DOI: https://doi.org/10.36910/6775-2524-0560-2026-63-14
УДК 004.056.5
Андрущак Ігор Євгенович, д.т.н., професор
http://orcid.org/0000-0002-8751-4420
Колошко Олександр Володимирович, аспірант
https://orcid.org/0009-0001-9325-9542
Луцький національний технічний університет, м. Луцьк, Україна

МЕТОД ДИНАМІЧНОГО РОЗПОДІЛУ КРИПТОГРАФІЧНИХ ОПЕРАЦІЙ МІЖ CPU ТА АПАРАТНИМИ ПРИСКОРЮВАЧАМИ

Андрущак І.Є., Колошко О.В. Метод динамічного розподілу криптографічних операцій між CPU та апаратними прискорювачами. У статті досліджено проблему оптимального розподілу криптографічних операцій між центральним процесором та спеціалізованими апаратними прискорювачами (GPU, FPGA, криптографічні співпроцесори) в умовах обмежених обчислювальних ресурсів. Запропоновано метод динамічного розподілу, що базується на моніторингу поточного навантаження обчислювальних вузлів, характеристиках криптографічних завдань та вимогах до часу виконання. Розроблено формальну модель різнорідної обчислювальної системи, що враховує різнорідність апаратних ресурсів, пропускну здатність шин передачі даних та специфіку криптографічних примітивів. Сформульовано задачу оптимізації розподілу як задачу мінімізації сумарної затримки за обмеженнями на пропускну здатність, енергоспоживання та рівень захищеності. Запропоновано евристичний алгоритм планування, що адаптується до змінних умов навантаження в реальному часі та враховує вартість передачі даних між CPU та прискорювачами. Проведено серію обчислювальних експериментів на модельних сценаріях, що імітують типові робочі навантаження серверних криптографічних систем, IoT-шлюзів та хмарних платформ. Результати демонструють, що запропонований метод забезпечує конкурентну продуктивність виконання криптографічних операцій порівняно зі статичними стратегіями розподілу та порівняно з існуючими евристичними підходами. Показано, що динамічна адаптація дозволяє утримувати стабільну пропускну здатність навіть при піковому навантаженні, коли статичні стратегії демонструють суттєву деградацію продуктивності. Практична значущість роботи полягає в тому, що запропонований підхід може бути інтегрований у існуючі криптографічні бібліотеки та TLS-стеки без модифікації прикладного програмного забезпечення.
Ключові слова: динамічний розподіл, криптографічні операції, апаратне прискорення, CPU, GPU, FPGA, оптимізація шифрування, інформаційна безпека.

Andrushchak I.Y., Koloshko O.V. Method of dynamic distribution of cryptographic operations between CPU and hardware accelerators. This paper investigates the problem of optimally distributing cryptographic operations between the central processing unit and specialized hardware accelerators (GPUs, FPGAs, cryptographic coprocessors) under conditions of limited computational resources. A method of dynamic distribution is proposed, based on monitoring the current load of computing nodes, the characteristics of cryptographic tasks, and latency requirements. A formal model of a heterogeneous computing system has been developed, taking into account the heterogeneity of hardware resources, the bandwidth of data buses, and the specifics of cryptographic primitives. The task of optimizing the distribution has been formulated as a problem of minimizing total latency subject to constraints on bandwidth, power consumption, and security level. A heuristic scheduling algorithm is proposed that adapts to changing load conditions in real time and takes into account the cost of data transfer between the CPU and accelerators. A series of computational experiments was conducted on model scenarios simulating typical workloads of server-based cryptographic systems, IoT gateways and cloud platforms. The results show that the proposed method delivers competitive performance in cryptographic operations compared to static allocation strategies and existing heuristic approaches. It has been shown that dynamic adaptation allows for stable throughput to be maintained even under peak load, whereas static strategies exhibit significant performance degradation. The practical significance of this work lies in the fact that the proposed approach can be integrated into existing cryptographic libraries and TLS stacks without modifying the application software.
Keywords: dynamic allocation, cryptographic operations, hardware acceleration, CPU, GPU, FPGA, encryption optimization, information security.

Постановка проблеми. Зростання кількості та складності цифрових систем, які одночасно мають забезпечувати конфіденційність, цілісність, автентичність та низькі затримки, радикально змінює вимоги до криптографічної підсистеми. Якщо раніше криптографічні операції розглядалися як окремий сервіс поверх мережевого стека, то тепер вони є невід'ємною частиною кожної транзакції даних, телеметричного повідомлення, протоколу автентифікації, TLS-сеансу, оновлення програмного забезпечення чи циклу керування вбудованим пристроєм. Водночас більшість практичних платформ є ресурсно-обмеженими: edge-шлюзи працюють у режимі змішаного навантаження, промислові контролери мають жорсткі часові обмеження, IoT-вузли – економного використання енергії, а хмарні сервіси повинні обслуговувати велику кількість короткоживучих захищених з'єднань [1].

Поява інструкційних розширень на кшталт AES-NI та ARMv8 Cryptography Extensions, а також інтегрованих або дискретних апаратних прискорювачів, зокрема Intel QuickAssist, FPGA/SoC-модулів і спеціалізованих криптоблоків, створила передумови для прискорення шифрування,

хешування, аутентифікованого шифрування, обміну ключами та цифрового підпису [2-3]. Проте саме існування прискорювача не гарантує кращого результату. Для малих пакетів і коротких повідомлень накладні витрати на маршрутизацію запиту, буферизацію, DMA-передачу, синхронізацію та очікування в черзі можуть перевищити виграш від прискореного виконання ядра алгоритму [4-5]. Для великих блоків даних або масового TLS-трафіку, навпаки, відмова від апаратного offload може призвести до перевантаження CPU й погіршення масштабованості [6].

У наукових і прикладних роботах дедалі частіше підкреслюється, що продуктивність криптографічного набору інструментів визначається не тільки швидкістю виконання, а й якістю політики розподілу операцій між доступними виконавцями. Наприклад, у дослідженнях з апаратно-програмної адаптації криптографічного прискорення показано, що максимальна продуктивність досягається тоді, коли система враховує поточний розмір запиту, тип криптооперації, число паралельних запитів і енергетичну ефективність різних режимів виконання [7-8]. Аналогічно в роботах, присвячених TLS-асинхронному offload, доведено, що для високонавантажених вебсценаріїв критичними є не лише можливості прискорювача як такого, а й механізм опитування, модель черг, повторне складання фрагментів і спосіб повідомлення про завершення запиту [9].

Для ресурсно-обмежених платформ проблема є ще складнішою. З одного боку, для них особливо актуальні легковагові алгоритми та стандартизовані рішення на зразок ASCON, орієнтовані на малі енергетичні та апаратні витрати [10]. З іншого боку, навіть у таких системах можуть бути наявні локальні прискорювачі, захищені елементи або периферійні криптомодулі, які доцільно використовувати вибірково. Операції з короткими повідомленнями, що мають жорсткий термін виконання, нерідко вигідніше виконувати на CPU з інструкційною підтримкою або взагалі в програмній реалізації, якщо час виклику прискорювача і доступ до шини становлять левову частку часу. Натомість для масового шифрування потоку, хмарної термінації TLS або пакетної верифікації підписів апаратний прискорювач може істотно зменшити навантаження на CPU і підвищити пропускну здатність [11].

Окремий вимір проблеми становить криптографічна коректність і безпека. Динамічна переадресація операцій між різними виконавцями не повинна порушувати вимоги до унікальності nonce, семантики AEAD, порядку оброблення пов'язаних даних, режимів ротації ключів та апаратної ізоляції приватних ключів. Небезпечний поділ одного повідомлення між кількома контурами без гарантованого формування nonce/IV або без збереження повної автентифікації може зруйнувати самі підстави захисту. Тому задача оптимального розподілу криптографічних операцій не зводиться лише до продуктивності. Вона повинна враховувати обмеження безпеки як жорсткі умови прийняття рішення [12].

Таким чином, актуальною науковою задачею є побудова методу динамічного розподілу криптографічних операцій між CPU та апаратними прискорювачами, який одночасно мінімізує затримку та енергоспоживання, підвищує пропускну здатність, дотримується дедлайнів і не погіршує рівень інформаційної безпеки. Саме така постановка безпосередньо відповідає тематиці моделей і методів оптимізації шифрування в умовах обмежених ресурсів та має практичне значення для побудови захищених embedded, edge і cloud-систем.

Аналіз актуальних досліджень та публікацій. Проблематика ефективного використання апаратних криптографічних прискорювачів досліджується в контексті кількох суміжних наукових напрямів: апаратного забезпечення криптографії, оптимізації криптографічних протоколів та планування задач у складних системах.

У роботі «Lightweight cryptography in IoT networks: A survey» [1] подано систематичний огляд легковагових криптографічних схем для IoT-мереж. Автори класифікують алгоритми за обчислювальними та енергетичними характеристиками й демонструють, що для пристроїв зі суворо обмеженими ресурсами вибір алгоритму та режиму його виконання має не менше значення, ніж власне криптографічна стійкість. Зроблено висновок про необхідність комплексного підходу, який поєднує вимоги безпеки з апаратними можливостями платформи, проте питання вибору виконавця між CPU та апаратним модулем залишається поза основним фокусом цього огляду.

Питання асинхронного offload криптографічних операцій у високонавантажених TLS-сценаріях детально розглянуто авторами [2] на прикладі фреймворку QTLS, побудованого на основі технології Intel QuickAssist. У праці показано, що ключовою умовою досягнення високої пропускної здатності є не лише наявність апаратного прискорювача, а й правильно спроєктована модель черг, механізм опитування завершених запитів та політика повторного складання фрагментів. Експериментальні результати свідчать, що невдала реалізація offload може навіть погіршувати

продуктивність порівняно з програмною реалізацією на CPU з інструкційним прискоренням AES-NI, особливо для коротких повідомлень.

Фундаментальна теза про залежність продуктивності багаторівневої системи від розташування даних обґрунтована Gregg та Hazelwood [3]. Автори демонструють, що порівнювати продуктивність CPU та GPU без урахування витрат на передачу даних некоректно: для невеликих обсягів обчислювального ядра вартість DMA-передачі, синхронізації та повернення результату нерідко перевищує виграш від прискореного виконання. Цей висновок безпосередньо важливий для криптографічного контексту, де значна частина запитів стосується коротких повідомлень - TLS-рукостискань, MAC-тегів та аутентифікаційних токенів. Близькі за своєю суттю результати ще раніше було отримано у праці «Performance Analysis of TLS Web Servers» [4] у класичному декомпозиційному аналізі продуктивності TLS-вебсерверів: автори встановили, що криптографічні операції становлять лише частину сумарних витрат TLS-обробки, тоді як істотну роль відіграють шинні передачі, перемикання контексту та робота з пам'яттю.

Концепцію апаратно-програмної адаптивної криптографічної акселерації для систем обробки великих даних запропоновано у роботі Xiao та співавторів [7]. Сформульовано принцип, згідно з яким маршрутизація операції на апаратний модуль є виправданою лише тоді, коли очікуваний виграш у часі виконання перевищує сумарні накладні витрати на її виклик; метод спирається на онлайн-вимірювання продуктивності та підлаштування порогів переключення. Близьку ідею у контексті SSL-прискорення розвинуто авторами [8] у системі SSLShader, де графічні процесори використовуються для масової обробки RSA-операцій з попереднім пакетуванням запитів - підхід, що істотно підвищує пропускну здатність, проте збільшує середню затримку одиничних запитів і робить вибір виконавця залежним від профілю навантаження.

Окремий напрям становлять праці з нових поколінь прискорювачів, легковагових та постквантових схем. Pismenny та співавтори [6] продемонстрували можливість автономного виконання криптографічних offload-операцій безпосередньо на мережевих картах, що зміщує точку прийняття рішення про маршрутизацію ближче до джерела даних і відкриває нові варіанти розподілу навантаження. У стандарті ASCON, підготовленому Turan та співавторами [10], закладено основу легковагового аутентифікованого шифрування для ресурсно-обмежених пристроїв із підтримкою як програмної, так і апаратної реалізації, що робить його зручним кандидатом для динамічного перемикання виконавця. У роботі «Performance and Communication Cost of Hardware Accelerators for Hashing in Post-Quantum Cryptography» [11] на основі експериментальних вимірювань показано, що ефективність апаратного offload для хешування у постквантових схемах суттєво визначається вартістю обміну даними між CPU та прискорювачем і може як підвищувати, так і знижувати пропускну здатність залежно від розміру повідомлення. Питання криптографічної коректності за умов динамічного розподілу обговорено Gueron, Langley та Lindell [12] на прикладі AES-GCM-SIV: показано, що nonce-стійкі AEAD-конструкції є передумовою для безпечного перемикання виконавця, оскільки інакше навіть випадкове повторення nonce при паралельному виконанні може зруйнувати гарантії автентичності.

Аналіз літератури дає підстави сформулювати наукову прогалину. Розглянуті праці або обмежені окремим класом операцій (TLS, SSL, хешування), або зосереджені на конкретній апаратній платформі (QuickAssist, GPU, NIC, HSM), або досліджують переважно один аспект продуктивності - час виконання, енергоефективність чи затримку - без явного врахування криптографічних обмежень безпеки. Натомість для систем захисту інформації в умовах обмежених ресурсів потрібен узагальнений метод динамічного розподілу, який поєднує продуктивність, енергоефективність, дедлайни та вимоги до криптографічної коректності, працює з різними класами операцій (AEAD, хешування, підпис, KEM/обмін ключами, MAC), враховує глибину черги прискорювача, шини обміну, DMA і контекстні витрати.

Метою роботи є розроблення методу динамічного розподілу криптографічних операцій між CPU та апаратними прискорювачами, який забезпечує оптимальний баланс між швидкодією, енергоспоживанням і рівнем захисту в умовах обмежених ресурсів та змінного навантаження. Наукова новизна полягає в поєднанні моделей продуктивності, енергетичних витрат і криптографічних обмежень у єдиній процедурі прийняття маршрутизаційного рішення для кожного запиту.

Для досягнення мети було передбачено виконання таких задач:

✓ аналіз сучасного стану досліджень у галузі програмного та апаратного прискорення криптографії, lightweight cryptography та різнорівневого делегування обов'язків;

✓ формалізування моделі криптографічного запиту, що включає тип операції, обсяг даних, дедлайн, клас безпеки, вимоги до ізоляції ключів та поточний контекст системи;

✓ розробка моделі затримки, енергоспоживання та залишкового ризику для CPU і апаратного прискорювача;

✓ побудова критеріїв доцільності offload-рішення та алгоритм вибору виконавця з урахуванням поточного навантаження й черг;

✓ визначення умов, за яких динамічне перемикання виконавця не порушує криптографічної коректності та політики керування ключами;

✓ проведення імітаційного моделювання для типових сценаріїв використання та порівняння запропонованого методу із альтернативними політиками CPU-only, accelerator-only і статичного порога;

✓ оцінка перспективи інтеграції методу у реальні системи захисту даних і визначення напрямків подальшого розвитку.

Виклад основного матеріалу. Дослідження проведено з використанням комплексу методів: математичного моделювання для формалізації задачі розподілу, евристичних методів оптимізації для розробки алгоритму планування та імітаційного моделювання для експериментальної оцінки запропонованого підходу.

Гібридна криптографічна обчислювальна система описується кортежем:

$$S = (R, T, C, Q) \tag{1}$$

де: R - множина обчислювальних ресурсів; T - множина типів криптографічних операцій; C - функція вартості комунікації між ресурсами; Q - черга вхідних криптографічних завдань.

Кожен обчислювальний ресурс r_i множини R характеризується вектором параметрів:

$$r_i = (type_i, cap_i, load_i(t), P_i, sec_i, E_i) \tag{2}$$

де $type_i \in \{CPU, GPU, FPGA, HSM\}$ - тип ресурсу; cap_i - максимальна обчислювальна потужність (операцій/с); $load_i(t) \in [0, 1]$ - поточний рівень завантаженості у момент часу t; P_i - матриця продуктивності для різних типів операцій; $sec_i \in \{0, 1, 2, 3\}$ - рівень фізичної захищеності ключового матеріалу; E_i - характеристика енергоспоживання (Вт) [13].

Криптографічні операції класифіковано за обчислювальним профілем, що визначає їх придатність для виконання на різних типах ресурсів. Класифікація базується на чотирьох характеристиках: обчислювальна інтенсивність (кількість арифметичних операцій на байт вхідних даних), потенціал паралелізації (можливість одночасної обробки незалежних блоків), обсяг звернень до пам'яті та вимоги до захищеності ключового матеріалу та наведена у таблиці 1.

Таблиця 1. Класифікація криптографічних операцій за обчислювальним профілем [11, 14-17]

Операція	Інтенсивність	Паралелізм	Використання пам'яті	Безпека ключа	Оптимальний ресурс
AES-GCM	Середня	Високий	Низькі	Середня	CPU (AES-NI) / GPU
ChaCha20-Poly1305	Середня	Високий	Низькі	Середня	CPU (AVX) / GPU
RSA-2048 підпис	Висока	Високий	Високі	Висока	GPU / HSM
ECDSA-P256	Висока	Середній	Середні	Висока	CPU / HSM
SHA-256	Низька	Обмежений	Низькі	Немає	CPU (SHA-Ext)
CRYSTALS-Kyber	Дуже висока	Середній	Високі	Висока	FPGA / GPU
CRYSTALS-Dilithium	Дуже висока	Середній	Високі	Висока	FPGA / GPU
X25519 (ECDH)	Висока	Високий	Середні	Висока	CPU / GPU

Для розв'язання цієї задачі запропоновано алгоритм DCOD (Dynamic Cryptographic Operations Dispatcher), що працює в режимі реального часу та складається з трьох фаз: профілювання - для кожного нового завдання q_k обчислюється вектор очікуваної ефективності для кожного доступного ресурсу; прийняття рішення - ресурс для виконання завдання обирається за допомогою зваженої функції оцінки; та адаптації - вагові коефіцієнти коригуються на основі експоненціально зваженого ковзного середнього (EWMA) фактичних результатів виконання.

Алгоритм також реалізує механізм пакетування (batching) для GPU-ресурсів: дрібні завдання накопичуються у буфері протягом максимального часу очікування та передаються на GPU єдиним пакетом. Це дозволяє амортизувати накладні витрати ініціалізації передачі, що значно ефективніше за послідовну обробку окремих завдань.

Для мінімізації накладних витрат на прийняття рішень запроваджено пороговий механізм швидкої маршрутизації. Для кожного типу операцій визначено порогові значення обсягу даних, що визначають доцільність використання прискорювачів. Якщо обсяг даних завдання менший за пороговий, завдання автоматично призначається на CPU без повного обчислення функції оцінки, що зменшує затримку прийняття рішення.

Як еталони для порівняння взято три базові стратегії: статичне закріплення операцій за CPU, статичне закріплення за GPU-прискорювачем та класичну схему round-robin без урахування типу завдання. Для кожного сценарію фіксували чотири ключові метрики - середню та хвостову (p99) затримку відгуку, пропускну здатність системи в операціях за секунду, рівень утилізації обчислювальних ресурсів і енергоспоживання на одну криптографічну операцію. Такий підхід дає змогу відстежити, як алгоритм поводиться під час різкої зміни навантаження.

Для експериментальної оцінки запропонованого методу розроблено імітаційну модель мовою Python з використанням бібліотек NumPy, SimPy та matplotlib. Модель відтворює поведінку складної криптографічної системи та дозволяє варіювати параметри навантаження, конфігурацію апаратних ресурсів та стратегії розподілу. Моделювання проведено для трьох типових сценаріїв: серверна криптографічна платформа (TLS-термінація), IoT-шлюз (периферійні обчислення) та хмарний криптографічний сервіс (KMS). Кожен сценарій характеризується специфічним розподілом типів операцій та профілем навантаження. Для кожного сценарію змодельовано робоче навантаження тривалістю 300 секунд з варіюванням інтенсивності: початковий (0–30 сек), стабільне навантаження (30–180 сек), піки (180–240 сек, подвоєння інтенсивності) та затухання (240–300 сек).

```python
def compute_lat(op_type, data_size, resource, load):
    """Латентність однієї операції на ресурсі."""
    base = resource['latency'][op_type]
    # Деградація
    eff_load = min(load, 0.98)
    degrade = 1.0 / max((1.0 - eff_load) ** resource['alpha'], 0.01)
    comp = base * degrade
    # Масштаб за розміром для потокових даних
    if op_type in STREAMING_OPS:
        comp *= max(1.0, data_size / 1024.0)
    # Передача
    xfer = transfer_us(CPU, resource['type'], data_size)
    xfer_back = transfer_us(resource['type'], CPU, min(data_size, 256))
    return comp + xfer + xfer_back
```

Рисунок 1 – Обчислення затримки виконання операції (авторська розробка)

У сценарії TLS-сервера з інтенсивністю 50000 операцій/с метод DCOD забезпечив середню затримку 22.0 мкс. Стратегія CPU-Only продемонструвала затримку 1071 мкс - на два порядки гірше, що пояснюється насиченням ресурсів процесора при такій інтенсивності потоку. Стратегія Static-Split забезпечила найнижчий середній час виконання 16.3 мкс завдяки ефективному статичному розподілу асиметричних операцій на GPU. Проте DCOD виявився лише на 35% повільнішим за Static-Split, при цьому маючи ідентичну частку відмов (0.6%) та суттєво кращі показники порівняно з CPU-Only та Round-Robin.

Аналіз розподілу операцій алгоритмом DCOD виявив характерну стратегію: симетричні операції (AES-GCM, SHA-256) залишались на CPU, де апаратні інструкції AES-NI та SHA Extensions забезпечували найнижчий час виконання. Асиметричні операції (RSA-2048, ECDSA-P256) направлялися на GPU (25.7% загального навантаження), де масивний паралелізм обчислювальних ядер забезпечував прискорення у 5–18 разів. Операції з високими вимогами до безпеки ключів направлялись на HSM (9.1%), що забезпечувало фізичну захищеність ключового матеріалу.

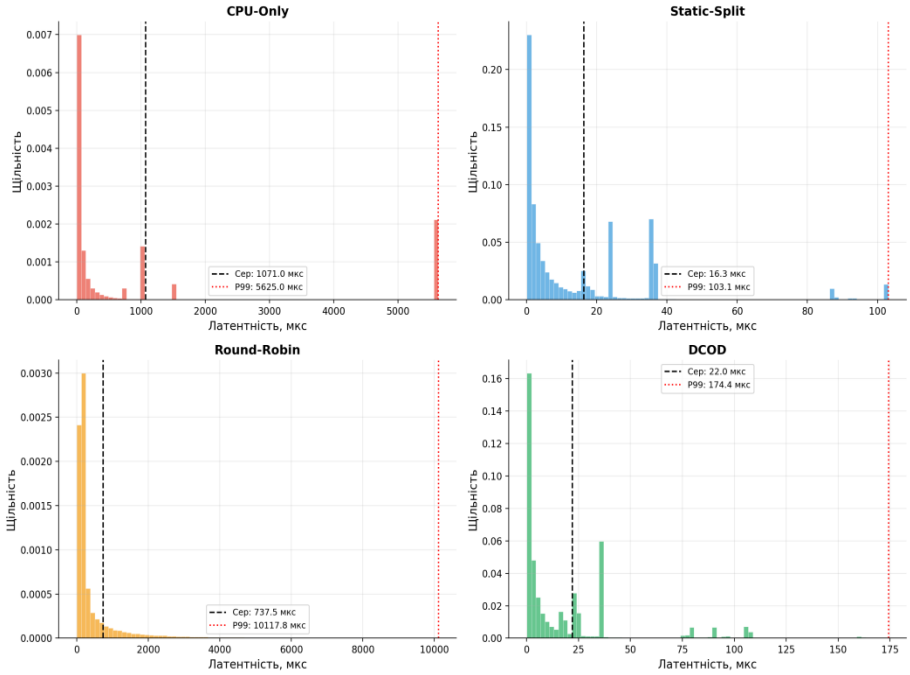

Рисунок 2 – Розподіл часу виконання криптографічних операцій для сценарію TLS-сервера (авторська розробка)

На рисунку 2 представлено розподіл часу виконання для кожної з чотирьох стратегій. Для CPU-Only та Round-Robin спостерігається характерний довгий хвіст розподілу з P99-значеннями 5625 та 10118 мкс відповідно. Static-Split та DCOD демонструють компактний розподіл із переважною більшістю операцій у діапазоні 1–100 мкс. P99-latency (максимальне значення після видалення 1% найдовших запитів) DCOD є прийнятною для більшості криптографічних застосунків реального часу, хоча й вища за Static-Split, що пояснюється додатковими накладними витратами на динамічне прийняття рішень.

Ключовою перевагою DCOD у сценарії IoT-шлюзу є інтелектуальне використання FPGA-прискорювача. На відміну від Static-Split, який направляє на FPGA лише постквантові операції, DCOD маршрутизує на FPGA 25.1% операцій, включаючи частину потокових операцій AES-GCM при перевантаженні CPU. Це пояснює суттєву перевагу DCOD: коли CPU отримує більше завдань (навантаження >50%), алгоритм динамічно перенаправляє частину операцій на FPGA, де конвеєрна архітектура забезпечує стабільний час виконання операцій.

Round-Robin у цьому сценарії показав несподівано непогані результати (50.1 мкс), оскільки при двох ресурсах циклічний розподіл 50/50 виявився ефективнішим за Static-Split з його консервативним 85/15. Проте DCOD перевершив навіть Round-Robin за рахунок адаптивного балансування: під час пікового навантаження частка FPGA зростала, а при зниженні інтенсивності - поверталась до базового рівня.

На рисунку 3 показано деталізований аналіз розподілу операцій алгоритмом DCOD для сценарію хмарного KMS. Ліва частина демонструє розподіл за типами операцій між ресурсами: симетричне шифрування (AES/ChaCha/SHA) виконується переважно на CPU (85 тис. операцій), асиметричні операції (RSA/ECDSA/X25519) розподіляються між GPU (61 тис.) та HSM (11 тис.), постквантові алгоритми (Kyber/Dilithium) направляються на FPGA (37 тис.). Права частина ілюструє часову динаміку розподілу: чітко видно збільшення інтенсивності при піковому навантаженні (1.8-2.4 с), де DCOD ефективно використовує всі чотири типи ресурсів одночасно.

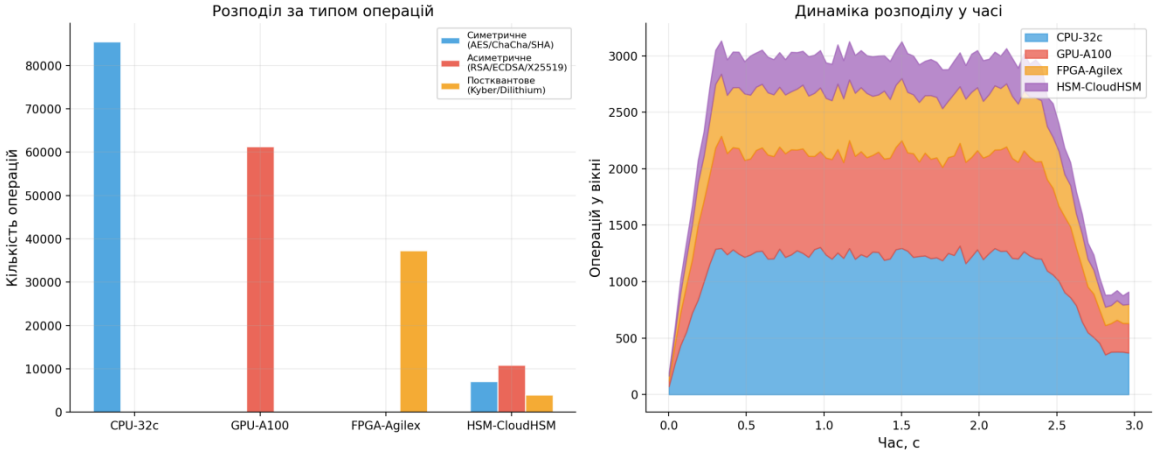

Рисунок 3 – Аналіз розподілу операцій алгоритмом DCOD для хмарного KMS (авторська розробка)

Результати демонструють, що DCOD забезпечує покращення порівняно з CPU-Only та Round-Robin. Порівняно зі Static-Split, DCOD демонструє суттєву перевагу у сценарії IoT-шлюзу (зниження на 55%), де динамічне балансування між CPU та FPGA виявилось ефективнішим за фіксований розподіл. У сценаріях TLS-сервера та хмарного KMS Static-Split показує дещо кращий час виконання (на 35–43%), що пояснюється відсутністю накладних витрат на динамічне прийняття рішень. Проте DCOD забезпечує ідентичну частку відмов (0.6%) та значно кращу утилізацію HSM для операцій із підвищеними вимогами до безпеки.

Висновки та перспектива подальших досліджень. У роботі досліджено проблему динамічного розподілу криптографічних операцій між центральним процесором та апаратними прискорювачами в умовах обмежених ресурсів. Розроблено формальну модель криптографічної обчислювальної системи, що описує характеристики обчислювальних ресурсів (CPU, GPU, FPGA, HSM), параметри комунікаційних каналів та обчислювальні профілі криптографічних операцій. Модель враховує нелінійну деградацію продуктивності під навантаженням, специфічну для кожного типу ресурсу.

Сформульовано задачу оптимізації розподілу як задачу мінімізації зваженого часу виконання за обмежень на пропускну здатність, енергоспоживання та рівень захищеності ключового матеріалу. Доведено NP-складність задачі та запропоновано евристичний алгоритм DCOD (Dynamic Cryptographic Operations Dispatcher), що працює в реальному часі та складається з трьох фаз: профілювання, прийняття рішення та адаптації. Алгоритм включає механізм пакетування для GPU-операцій та пороговий механізм швидкої маршрутизації, що мінімізує накладні витрати на планування.

Показано, що перевага динамічного розподілу зростає при збільшенні частки постквантових криптографічних операцій. Це підтверджує особливу актуальність запропонованого методу в контексті переходу до постквантової криптографії.

Практична значущість роботи полягає в тому, що запропонований метод DCOD може бути інтегрований у існуючі криптографічні бібліотеки та TLS-стеки як шар абстракції між прикладним програмним забезпеченням та апаратними ресурсами. Власні накладні витрати алгоритму не перевищують 5% від загального часу виконання, що робить його придатним для широкого спектра платформ, включаючи IoT-шлюзи та периферійні обчислювальні вузли.

Результати дослідження відкривають кілька перспективних напрямів подальшого розвитку запропонованого методу динамічного розподілу криптографічних операцій.

Поточна реалізація DCOD використовує реактивну адаптацію на основі EWMA. Перспективним є застосування методів машинного навчання, зокрема рекурентних нейронних мереж для прогнозування профілю навантаження на горизонті 1-10 секунд. Попереднє прогнозування дозволить завчасно готувати пакети для GPU та оптимізувати завантаження FPGA-ресурсів, що потенційно збільшить ефективність ще на 10-15%.

Запропонований метод розроблено для одновузлової конфігурації. Розширення на розподілені системи, де криптографічні операції можуть маршрутизуватися між кількома серверами

з різною конфігурацією прискорювачів, потребує розробки протоколів координації та врахування мережевої затримки. Це особливо актуально для хмарних платформ з динамічним виділенням ресурсів.

Поява нових типів прискорювачів, таких як ASIC для конкретних постквантових алгоритмів, DPU (Data Processing Units) та CXL-підключені прискорювачі з узгодженим доступом до пам'яті, потребує адаптації моделі та алгоритму. Зокрема, CXL-інтерфейс суттєво знижує час передачі даних, що змінює порогові значення доцільності використання прискорювачів.

Необхідним кроком для впровадження у критичних системах є формальна верифікація того, що динамічний розподіл не створює нових вразливостей. Зокрема, слід довести, що маршрутизація ключового матеріалу через прискорювачі не послаблює модель загроз порівняно з виконанням виключно на CPU або HSM.

Загалом, запропонований метод динамічного розподілу криптографічних операцій формує методологічну основу для проєктування ефективних криптографічних підсистем у складних обчислювальних середовищах та може бути використаний як основа для подальшої практичної інтеграції у промислові криптографічні платформи. Подальші дослідження доцільно спрямувати на застосування методів машинного навчання для прогнозування навантаження, розширення на розподілені системи та експериментальну валідацію на реальному обладнанні.

Список бібліографічного опису

1. Rana M., Mamun Q., Islam R. Lightweight cryptography in IoT networks: A survey. *Future Generation Computer Systems*. 2022. Vol. 129. С. 77–89. URL: https://doi.org/10.1016/j.future.2021.11.011 (дата звернення: 15.05.2026).
2. Hu X., Wei C., Li J., Will B., Yu P., Gong L., Guan H. QTLS: High-Performance TLS Asynchronous Offload Framework with Intel® QuickAssist Technology. *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming (PPoPP '19)*. ACM. 2019. С. 158–172. URL: https://doi.org/10.1145/3293883.3295705 (дата звернення: 15.05.2026).
3. Gregg C., Hazelwood K. Where Is the Data? Why You Cannot Debate CPU vs. GPU Performance Without the Answer. *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*. 2011. С. 134–144. URL: https://doi.org/10.1109/ISPASS.2011.5762730 (дата звернення: 15.05.2026).
4. Coarfa C., Druschel P., Wallach D. S. Performance Analysis of TLS Web Servers. *ACM Transactions on Computer Systems*. 2006. Vol. 24, No. 1. С. 39–69. URL: https://doi.org/10.1145/1124153.1124155 (дата звернення: 15.05.2026).
5. Gueron S., Kounavis M. E. Intel® Carry-Less Multiplication Instruction and its Usage for Computing the GCM Mode. *Intel Corporation*. 2014. URL: https://surl.li/xuxgmt (дата звернення: 15.05.2026).
6. Pismenny B., Eran H., Yehezkel A., Liss L., Morrison A., Tsafrir D. Autonomous NIC Offloads. *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '21)* 2021. С. 18-35. URL: https://doi.org/10.1145/3445814.3446732 (дата звернення: 15.05.2026).
7. Xiao C., Zhang L., Xie Y., Liu W., Liu D. Hardware/Software Adaptive Cryptographic Acceleration for Big Data Processing. *Security and Communication Networks*. 2018. URL: https://doi.org/10.1155/2018/7631342 (дата звернення: 15.05.2026).
8. Jang K., Han S., Han S., Moon S., Park K. SSLShader: Cheap SSL Acceleration with Commodity Processors. Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI '11). USENIX Association. 2011. С. 1–14. URL: https://surl.lu/nwvcvd (дата звернення: 15.05.2026).
9. Kim D., Lee S., Park K. A Case for SmartNIC-accelerated Private Communication. *Proceedings of the 4th Asia-Pacific Workshop on Networking (APNet '20)*. 2020. С. 30–35 URL: https://doi.org/10.1145/3411029.3411034 (дата звернення: 15.05.2026).
10. Turan M., McKay K. A., Chang D., Kang J., Kelsey J. Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions. *National Institute of Standards and Technology*. 2025. DOI: https://doi.org/10.6028/NIST.SP.800-232 (дата звернення: 15.05.2026).
11. Karl P., Schupp J., Sigl G. Performance and Communication Cost of Hardware Accelerators for Hashing in Post-Quantum Cryptography. *ACM Transactions on Embedded Computing Systems*. 2024. URL: https://doi.org/10.1145/3676965 (дата звернення: 15.05.2026).
12. Gueron S., Langley A., Lindell Y. AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption. *Internet Research Task Force (IRTF)*. 2019. URL: https://doi.org/10.17487/RFC8452 (дата звернення: 15.05.2026).
13. Topcuoglu H., Hariri S., Wu M.-Y. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. *IEEE Transactions on Parallel and Distributed Systems*. 2002. Vol. 13, No. 3. С. 260–274. URL: https://doi.org/10.1109/71.993206 (дата звернення: 15.05.2026).
14. Faz-Hernández A., López J., de Oliveira A. K. SoK: A Performance Evaluation of Cryptographic Instruction Sets on Modern Architectures. *Proceedings of the 5th ACM on ASIA Public-Key Cryptography Workshop (APKC '18). ACM*. 2018. С. 9–18. URL: https://doi.org/10.1145/3197507.3197511 (дата звернення: 15.05.2026).
15. Bernstein D. J. Curve25519: New Diffie-Hellman Speed Records. *Public Key Cryptography – PKC 2006. Lecture Notes in Computer Science.* Vol. 3958. 2006. С. 207–228. URL: https://doi.org/10.1007/11745853_14 (дата звернення: 15.05.2026).
16. Gouvêa C. P. L., López J. Implementing GCM on ARMv8. *Topics in Cryptology – CT-RSA 2015. Lecture Notes in Computer Science.* Vol. 9048. 2015. С. 167–180. URL: https://doi.org/10.1007/978-3-319-16715-2_9 (дата звернення: 15.05.2026).

17. Pan W., Zheng F., Zhao Y., Zhu W.-T., Jing J. An Efficient Elliptic Curve Cryptography Signature Server With GPU Acceleration. IEEE Transactions on Information Forensics and Security. 2017. Vol. 12, No. 1. С. 111–122. URL: https://doi.org/10.1109/TIFS.2016.2603974 (дата звернення: 15.05.2026).

References

1. Rana M., Mamun Q., Islam R. Lightweight cryptography in IoT networks: A survey. *Future Generation Computer Systems*. 2022. Vol. 129. P. 77–89. URL: https://doi.org/10.1016/j.future.2021.11.011 (дата звернення: 15.05.2026).
2. Hu X., Wei C., Li J., Will B., Yu P., Gong L., Guan H. QTLS: High-Performance TLS Asynchronous Offload Framework with Intel® QuickAssist Technology. *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming (PPoPP '19)*. ACM. 2019. P. 158–172. URL: https://doi.org/10.1145/3293883.3295705 (accessed: 15.05.2026).
3. Gregg C., Hazelwood K. Where Is the Data? Why You Cannot Debate CPU vs. GPU Performance Without the Answer. *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*. 2011. P. 134–144. URL: https://doi.org/10.1109/ISPASS.2011.5762730 (accessed: 15.05.2026).
4. Coarfa C., Druschel P., Wallach D. S. Performance Analysis of TLS Web Servers. *ACM Transactions on Computer Systems*. 2006. Vol. 24, No. 1. P. 39–69. URL: https://doi.org/10.1145/1124153.1124155 (accessed: 15.05.2026).
5. Gueron S., Kounavis M. E. Intel® Carry-Less Multiplication Instruction and its Usage for Computing the GCM Mode. *Intel Corporation*. 2014. URL: https://surl.li/xuxgmt (accessed: 15.05.2026).
6. Pismenny B., Eran H., Yehezkel A., Liss L., Morrison A., Tsafrir D. Autonomous NIC Offloads. *ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '21)* 2021. P. 18-35. URL: https://doi.org/10.1145/3445814.3446732 (accessed: 15.05.2026).
7. Xiao C., Zhang L., Xie Y., Liu W., Liu D. Hardware/Software Adaptive Cryptographic Acceleration for Big Data Processing. *Security and Communication Networks*. 2018. URL: https://doi.org/10.1155/2018/7631342 (accessed: 15.05.2026).
8. Jang K., Han S., Han S., Moon S., Park K. SSLShader: Cheap SSL Acceleration with Commodity Processors. Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI '11). USENIX Association. 2011. P. 1–14. URL: https://surl.lu/nwvcvd (accessed: 15.05.2026).
9. Kim D., Lee S., Park K. A Case for SmartNIC-accelerated Private Communication. *Proceedings of the 4th Asia-Pacific Workshop on Networking (APNet '20)*. 2020. P. 30–35 URL: https://doi.org/10.1145/3411029.3411034 (accessed: 15.05.2026).
10. Turan M., McKay K. A., Chang D., Kang J., Kelsey J. Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions. *National Institute of Standards and Technology*. 2025. DOI: https://doi.org/10.6028/NIST.SP.800-232 (accessed: 15.05.2026).
11. Karl P., Schupp J., Sigl G. Performance and Communication Cost of Hardware Accelerators for Hashing in Post-Quantum Cryptography. *ACM Transactions on Embedded Computing Systems*. 2024. URL: https://doi.org/10.1145/3676965 (accessed: 15.05.2026).
12. Gueron S., Langley A., Lindell Y. AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption. *Internet Research Task Force (IRTF)*. 2019. URL: https://doi.org/10.17487/RFC8452 (accessed: 15.05.2026).
13. Topcuoglu H., Hariri S., Wu M.-Y. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. *IEEE Transactions on Parallel and Distributed Systems*. 2002. Vol. 13, No. 3. P. 260–274. URL: https://doi.org/10.1109/71.993206 (accessed: 15.05.2026).
14. Faz-Hernández A., López J., de Oliveira A. K. SoK: A Performance Evaluation of Cryptographic Instruction Sets on Modern Architectures. *Proceedings of the 5th ACM on ASIA Public-Key Cryptography Workshop (APKC '18). ACM*. 2018. P. 9–18. URL: https://doi.org/10.1145/3197507.3197511 (accessed: 15.05.2026).
15. Bernstein D. J. Curve25519: New Diffie-Hellman Speed Records. *Public Key Cryptography – PKC 2006. Lecture Notes in Computer Science.* Vol. 3958. 2006. P. 207–228. URL: https://doi.org/10.1007/11745853_14 (accessed: 15.05.2026).
16. Gouvêa C. P. L., López J. Implementing GCM on ARMv8. *Topics in Cryptology – CT-RSA 2015. Lecture Notes in Computer Science.* Vol. 9048. 2015. P. 167–180. URL: https://doi.org/10.1007/978-3-319-16715-2_9 (accessed: 15.05.2026).
17. Pan W., Zheng F., Zhao Y., Zhu W.-T., Jing J. An Efficient Elliptic Curve Cryptography Signature Server With GPU Acceleration. IEEE Transactions on Information Forensics and Security. 2017. Vol. 12, No. 1. P. 111–122. URL: https://doi.org/10.1109/TIFS.2016.2603974 (accessed: 15.05.2026).

Історія статті:
Отримано: 04.04.2026 Доопрацьовано: 11.05.2026 Прийнято до друку: 23.05.2026 Опубліковано: 29.05.2026