

**ПРОБЛЕМИ ІНТЕГРАЦІЇ
ПОСТКВАНТОВИХ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ
У ПРОГРАМНІ СИСТЕМИ НА КЕРОВАНИХ ПЛАТФОРМАХ**

Божко І.К., Четвериков Г.Г.

e-mail: ivan.bozhko@nure.ua, grigorij.chetverykov@nure.ua

Науковий керівник – д.т.н., проф. Четвериков Г.Г.

Харківський національний університет радіоелектроніки, каф. ПІ
м. Харків, Україна

The migration to post-quantum cryptography (PQC) is becoming an important and urgent task for the software engineering industry following the NIST standardization of lattice-based algorithms ML-KEM and ML-DSA in 2024. While reference implementations of these algorithms are developed in C/C++ with low-level hardware optimizations, a significant portion of enterprise software operates on managed runtime platforms such as .NET and JVM. This paper presents a systematic analysis of the key challenges associated with integrating PQC algorithms into managed environments: performance issues caused by garbage collection and memory allocations, limited access to SIMD instructions, and increased sizes of cryptographic keys and ciphertexts. A classification of potential optimization approaches available in modern .NET is proposed, including stack allocation via `Span<T>`, hardware intrinsics, and memory pooling strategies. The results provide a structured foundation for further empirical research on PQC performance optimization in managed software ecosystems.

У 2024 році Національний інститут стандартів і технологій США (NIST) затвердив перші стандарти постквантової криптографії: FIPS 203 (ML-KEM, раніше CRYSTALS-Kyber) для механізму інкапсуляції ключів та FIPS 204 (ML-DSA, раніше CRYSTALS-Dilithium) для цифрового підпису. Ці алгоритми засновані на задачах теорії решіток, зокрема на задачі навчання з помилками (Learning With Errors, LWE), яка вважається стійкою як проти класичних, так і проти квантових атак [1]. Дослідження алгоритмів шифрування на основі решіток, зокрема NTRU-подібних схем, активно проводяться з кінця 2010-х років [2]. Стандартизація означає початок переходу: організації мають інтегрувати нові алгоритми у своє ПЗ до появи релевантних квантових комп'ютерів, оскільки існує загроза атак типу «збери зараз – розшифруй потім» (harvest now, decrypt later) [3].

Еталонні реалізації стандартизованих PQC-алгоритмів створено мовами C/C++ із застосуванням низькорівневих оптимізацій: SIMD-інструкцій (AVX2/AVX-512), ручного керування пам'яттю та стекових алокацій. Проте значна частина корпоративного ПЗ функціонує на керованих платформах, таких як .NET та JVM. Вони забезпечують продуктивність розробки та безпеку пам'яті, проте створюють специфічні обмеження для криптографічних обчислень.

На підставі аналізу архітектурних особливостей керованих платформ можна виділити три основні групи проблем інтеграції PQC-алгоритмів.

1. Проблеми продуктивності обчислень. Ключовою обчислювальною операцією решіткових алгоритмів є числове теоретичне перетворення (Number Theoretic Transform, NTT) – аналог швидкого перетворення Фур'є у скінченних полях. У нативних реалізаціях NTT оптимізується за допомогою AVX2-інструкцій, що дозволяє обробляти кілька коефіцієнтів паралельно. У керованому коді прямий доступ до SIMD-інструкцій став можливим лише з появою простору імен System.Runtime.Intrinsics у .NET Core 3.0, проте існуючі криптографічні бібліотеки (зокрема, BouncyCastle) поки що не використовують ці можливості повною мірою [4].

2. Проблеми керування пам'яттю. PQC-алгоритми оперують значними обсягами даних: розмір відкритого ключа ML-KEM-768 становить 1184 байти, а шифротексту – 1088 байтів, що значно перевищує відповідні параметри RSA-2048 та ECDSA. У керованих середовищах виділення та звільнення таких масивів відбувається на керованій купі (managed heap) з подальшим збиранням сміття (Garbage Collection, GC). При інтенсивних криптографічних операціях це може спричинити часті паузи GC, що є критичним для серверних застосунків, вимогливих до часу відповіді. Крім того, обнулення криптографічних буферів для запобігання витoku ключового матеріалу ускладнюється в керованому середовищі, де GC може переміщувати об'єкти в пам'яті.

3. Проблеми збільшених розмірів криптографічних параметрів. Порівняно з класичними алгоритмами, PQC-алгоритми мають значно більші розміри ключів та підписів. Наприклад, підпис ML-DSA-65 займає 3309 байтів проти 64 байтів ECDSA-P256. Це впливає на пропускну здатність мережевих протоколів (TLS handshake), обсяг серіалізованих даних та навантаження на системи зберігання сертифікатів. Для програмних систем на керованих платформах це означає додаткове навантаження на серіалізацію, буферизацію та мережевий ввід-вивід.

Платформа .NET надає низку механізмів, що дозволяють зменшити розрив у продуктивності. Їх класифікацію наведено у таблиці 1.

Таблиця 1 – Підходи до оптимізації PQC у середовищах .NET

Підхід	Механізм .NET	Очікуваний ефект
Stack Allocation	Span<T>, stackalloc	Зменшення навантаження на GC
Hardware Intrinsics	System.Runtime.Intrinsics (AVX2)	Прискорення NTT
Memory Pool	ArrayPool<T>	Повторне використання буферів
Native Integration	P/Invoke, NativeAOT	Виклик оптимізованого C-коду

Запропонована класифікація охоплює підходи різного рівня втручання: від використання стандартних засобів платформи (`Span<T>`, `ArrayPool<T>`) до глибокої інтеграції з нативним кодом через `P/Invoke` або компіляцію `NativeAOT`. Кожен підхід має свої компроміси між продуктивністю, портативністю коду та складністю підтримки. Зокрема, використання апаратних інтринсіків (`hardware intrinsics`) потенційно забезпечує найбільший приріст швидкодії для операції NTT, але обмежує портативність конкретними архітектурами процесорів. Інтеграція через `P/Invoke` дозволяє використати вже існуючі оптимізовані C-реалізації, але створює залежність від нативних бібліотек та ускладнює розгортання [5].

Проведений аналіз показує, що інтеграція постквантових криптоалгоритмів у ПЗ на керованих платформах тягне за собою комплекс специфічних проблем продуктивності, пов'язаних з особливостями керування пам'яттю, обмеженим доступом до апаратних оптимізацій та збільшеними розмірами криптографічних параметрів. Систематизовано основні підходи до оптимізації, доступні на платформі `.NET`. Подальші дослідження будуть спрямовані на емпіричну оцінку ефективності кожного з визначених підходів шляхом порівняльного бенчмаркінгу реалізацій `ML-KEM` та `ML-DSA` у `.NET` відносно еталонних C-реалізацій.

Список використаних джерел:

1. NIST. FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard. National Institute of Standards and Technology, 2024. 39 p. DOI: <https://doi.org/10.6028/NIST.FIPS.203>
2. Gorbenko, I., Kachko, O., Akolzina, O. Research of post-quantum public key encryption algorithm. International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo). 2018. DOI: 10.1109/UkrMiCo43733.2018.9047585
3. Mosca M. Cybersecurity in an era with quantum computers: will we be ready? IEEE Security & Privacy. 2018. Vol. 16, No. 5. P. 38–41. DOI: <https://doi.org/10.1109/MSP.2018.3761723>
4. The Legion of the Bouncy Castle. C# Cryptography APIs. URL: <https://www.bouncycastle.org/csharp/> (дата звернення: 01.03.2026).
5. Microsoft. System.Runtime.Intrinsics Namespace. .NET Documentation. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.runtime.intrinsics> (дата звернення: 01.03.2026).