрат, навіть за рахунок зниження рівня безпеки геш-функція SPONGENT-88 надає найменшу апаратну складність, яка стано-

вить лише 738 GE, та найбільшу серед розглянутих у цій категорії стійкість до колізій. Через цей показник вона є ідеальною для таких пристроїв, як пасивні RFID-мітки. Інші функції, такі як HVH-88 (857 GE) та PHOTON-80 (865 GE), також демонструють низькі апаратні витрати та є гарними альтернативами. Порівняно з ними, DM-PRESENT-64 має значно вищу складність (1600 GE).

У табл. 2 подано оцінки апаратної складності (GE), пропускної здатності (кбіт/с) та енергоспоживання засобів, що реалізують геш-функції. Апаратна складність, пропускна здатність та енергоспоживання подані двома оцінками, перша з яких — для паралельної реалізації, а друга — для послідовної. Для більшості геш-функцій, таких як ARMADILLO, SPONGENT та PHOTON, розробники пропонують послідовну реалізацію з мінімальними апаратними витратами та паралельну реалізацію з підвищеною продуктивністю за рахунок паралельної обробки і збільшених апаратних ресурсів.

Таблиця 2

Технічні характеристики засобів гешування

Геш-функція	Довжина геш-значення, біт	Технологія, мкм	Апаратна складність, GE	Пропускна здатність, кбіт/с, якщо 100 кГц	Енергоспоживання, мкВт
Геш-функції на основі конструкції Меркла–Дамґарда					
ARMADILLO[14]	80	0,18	4030/2923	1090/272	77/44
	128		6025/4353	1000/250	118/65
	160		7492/5406	1000/250	158/83
	192		8999/6554	1000/250	183/102
	256		11914/8653	1000/250	251/137
DM-PRESENT [15]	64	0,18	2530/1600	387,88/14,63	7,49/1,83
H-PRESENT [15]	128		4256/2330	200/11,45	—
C-PRESENT [15]	192		8048/4600	59,26/1,9	—
Геш-функції на основі конструкції «губка»					
U-Quark [21]	136	0,18	2392/1379	11,76/1,47	4,07/2,44
D-Quark [21]	176		2819/1702	18,18/2,27	4,76/3,10
S-Quark [21]	256		4640/2296	50,0/3,13	8,39/4,35
SPONGENT [18]	88	0,13	1127/738	111,3/35,8	2,31/1,57
	128		1687/1060	11,43/0,34	3,58/2,20
	160		2190/1329	17,78/0,40	4,47/2,85
	224		2903/1728	13,33/0,22	5,97/3,73
	256		3281/1950	11,43/0,17	6,62/4,21
PHOTON [19]	80	0,18	1168/865	15,15/2,82	—
	128		1708/1122	10,26/1,61	—
	160		2117/1396	20/2,70	—
	224		2786/1736	15,69/1,86	—
	256		4362/2177	20,51/3,21	—
GLUON [22]	128	0,18	2071	12,12	—
	160		2800	32	—
	224		4724	58,18	—
SPN-Hash [11]	128	0,18	4600/2777	55,7/36,1	—
	256		8500/4625	111,3/35,8	—
HVH [11]	88	0,18	1129/857	44,44/2,02	—
	128		1537/1145	44,44/1,31	—
	160		1876/1385	88,89/2,02	—
	224		2420/1769	88,89/1,48	—
	256		2680/2009	177,78/2,54	—
THF [25]	128	0,13	1296	1255/250	4,52/3,14
	256		1296	1255/250	5,22/3,87
Hash-One [26]	160	0,18	2130/1006	46/2	—

SPONGENT-128 у паралельній реалізації досягає 11,43 кбіт/с за 1060 GE, тоді як ARMADILLO-128 забезпечує 1000 кбіт/с, але потребує 4353 GE. SPONGENT-88 у послідовній реалізації з 738 GE

забезпечує 35,8 кбіт/с, через що він є придатним для пристроїв з високими вимогами до апаратних витрат, таких як пасивні RFID-мітки. DM-PRESENT-64 демонструє більшу пропускну здатність (387,88 кбіт/с) за 1600 GE, що обмежує його застосування в подібних системах. Для високопродуктивних систем 256-бітні функції, як HVH-256 та S-Quark-256, забезпечують значну пропускну здатність. HVH-256 у паралельній реалізації досягає 177,78 кбіт/с за 2680 GE, а S-Quark-256 — 50,0 кбіт/с за 4640 GE. Варто зазначити, що SPN-Hash також демонструє високу пропускну здатність у обох реалізаціях, проте через велику апаратну складність недоцільне її використання в малоресурсних системах. Геш-функція THF-256 демонструє пропускну здатність 1255 кбіт/с за найменших апаратних витрат 1296 GE, проте відсутність даних про криптографічну стійкість обмежує її практичне використання.

Дослідження геш-функцій показує, що енергоспоживання прямо залежить від апаратних витрат: чим більші витрати GE, тим більше енергії споживає мікросхема. Наприклад, енергоспоживання для SPONGENT у послідовній та паралельній реалізаціях для геш-значень довжиною 256 біт становить 4,21 та 6,62 мкВт відповідно і є одним з найкращих серед розглянутих. Найбільші показники енергоспоживання у геш-функції ARMADILLO, що споживає від 44 до 251 мкВт залежно від реалізації та DM-PRESENT, яка для обчислення геш-значення довжиною 64 біт вимагає від 1,83 мкВт (для послідовної) до 7,43 мкВт (у паралельній реалізації). Для деяких геш-функцій, таких як PHOTON, Lesamnta-LW, SPN-Hash, HVH, Hash-One та DeeR-Hash, автори не наводять оцінку енергоспоживання, що обмежує можливість надати обґрунтовану рекомендацію щодо практичного використання.

Для виконання узагальненого порівняння та визначення ефективності засобів гешування введено інтегральний коефіцієнт ефективності, який дозволяє оцінити одночасно декілька параметрів: криптографічну стійкість, пропускну здатність, апаратну складність та енергоспоживання. Необхідність застосування інтегрального показника зумовлена різнорідною природою зазначених характеристик геш-функцій, які вимірюються в різних одиницях і, як наслідок, не дають змоги сформувати однозначний висновок щодо їх відносної ефективності. Наприклад, геш-функція може демонструвати високу пропускну здатність, але вимагати надмірних апаратних ресурсів, тоді як інші засоби гешування мають низьку апаратну складність, але недостатню криптографічну стійкість. Таким чином, інтеграція окремих параметрів у єдиний вимірювальний показник дозволяє отримати комплексну оцінку, що відображає баланс між основними вимогами до геш-функцій у системах з обмеженими апаратними можливостями. Інтегральний коефіцієнт ефективності $K_{\text{еф}}$ визначається за формулою

$$K_{\text{еф}} = \frac{S^a \cdot T^b}{GE^c \cdot P^d}, \tag{7}$$

де S — рівень криптографічної стійкості в бітах (визначається як мінімальне значення з показників стійкості до колізій, атак на перший та другий прообрази), T — пропускна здатність (кбіт/с), GE — апаратна складність, P — енергоспоживання (мкВт), a, b, c, d — коефіцієнти впливу кожної з характеристик на загальну оцінку ефективності, при цьому $a + b + c + d = 1$.

У роботі застосовано значення $a = 0{,}45$; $b = 0{,}1$; $c = 0{,}4$; $d = 0{,}05$. Такий вибір коефіцієнтів впливу зумовлений тим, що криптографічна стійкість ($a = 0{,}45$) є визначальним, безумовним критерієм для будь-якого криптографічного алгоритму. Засіб гешування з недостатнім рівнем криптографічної стійкості не може бути рекомендований до застосування, незалежно від його інших властивостей. Апаратна складність ($c = 0{,}4$) має аналогічний критичний вплив, оскільки вона безпосередньо визначає можливість фізичної реалізації геш-функції в пристроях з обмеженими апаратними можливостями. Пропускна здатність ($b = 0{,}1$) та енергоспоживання ($d = 0{,}05$) мають помірний вплив, оскільки навіть відносно низькі швидкості гешування є достатніми для більшості IoT-сценаріїв, а енергоспоживання є важливим лише для автономних пристроїв. У випадках, коли для певного засобу гешування відсутні оцінки енергоспоживання (наприклад, для HVH, Hash-One, GLUON), використовується підхід перерозподілу впливу параметра в результаті якого $P^d = 1$, а вага параметра d рівномірно розподіляється на інші параметри S, T та GE, що продовжують впливати на значення $K_{\text{еф}}$. Такий підхід дозволяє зберегти коректність порівняння та уникнути штучного зниження ефективності алгоритмів через неповні характеристики.

Результати визначення інтегрального коефіцієнта ефективності подано у вигляді діаграм на рис. 1, 2 для паралельних і послідовних реалізацій геш-функцій відповідно. Аналіз отриманих зна-

чень демонструє, що засоби гешування, побудовані на конструкції «губка», забезпечують найзбалансованіше співвідношення між криптографічною стійкістю, апаратною складністю та пропускною здатністю.

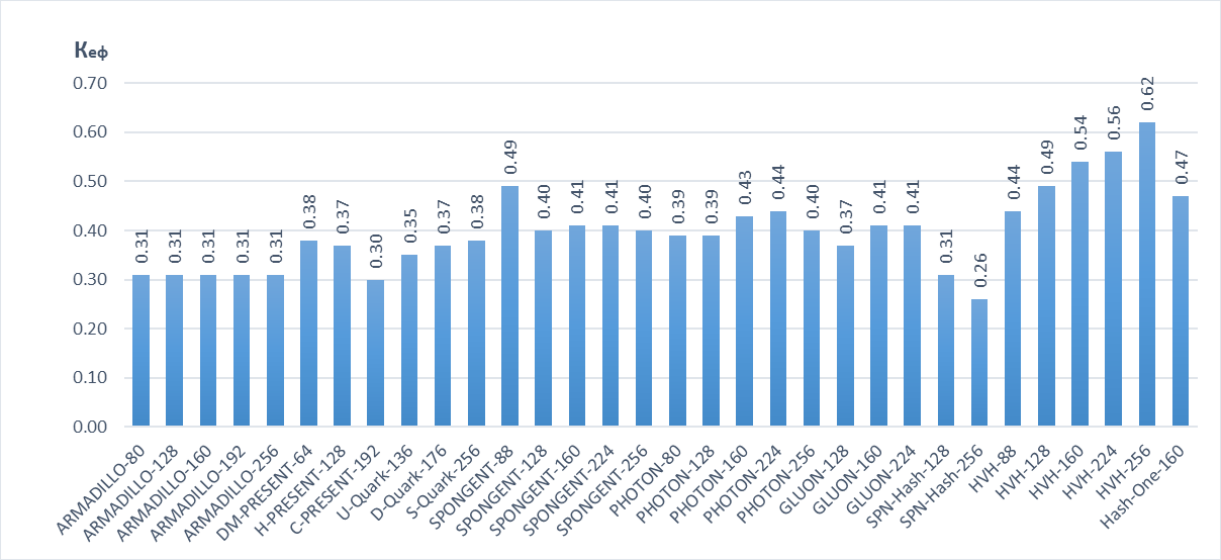

Рис. 1. Діаграма порівняння коефіцієнта ефектисності геш-функцій за паралельної реалізації

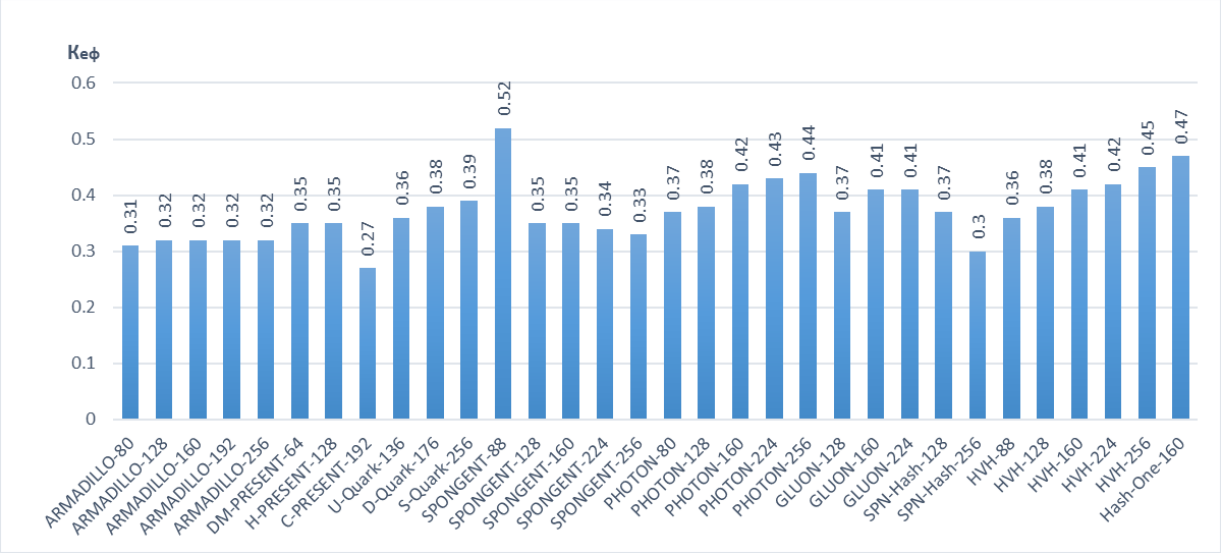

Рис. 2. Діаграма порівняння коефіцієнта ефектисності геш-функцій за послідовної реалізації

У контексті паралельної апаратної реалізації найвищі значення $K_{еф}$ демонструє геш-функція HVH в усьому дослідженому діапазоні довжин геш-значення (від 88 до 256 біт). Високі значення ефективності демонструє SPONGENT-88 за рахунок високої пропускної здатності та найменших апаратних витрат. Hash-One також продемонструвала одні з найвищих значень $K_{еф}$, як за паралельної, так і за послідовної реалізаціях, що свідчить про доцільність її використання в різних класах малоресурсних систем. Геш-функції на основі конструкцій Меркла–Дамґарда ARMADILLO та PRESENT, незважаючи на їхню високу пропускну здатність, вимагають значних апаратних ресурсів та більшого енергоспоживання, що підтверджують посередні значення коефіцієнта ефективності в порівнянні з іншими розглянутими рішеннями.

Аналіз результатів для послідовних апаратних реалізацій демонструє, що найвищі значення коефіцієнта ефективності $K_{еф}$ серед малоресурсних геш-функцій у SPONGENT-88 та Hash-One-160. Проте, у випадках, коли вимоги до безпеки є максимальними і потрібна більша довжина геш-значення (до 256 біт), алгоритми HVH-256 та PHOTON-256 також мають високі значення $K_{еф}$, через що вони є найкращими кандидатами для впровадження у системи, де пріоритетом є криптографічна стійкість, зі збереженням прийнятного рівня апаратних витрат. Найгірші значення інтегрального

коефіцієнта ефективності $K_{еф}$ серед розглянутих геш-функцій продемонстрували C-PRESENT-192 та SPN-hash, що є прямим наслідком їхньої високої апаратної складності. Для деяких геш-функцій фіксується кореляція між довжиною геш-значення та значенням інтегрального коефіцієнта ефективності $K_{еф}$.

Для низки алгоритмів (зокрема, PHOTON, Quark та HVH) спостерігається зростання $K_{еф}$ зі збільшенням довжини геш-значення, причому для HVH це підвищення є найсуттєвішим. Водночас, геш-функції SPONGENT, SPN-hash та PRESENT демонструють протилежну тенденцію — зниження $K_{еф}$. Подібна відмінність свідчить про те, що геш-функції з $K_{еф}$, що зростає, є масштабованішими, оскільки додаткові апаратні витрати, необхідні для забезпечення більшої довжини гешу, компенсуються пропорційно вищим сукупним приростом криптографічної стійкості та/або пропускної здатності. Натомість, у геш-функціях, для яких значення $K_{еф}$ зменшується, апаратна складність та витрати енергоспоживання зростають непропорційно швидше, ніж досягається вищий рівень криптографічної стійкості та/або пропускної здатності.

Висновки

У ході проведеного дослідження проаналізовано основні характеристики малоресурсних геш-функцій, серед яких апаратна складність реалізації, довжина геш-значення, пропускна здатність, енергоспоживання та криптографічна стійкість до колізій. Запропоновано узагальнену оцінку ефективності засобів гешування шляхом введення інтегрального коефіцієнта ефективності, який поєднує ключові технічні та криптографічні властивості геш-функцій і дозволяє отримати комплексну характеристику придатності алгоритмів для малоресурсних систем.

Порівняльний аналіз показав, що геш-функції, побудовані на основі ітеративної конструкцій Меркла–Дамґарда, в цілому демонструють прийнятний рівень криптографічної стійкості та можуть застосовуватися в окремих випадках, проте характеризуються посередніми та низькими значеннями інтегрального коефіцієнта ефективності. Їхня реалізація часто пов’язана зі значними апаратними витратами, ще обмежує можливості широкого їх використання у пристроях з жорсткими ресурсними обмеженнями та підкреслює необхідність пошуку оптимізованих рішень, які б забезпечували збалансоване поєднання безпеки, енергоефективності та простоти апаратної реалізації. Геш-функції, побудовані на основі конструкції «губка», продемонстрували найкращий компроміс між безпекою, продуктивністю та апаратними витратами. Зокрема, SPONGENT вирізняється мінімальними вимогами до апаратних ресурсів, через що він є придатним для використання у сенсорних мережах та RFID-системах, тоді як HVH для обчислення геш-значень довжиною 128 та 256 біт забезпечує високу пропускну здатність та криптографічну стійкість за порівняно невеликих апаратних витратах, демонструють одні з найвищих значень інтегрального коефіцієнта ефективності серед розглянутих геш-функцій.

Особливої уваги заслуговують геш-функції DeeR-Hash та Hash-One, які показали достатній рівень криптографічної стійкості та найменші апаратні витрати для обчислення геш-значення довжиною 160 біт. Їхня ефективність пояснюється використанням комбінованих конструкцій з декількома нелінійними та лінійними регістрами зсуву у поєднанні з найпростішими операціями, що дозволяє суттєво зменшити апаратні витрати зі збереженням достатнього рівня безпеки. Hash-One демонструє одне з найвищих значень інтегрального коефіцієнта ефективності, що свідчить про перспективність подальшого розвитку подібних схем і їхню придатність для використання у пристроях, де критично важливою є компактність реалізації. Серед інших методів вирізняється геш-функція THF, яка забезпечує високу пропускну здатність для обчислення 256-бітних геш-значень з найменшими витратами. Водночас її криптографічні властивості поки не досліджені належним чином, що не дозволяє зробити остаточні висновки щодо можливості практичного застосування.

У цілому, отримані результати підтверджують перспективність подальшого розвитку конструкцій типу «губка» та комбінованих схем на основі регістрів зсуву. Подальші дослідження авторів будуть спрямовані на розробку нової геш-функції, яка поєднуватиме ітеративну конструкцію, де в якості функцій ущільнення будуть використані прості комбіновані схеми з лінійними регістрами зсуву (LFSR) та простими логічними операціями з метою мінімізації апаратних витрат зі збереженням необхідного рівня криптографічної стійкості.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] S. Sinha, “State of IoT 2024: Number of connected IoT devices growing 13% to 18.8 billion globally,” *IoT Analytics*, 3 вер. 2024. [Electronic resource]. Available: https://iot-analytics.com/number-connected-iot-devices/ . Accessed: 21.07.2025.

[2] A. Mileva, V. Dimitrova, and O. Kara, “Catalog and Illustrative Examples of Lightweight Cryptographic Primitives,” in *Security of Ubiquitous Computing Systems*, G. Avoine, J. Hernandez-Castro, Ed. Cham, Switzerland: Springer, 2021, с. 21-47. https://doi.org/10.1007/978-3-030-10591-4 .

[3] R. Kalaria, A. S. M. Kayes, W. Rahayu, and E. Pardede, “A secure mutual authentication approach to fog computing environment,” *Comput. & Secur.*, vol. 111, pp. 102-483, 2021. https://doi.org/10.1016/j.cose.2021.102483 .

[4] K.-L. Tsai, F.-Y. Leu, L.-L. Hung, and C.-Y. Ko, “Secure session key generation method for LoRaWAN servers,” *IEEE Access*, vol. 8, pp. 54631-54640, 2020. https://doi.org/10.1109/ACCESS.2020.2978100 .

[5] D. I. Afryansyah, M. Magfirawaty, and K. Ramli, “The Development and Analysis of TWISH: A Lightweight-Block-Cipher-TWINE-Based Hash Function,” in *2018 Thirteenth Int. Conf. Digit. Inf. Manage. (ICDIM)*, Berlin, Germany, Sep. 24-26, 2018. IEEE, 2018. https://doi.org/10.1109/ICDIM.2018.8847056 .

[6] D. N. Gupta, and R. Kumar, “Sponge based lightweight cryptographic hash functions for IoT applications,” in *2021 Int. Conf. Intell. Technol. (CONIT)*, Hubli, India, Jun. 25-27, 2021. IEEE, 2021. https://doi.org/10.1109/CONIT51480.2021.9498572.

[7] V. A. Thakor, M. A. Razzaque, and M. R. Khandaker, “Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities,” *IEEE Access*, vol. 9, pp. 28177-28193, 2021. https://doi.org/10.1109/ACCESS.2021.3052867 .

[8] Z. A. Al-Odat, E. M. Al-Qtiemat, and S. U. Khan, “An efficient lightweight cryptography hash function for big data and IoT applications,” in *2020 IEEE Cloud Summit*, Harrisburg, PA, Oct. 21-22, 2020. IEEE, 2020. https://doi.org/10.1109/IEEECloudSummit48914.2020.00016 .

[9] M. Rana, Q. Mamun, and R. Islam, “Lightweight cryptography in IoT networks: A survey,” *Future Gener. Comput. Syst.*, vol. 129, pp. 77-89, Apr. 2022. https://doi.org/10.1016/j.future.2021.11.011 .

[10] W. H. G. Wali, and N. C. Iyer, “Power, Performance and Area Analysis of Sponge Based Lightweight HASH Function,” *Int. J. Elect. Electron. Eng. & Telecommun.*, pp. 245-256, 2023. https://doi.org/10.18178/ijeetc.12.4.245-256 .

[11] S. Windarta, Suryadi, K. Ramli, B. Pranggono, and T. S. Gunawan, “Lightweight cryptographic hash functions: Design trends, comparative study, and future directions,” *IEEE Access*, vol. 10, pp. 82272-82294, 2022. https://doi.org/10.1109/ACCESS.2022.3195572 .

[12] В. І. Селезньов, «Аналіз методів малоресурсного гешування,» на *LII науково-технічна конференція підрозділів ВНТУ*. Вінниця, 2023. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2023/paper/download/18664/15557 . Дата звернення: 25.07.2025.

[13] I. El Hanouti, H. El Fadili, S. Hraoui, and A. Jarjar, “A Lightweight Hash Function for Cryptographic and Pseudo-Cryptographic Applications,” in *Lecture Notes in Electrical Engineering*. Singapore: Springer Singap., 2021, pp. 495-505. https://doi.org/10.1007/978-981-33-6893-4_46 .

[14] S. Badel, et all., “Armadillo: A multi-purpose cryptographic primitive dedicated to hardware,” in *Cryptographic Hardware and Embedded Systems – CHES 2010*, S. Mangard, F.-X. Standaert, Berlin, Germany: Springer, 2010, pp. 398-412. https://doi.org/10.1007/978-3-642-15031-9_27 .

[15] A. Bogdanov, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, and Y. Seurin, “Hash functions and RFID tags: Mind the gap,” in *Cryptographic Hardware and Embedded Systems – CHES 2008*, Berlin, Germany: Springer, 2008, pp. 283-299. https://doi.org/10.1007/978-3-540-85053-3_18 .

[16] A. Y. Poschmann, “Lightweight cryptography cryptographic engineering for a pervasive world,” 2009. [Electronic resource]. Available: https://eprint.iacr.org/2009/516.pdf .

[17] S. Hirose, K. Ideguchi, H. Kuwakado, T. Owada, B. Preneel, and H. Yoshida, “An AES Based 256-bit Hash Function for Lightweight Applications: Lesamnta-LW,” *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, E95-A, no. 1, pp. 89-99, 2012. https://doi.org/10.1587/transfun.E95.A.89 .

[18] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and I. Verbauwhede, “SPONGENT: The Design Space of Lightweight Cryptographic Hashing,” *IEEE Trans. Comput.*, vol. 62, no. 10, pp. 2041-2053, Oct. 2013. https://doi.org/10.1109/TC.2012.196 .

[19] J. Guo, T. Peyrin, and A. Poschmann, “The photon family of lightweight hash functions,” in *Advances in Cryptology – CRYPTO 2011*, P. Rogaway, Ed. Berlin, Germany: Springer, 2011, pp. 222-239. https://doi.org/10.1007/978-3-642-22792-9_13 .

[20] J.-P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia, “QUARK: A lightweight hash,” *in Cryptographic Hardware and Embedded Systems – CHES 2010*, S. Mangard, F.-X. Standaert, Ed. Cham, Switzerland: Springer, с. 1-15, 2010. https://doi.org/10.1007/978-3-642-15031-9_1 .

[21] J.-P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia, “Quark: A lightweight hash,” *J. Cryptol.*, vol. 26, № 2, pp. 313-339, 2012. https://doi.org/10.1007/s00145-012-9125-6 .

[22] T. P. Berger, J. D’Hayer, K. Marquet, M. Minier, and G. Thomas, “The GLUON family: A lightweight hash function family based on FCSRs,” in *Progress in Cryptology – AFRICACRYPT 2012*, Springer, pp. 306-323, 2012. https://doi.org/10.1007/978-3-642-31410-0_19 .

[23] J. Choy, et all., “SPN-Hash: Improving the provable resistance against differential collision attacks,” in *Progress in Cryptology – AFRICACRYPT 2012*, Springer, pp. 270-286, 2012. https://doi.org/10.1007/978-3-642-31410-0_17 .

[24] Y. Huang, S. Li, W. Sun, X. Dai, and W. Zhu, “HVH: A lightweight hash function based on dual pseudo-random transformation,” in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, Cham, Switzerland: Springer, pp. 492-505, 2021. https://doi.org/10.1007/978-3-030-68884-4_41 .

[25] M. K. S. Manyam, and K. N. Rao, “Performance analysis of THF: A novel lightweight cryptography hash function,” *J. Theor. Appl. Inf. Technol.*, vol. 103, № 9, 2025. [Electronic resource]. Available: https://www.jatit.org/volumes/Vol103No9/16Vol103No9.pdf .

[26] P. M. Mukundan, S. Manayankath, C. Srinivasan, and M. Sethumadhavan, “Hash-One: A lightweight cryptographic hash function,” *IET Inf. Secur.*, vol. 10, № 5, pp. 225-231, 2016. https://doi.org/10.1049/iet-ifs.2015.0385 .

[27] D. N. Gupta, and R. Kumar, “DeeR-Hash: A lightweight hash construction for Industry 4.0/IoT,” *J. Sci. Ind. Res.*, vol. 82, № 1, pp. 142-150, 2023. https://doi.org/10.56042/jsir.v82i1.69938 .

[28] I. B. Damgård, “A Design Principle for Hash Functions,” in *Advances in Cryptology — CRYPTO’ 89 Proceedings*. New York, NY: Springer N. Y., n.d., pp. 416-427. https://doi.org/10.1007/0-387-34805-0_39 .

[29] Tri Ade Nia, Rudyanto Ompungsunggu, Arrant Ardi Sianipar, J. Jesimanta, and Yoramo Waruwu, “Merkle-Damgård as the Foundation of Hash Cryptography: A Study of Advantages and Limitations,” *J. Tek. Indones.*, vol. 3, № 1, pp. 24-29, May 2024. https://doi.org/10.58471/ju-ti.v3i01.652 .

[30] S. M. Matyas, C. H. Meyer, and J. Oseas, “Generating Strong One-Way Functions with Cryptographic Algorithm,” *IBM Technical Disclosure Bulletin*, v. 27, no. 10A, pp. 5658-5659, Mar 1985.

[31] B. Preneel, R. Govaerts, and J. Vandewalle, “Hash Functions Based on Block Ciphers: A Synthetic Approach,” in *Advances in Cryptology — CRYPTO’ 93*. Berlin, Heidelberg: Springer Berl. Heidelb., 1993, pp. 368-378., https://doi.org/10.1007/3-540-48329-2_31.

[32] O. J. Permana, B. H. Susanti, and M. Christine, “Fixed-point attack on Davies–Meyer hash function scheme based on SIMON, SPECK, and SIMECK algorithms,” in *Saf. Problems Civil Eng. Crit. Infrastructures (SPCECI2021)*, VII Int. Conference. 2023. https://doi.org/10.1063/5.0119689 .

[33] G. Bertoni, J. Daemen, M. Peeters, and G. van Assche, “Sponge functions,” in *ECRYPT Workshop on Hash Functions, 2007*. [Electronic resource]. Available: https://www.researchgate.net/publication/242285874_Sponge_Functions .

[34] J.-S. Coron, J. Patarin, and Y. Seurin, “The Random Oracle Model and the Ideal Cipher Model Are Equivalent,” in *Lecture Notes in Computer Science*. Berlin, Germany: Springer, pp. 1-20, 2008. https://doi.org/10.1007/978-3-540-85174-5_1 .

[35] NIST, “Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process,” 2018. [Electronic resource]. Available: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf .

Рекомендована кафедрою захисту інформації ВНТУ

Стаття надійшла до редакції 5.09.2025

Лужецький Володимир Андрійович — д-р техн. наук, професор, завідувач кафедри захисту інформації, e-mail: lva.kzi2002@gmail.com ;

Селезньов Віталій Ігорович — аспірант кафедри захисту інформації, e-mail: seleznov.vitalii@gmail.com.

Вінницький національний технічний університет, Вінниця

V. A. Luzhetskyi[1]
V. I. Seleznov[1]

Review of the Approaches and Methods for Lightweight Data Hashing

[1]Vinnytsia National Technical University

This paper presents a review of modern approaches to the design of hash functions, which are a fundamental element of cryptographic protection in the systems with limited hardware capabilities, particularly in Internet of Things (IoT) devices, embedded microcontrollers, and sensor networks. The relevance of the study is driven by the intensive development of these technologies and the necessity of ensuring a high level of security with minimal resource consumption. The main objective of the work is to analyze the structural features, cryptographic properties, and technical characteristics of tools that implement lightweight hash functions. Two fundamentally different approaches to hash function design are considered: the iterative Merkle–Damgård construction and the sponge construction.

The study covers the analysis of 12 hash functions and their families that employ one of these constructions. Each hash function was evaluated according to key characteristics, such as hardware complexity, hash output length, cryptographic resistance, throughput, and energy consumption. Thus, comparative data were obtained that made it possible to identify the strengths and weaknesses of each hash function in the context of their implementation for lightweight systems. To generalize the results, an integral efficiency coefficient is proposed, which accounts for the influence of four key parameters with corresponding weights.

The comparative analysis showed that hash functions based on the Merkle–Damgård construction, although demonstrating an acceptable level of cryptographic resistance, require significant hardware costs, which limits their application in resource-constrained systems. In contrast, algorithms developed based on the sponge construction demonstrated the highest values of the integral efficiency coefficient, confirming the best compromise between security, performance, and hardware costs. Specifically, SPONGENT-88 stands out with minimal hardware requirements, and HVH provides high throughput and cryptographic resistance with comparatively low costs. The DeeR-Hash and Hash-One hash functions, which use combined schemes, require the lowest hardware costs for 160-bit hash values while maintaining a high level of cryptographic resistance. The analysis confirms that further research in the direction of the sponge construction and combined schemes with shift registers is the most promising, as these approaches provide the best ratio between cryptographic resistance, performance, and hardware efficiency.

Keywords: lightweight cryptography, hash function, Merkle–Damgård construction, sponge construction, hashing scheme, cryptographic characteristics, technical characteristic.

Luzhetskyi Volodymyr A. — Dr. Sc. (Eng.), Professor, Head of the Chair of Information Security, e-mail: lva.kzi2002@gmail.com ;

Seleznov Vitalii I. — Post-Graduate Student of the Chair of Information Security, e-mail: seleznov.vitalii@gmail.com