ISSN 0485-8972

УДК 621.391:519.2 **DOI:10.30837/rt.2026.1.224.05**

Є.В. КОТУХ, канд. техн. наук

АРХІТЕКТУРНИЙ ПІДХІД ДО РЕАЛІЗАЦІЇ СТАНДАРТІВ ПОСТКВАНТОВОЇ КРИПТОГРАФІЇ ЗАСОБАМИ ФУНКЦІОНАЛЬНОЇ МОВИ ПРОГРАМУВАННЯ OCAML

Постановка проблеми

Стрімкий розвиток квантових обчислень створює екзистенційну загрозу для сучасних криптографічних систем із відкритим ключем. Алгоритм Шора дозволяє факторизувати цілі числа та обчислювати дискретні логарифми з використанням квантового обчислювача за поліноміальний час. Це призводить до зниження безпеки алгоритмів RSA, ECDSA та ECDH та дозволяє реалізувати атаки типу “збирай зараз, дешифруй пізніше” (harvest now, decrypt later). У відповідь на цю загрозу Національний інститут стандартів і технологій США (NIST) у серпні 2024 р. фіналізував три стандарти постквантової криптографії (PQC): FIPS 203 або ML-KEM, FIPS 204 або ML-DSA та FIPS 205 або SLH-DSA. Четвертий стандарт FIPS 206 або FN-DSA перебуває у стадії чернетки з очікуваною фіналізацією наприкінці 2026 р.

Водночас практична реалізація цих стандартів виконана переважно з використанням мов програмування C/C++, Rust та Go. Сучасні функціональні мови програмування де на рівні дизайну та архітектури перевага віддається надійності та безпеці створення криптографічних додатків, зокрема крпитографічних бібліотек постквантових алгоритмів залишаються поза увагою розробників PQC-бібліотек. Ця ситуація призводить до протиріччя – мова програмування, що використовується для доведення коректності реалізації PQC-алгоритмів, не має жодної реалізації даних алгоритмів. Отже, існує нагальна потреба в систематичному дослідженні архітектурних підходів до реалізації стандартів NIST PQC засобами функціональної мови програмування OCaml із забезпеченням стійкості до атак по побічних каналах та можливості формальної верифікації.

Аналіз останніх досліджень і публікацій

Стандартизація PQC-алгоритмів NIST ґрунтується на багаторічних дослідженнях, систематизованих у роботах [1, 2]. Підпис SPHINCS+ описано в роботі [3], алгоритм FALCON, заснований на NTRU-решітках, досліджується в роботі [4]. Формальна верифікація PQC-алгоритмів досягла значної зрілості. В роботах [5, 6] виконано доведення функціональної коректності ML-KEM у системі EasyCrypt та виконання вимог IND-CCA-безпеки. Для алгоритму Dilithium в роботі [7] було запропоновано аналогічне доведення безпеки в EasyCrypt, що виявило та виправило критичну прогалину в кількох опублікованих ROM/QROM-доведеннях для схеми FSwA. Щільне доведення безпеки SPHINCS+ також формально верифіковане в EasyCrypt та представлено в роботі [8]. В дослідженні [9] проєкт libcrux від компанії Cryspen формально верифікував реалізацію ML-KEM мовою Rust за допомогою тулчейну hax. Ця верифікація виявила вразливість KyberSlash, що дає витік часових характеристик через згенероване компілятором ділення на модуль q, що вразило численні реалізації [10]. Проєкт Formosa Crypto / libjade містить Jasmin-реалізації ML-KEM із верифікованим x86_64-асемблером [11].

Мова програмування OCaml (Objective Caml) виникла у середині 1990-х років як розвиток діалектів ML у дослідницькому середовищі INRIA та поєднала строготипізовану функціональну парадигму з ефективною компіляцією у нативний код. Завдяки потужній системі статичної типізації з автоматичним виведенням типів, параметризованим модулям і підтримці незмінних структур даних, OCaml забезпечує високий рівень формальної коректності програм і зменшує ризик типових помилок пам’яті, що є критично важливим у криптографічних застосуваннях. Функціональна семантика, прозорість побічних ефектів і можливість модульної абстракції математичних структур (кілець, полів, решіток) роблять мову зручною для

реалізації складних криптографічних примітивів та протоколів. У контексті постквантової криптографії, де алгоритми базуються на складних алгебраїчних конструкціях (SIS, LWE, модульні решітки) та вимагають строгого контролю над інваріантами й нормами, використання OCaml є обґрунтованим через її здатність забезпечувати формальну строгість, безпечну абстракцію та передбачувану поведінку коду. Реалізація постквантових алгоритмів на OCaml може поєднати дослідницьку гнучкість із високим рівнем надійності, що особливо актуально для розробки та верифікації сучасних криптографічних стандартів. Щодо інструментів формальної верифікації для OCaml, проєкт coq-of-ocaml від Formal Land продемонстрував здатність трансляції OCaml-коду в Coq/Rocq для формального доведення, обробивши понад 100 000 рядків OCaml-коду блокчейну Tezos із приблизно 60 000 рядків Rocq-доведень [12]. Проєкт fiat-crypto генерує Coq-верифікований C-код для арифметики полів і вже використовується бібліотекою mirage-crypto-ec для еліптичних кривих [13].

Водночас систематичний пошук на GitHub (понад 518 PQC-репозиторіїв) у реєстрі opam та на discuss.ocaml.org не виявив жодної реалізації PQC-алгоритмів мовою OCaml. Бібліотека liboqs надає C-реалізації всіх стандартизованих алгоритмів із офіційними біндінгами для C++, Go, Java, Python, Rust та .NET, проте біндінги для OCaml відсутні [14]. Таким чином, невирішеною залишається проблема системного підходу до реалізації PQC-стандартів NIST у функціональній парадигмі з урахуванням вимог постійного часу виконання та формальної верифікації.

Метою статті є розроблення архітектурних засад та обґрунтування стратегії реалізації фіналізованих стандартів постквантової криптографії FIPS 203, FIPS 204, FIPS 205 засобами функціональної мови програмування OCaml із забезпеченням стійкості до атак по побічних каналах та інтеграцією механізмів формальної верифікації.

Для досягнення мети поставлено наступні завдання:

- провести аналіз математичних вимог стандартизованих PQC-алгоритмів та їх відповідності наявним засобам екосистеми OCaml;
- обґрунтувати архітектурну модель гібридної реалізації OCaml+C, що забезпечує виконання криптографічних операцій із постійним часом;
- розробити модульну структуру бібліотеки з використанням параметризації;
- визначити стратегію формальної верифікації через наявні OCaml-орієнтовані інструменти;
- сформувати рекомендації, щодо поетапної реалізації запропонованого підходу.

Виклад основного матеріалу дослідження

Проблема набуває особливої актуальності в контексті екосистеми MirageOS – операційної системи-унікернелу, повністю написаної на OCaml, що використовується для побудови безпечної мережевої інфраструктури. Відсутність PQC-підтримки блокує перехід цієї екосистеми на квантовостійку криптографію. Стандарт FIPS 203 визначає механізм інкапсуляції ключів на основі задачі Module-LWE над кільцем поліномів $Zq[X]/(X^{256}+1)$ з модулем $q = 3329$. Ключовою обчислювальною операцією є теоретико-числове перетворення (NTT) з примітивним коренем 256-го ступеня з одиниці $\zeta = 17 \bmod 3329$. Оскільки $3329 = 13 \cdot 256 + 1$, примітивні корені 256-го ступеня існують. А корені 512-го ступеня не існують, що визначає специфіку NTT, що полягає в тому, що поліноми ступеня 255 відображаються у 128 пар поліномів ступеня 1. Три набори параметрів ML-KEM-512, ML-KEM-768 та ML-KEM-1024 відповідають рівням безпеки NIST 1, 3 та 5. Порівняльний аналіз проведено в табл. 1.

Таблиця 1

Порівняльні характеристики стандартизованих PQC-алгоритмів

Стандарт	Структура	q	PK / SK, Б	Підпис/CT, Б	Складність
ML-KEM-768	$Zq[X]/(X^{256}+1)$	3 329	1 184 / 2 400	1 088	Середня
ML-DSA-65	$Zq[X]/(X^{256}+1)$	8 380 417	1 952 / 4 032	3 309	Середньо-висока
SLH-DSA-128f	Хеш-дерева	—	32 / 64	17 088	Низька
FN-DSA-512	NTRU-решітка	12 289	897 / 1 281	666	Дуже висока

Стандарт FIPS 204 використовує ту саму кільцеву структуру з модулем q = 8 380 417 та фреймворк FSwA. Процес підписання включає відбір із відхиленням, що потребує в середньому 4–7 ітерацій, та генерацію поліномів-викликів через SHAKE-256. Усі операції є цілочисельною арифметикою без використання чисел із плаваючою комою.

Стандарт FIPS 205 принципово відрізняється від решітчастих схем. Це підпис, оснований на хеш-функції. Реалізація використовує гіпердерева з дерев Меркла, одноразові підписи WOTS+ та кількаразові підписи FORS. Безпека ґрунтується виключно на властивостях хеш-функцій. Дванадцять наборів параметрів охоплюють варіанти SHA-2 та SHAKE на трьох рівнях безпеки у швидкій та компактній конфігураціях.

Чернетка FIPS 206 описує підпис на NTRU-решітках (q = 12 289), що генерує найменші PQC-підписи (~666 байт для 512 біт), проте вимагає дискретного гаусівського семплінгу на основі FFT із числами подвійної точності з плаваючою комою. Пакет CNSA 2.0 Агентства національної безпеки США не затверджує FN-DSA через високу схильність до помилок реалізації. У березні 2025 р. NIST також обрав HQC як резервний KEM на основі кодів.

Незважаючи на повну відсутність PQC-реалізацій, екосистема OCaml містить низку бібліотек, що формують математичну основу для їх побудови. Бібліотеки ff та polynomial надають арифметику скінченних полів через функтори та FFT-множення поліномів за алгоритмом Кулі–Тьюкі [15]. Створення поля коефіцієнтів Kyber реалізовано через функтор ff бібліотеки MakeFp. Однак ці бібліотеки потребують адаптації для арифметики кільця факторів та використовують бібліотеку довгої арифметики Zarith, що є надмірним для малих фіксованих модулів.

Бібліотека mirage-crypto демонструє перевірену гібридну архітектуру OCaml+C. C-заглушки використовуються для чутливих до часу примітивів (AES з AES-NI. Арифметику еліптичних кривих реалізовано в бібліотеці fiat-crypto. Запропоновано використання OCaml для протокольної логіки та типобезпечного API [16]. Модуль еліптичних кривих mirage-crypto-ec вже використовує Coq-верифікований C-код з бібліотекою fiat-crypto. Такий архітектурний підхід пропонується для розробки криптопримітивів PQC.

Бібліотека hacl-star надає формально верифіковані криптопримітиви (SHA-3/SHAKE, SHA-2, Curve25519, Ed25519), написані на F* та скомпільовані до C через KaRaMeL, з автогенерованими OCaml ctypes-біндінгами [17]. Бібліотека digestif забезпечує хеш-реалізації SHA-2 та SHA-3. Бібліотека eqaf реалізує порівняння з постійним часом виконання. Аналіз показує, що бібліотека liboqs є найповнішою C-бібліотекою PQC із офіційними біндінгами для шести мов, але OCaml серед них відсутня [14].

Критичним архітектурним обмеженням є неможливість гарантувати постійний час виконання в чистому OCaml. Це було емпірично підтверджено проєктом eqaf, де чиста OCaml-функція порівняння з постійним часом провалила тести синхронізації, що змусило перейти на C-заглушки [18]. Перешкоди є фундаментальними. Досліджено, що збирач сміття (garbage collector) створює непередбачувану варіативність часу, а цілочисельні типи (int32, int64, nativeint) алокуються в купі на кожну операцію, створюючи зайве навантаження. Оптимізації компілятора Flambda можуть перетворити бітові операції з постійним часом на код із розгалуженнями.

На підставі проведеного аналізу вимог до продуктивності, безпеки та типобезпечності запропоновано багаторівневу гібридну архітектуру реалізації постквантових криптографічних алгоритмів у середовищі OCaml. Архітектура ґрунтується на принципі чіткого розмежування високорівневої протокольної логіки та низькорівневих арифметичних примітивів з метою мінімізації довіреної обчислювальної бази та забезпечення властивостей виконання з постійним часом.

Рівень 1 – Типобезпечний прикладний рівень (OCaml API).

Верхній рівень реалізує абстрактні інтерфейси криптографічних примітивів (KEM, цифровий підпис) у вигляді сигнатур модулів та функторів для параметричної інстанціації алгоритмів. На цьому рівні забезпечуються інкапсуляція типів відкритих і секретних ключів,

контроль інваріантів параметрів, серіалізація та десеріалізація, управління життєвим циклом ключів, інтеграція з існуючими криптографічними бібліотеками OCaml (наприклад, mirage-crypto).

Реалізація цього рівня мінімізує можливість логічних помилок завдяки статичній системі типів OCaml.

Рівень 2 – **Алгебраїчний та протокольний рівень.**

Середній рівень реалізує специфічні для постквантових схем математичні конструкції, такі як операції над поліномами та елементами кільця факторів, перетворення типу NTT, механізми кодування та стиснення, допоміжні криптографічні структури (наприклад, дерева хешів).

Цей рівень забезпечує формалізоване відображення математичної моделі алгоритму на програмну реалізацію, зберігаючи при цьому контроль над представленням даних та інваріантами.

Рівень 3 – **Низькорівневі примітиви з постійним часом виконання.**

Найнижчий рівень містить оптимізовані реалізації арифметичних та криптографічних примітивів мовою C із гарантіями виконання за визначений час, а саме, модульна арифметика з редукцією, NTT-перетворення, вибірки розподілів, криптографічні хеш-функції (SHA-3, SHAKE) з використанням перевірених реалізацій, зокрема HACL*.

Взаємодія з OCaml здійснюється через інтерфейс FFI з мінімізацією алокацій та копіювання пам’яті, що дозволяє уникнути неконтрольованого впливу збирача сміття на обробку секретних даних.

Для представлення коефіцієнтів поліномів доцільним є використання Bigarray.Array1.t з елементами int32. Така структура розміщує буфер поза керованою купою OCaml, забезпечує стабільні C-сумісні покажчики для FFI та не підлягає скануванню збирачем сміття. Це мінімізує ризик неконтрольованого копіювання секретних даних та зменшує накладні витрати взаємодії з GC. Порівняно з Bytes.t Bigarray забезпечує гарантований непереміщуваний layout та спрощує інтеграцію з низькорівневими constant-time примітивами. За потреби буфер може бути явно обнулений через explicit_bzero.

Нативний тип int у OCaml (63-бітний на 64-бітних платформах, unboxed) є достатнім для модульної арифметики з параметрами $q = 3329$ (ML-KEM) та $q = 8\,380\,417$ (ML-DSA) за умови контролю переповнення проміжних значень. Практичний досвід, зокрема вразливість KyberSlash, демонструє небезпеку використання апаратної операції ділення, що може спричиняти таймінгові витоки. Застосування редукцій Барретта або Монтгомері в низькорівневих constant-time C-примітивах дозволяє усунути витоки, пов’язані з операціями ділення, за умови дотримання принципу виконання з постійним часом.

Система модулів OCaml із підтримкою сигнатур та функторів забезпечує природний механізм параметризації постквантових криптографічних алгоритмів. Для реалізації ML-KEM доцільно визначити сигнатуру модуля KEM_PARAMS, що інкапсулює параметри безпеки, зокрема ранг модуля $k \in \{2,3,4\}$, параметри розподілів η_1, η_2, коефіцієнти стиснення d_u, d_v, а також глобальні константи qта n. Функтор Make_ML_KEM може приймати модуль цієї сигнатури та генерувати повну реалізацію алгоритму для відповідного рівня безпеки ML-KEM.

Аналогічний підхід застосовується до ML-DSA, де параметри визначаються алгебраїчною структурою решітки та граничними нормами. Для SLH-DSA функторизація можлива шляхом параметризації висот дерев, довжин хеш-функцій та FORS-параметрів.

Запропонований підхід узгоджується з усталеною практикою екосистеми OCaml. Зокрема, бібліотеки bls12-381 та mirage-crypto-ec використовують сигнатури модулів як формальні інтерфейси з подальшою інстанціацією через функтори для конкретних параметрів. У бібліотеці ff визначено сигнатури PRIME_FIELD та POLYNOMIAL_RING, які реалізуються через функтори для кожного набору параметрів поля. Модульна структура проєкту може

включати такі компоненти: pqc-ring – арифметика кільця поліномів, скінченні поля, NTT, pqc-kem – інтерфейси та реалізації KEM, pqc-sig – інтерфейси та реалізації цифрових підписів, pqc-mirage – інтеграція з екосистемою MirageOS, pqc-ct – низькорівневі constant-time примітиви, pqc-test – тестування та перевірка відповідності KAT.

Такий підхід забезпечує чітке розмежування абстрактних інтерфейсів і конкретних реалізацій, спрощує підтримку різних рівнів безпеки та зменшує ризик логічних помилок завдяки статичній системі типів OCaml.

Унікальною перевагою OCaml є те, що провідні інструменти верифікації PQC самі побудовані на цією мовою. EasyCrypt, що використовує OCaml, уже має машинно-перевірені доведення для ML-KEM [5, 6], ML-DSA [7] та SLH-DSA [8]. Компілятор Jasmin, що реалізовано на OCaml та Coq, генерує верифікований x86_64-асемблер для проєкту libjade [11]. Беручи до уваги поточний стан пропонується три паралельні шляхи верифікації. По-перше, це використання конвеєра EasyCrypt та Jasmin, що працює як верифікований асемблер, є найбільш зрілим рішенням та вже має реалізації промислового використання. По-друге, це використання конвеєра Coq/Rocq, що дозволяє написати PQC на OCaml, транслювати в Coq та довести властивості. Такий підхід є новаторським і ніким не застосовувався до PQC. По-третє, це використання розширення бібліотеки fiat-crypto для генерації верифікованого C-коду NTT та арифметики кільця поліномів – це розвиток підходу, вже перевіреного для еліптичних кривих у mirage-crypto-ec [13].

Для забезпечення коректності реалізації інтегровано KAT-вектори NIST, згенеровані на основі еталонних реалізацій pq-crystals, що гарантує бітову відповідність специфікації. Додатково застосовано property-based тестування (QCheck) для перевірки алгебраїчних інваріантів, зокрема коректності KEM-декапсуляції, оборотності NTT/INTT та аксіом кільця $\mathbb{Z}_q[x]/(x^n+1)$.

Для аналізу побічних каналів інтегровано dudect та інструментальні засоби Valgrind і TIMECOP, що дозволяють статистично та інструментально виявляти потенційні таймінгові витоки.

SLH-DSA є найприроднішою відправною точкою для реалізації на OCaml, оскільки не вимагає арифметики кільця поліномів, обмежуючись лише обчисленням хеш-функцій та операції з деревами Меркла. Алгебраїчні типи даних OCaml дозволяють елегантно виразити деревоподібні структури. Ланцюги WOTS+ виражені як рекурсивні застосування хешів, FORS як k незалежних дерев Меркла, а гіпердерево як d шарів XMSS. Потоки керування залежать від публічних даних, тому проблеми постійного часу обмежуються реалізацією хеш-функції.

Для ML-KEM основним викликом є реалізація NTT, яка має бути на C для забезпечення постійного часу. Нативний 63-бітний int OCaml достатній для арифметики q = 3329, але редукція Барретта має уникати згенерованого компілятором ділення.

CBD-семплінг із виходу SHAKE в ML-KEM реалізується прямолінійно без секретно-залежного rejection sampling, що спрощує забезпечення constant-time властивостей у процесі інкапсуляції.

Натомість у ML-DSA основним джерелом складності є цикл rejection sampling під час підписання, кількість ітерацій якого потенційно корелює з секретними значеннями та може призводити до таймінгових витоків. Модуль $q = 8\,380\,417$ безпечно вміщується в 63-бітний нативний int Ocaml, а добутки двох коефіцієнтів також не спричиняють переповнення. Однак у внутрішніх циклах множення поліномів та NTT необхідна своєчасна редукція (Барретта або Монтгомері) для запобігання переповненню при накопиченні сум. Додатковими складними компонентами ML-DSA є обчислення вектора підказок та генерація challenge-поліномів із контрольованою вагою, що вимагає строгої, визначеної у часі, реалізації.

Для FN-DSA дискретний гаусівський семплінг потребує високоточної арифметики з плаваючою комою, що ускладнює забезпечення строгих, визначених у часі, гарантій у ран-

таймі OCaml. Апаратна плаваюча арифметика не надає детермінованого контролю над часовими характеристиками та режимами округлення, що є критичним для безпечної реалізації.

Генерація ключів вимагає розв'язання рівняння NTRU та обчислення поліноміального НСД високої точності, що додатково підвищує вимоги до арифметичної стабільності.

З огляду на це, доцільним є використання FFI-обгортки навколо перевіреної C-реалізації, яка застосовує цілочисельну емуляцію плаваючої арифметики та орієнтована на визначене у часі виконання, замість спроб нативної реалізації критичних компонентів у OCaml.

Рекомендації щодо реалізації PQC алгоритмів з використанням OCaml

Запропонована стратегія реалізації постквантових криптографічних алгоритмів у середовищі OCaml базується на поетапному впровадженні компонентів різного рівня складності з поступовим переходом від інтеграції перевірених зовнішніх реалізацій до створення нативних функціональних модулів із перспективою формальної верифікації. На першому етапі (місяці 1–2) передбачається створення OCaml-біндінгів до бібліотеки liboqs, що підтримується ініціативою Open Quantum Safe. Архітектурно рекомендується орієнтуватися на модель, подібну до HACL*, де криптографічно критична логіка залишається в C-реалізації, а OCaml використовується як типобезпечний високорівневий інтерфейс. Доцільно застосовувати традиційні C-заглушки замість ctypes-foreign через libffi, оскільки останній може додавати приблизно 150 наносекунд накладних витрат на кожен виклик. Результатом фази має стати перший PQC-пакет для OCaml, опублікований у менеджері пакетів opam.

Друга фаза (місяці 2–4) присвячена чистій функціональній реалізації hash-based підпису SLH-DSA. Найбільш природним для функціональної парадигми OCaml є алгоритм SLH-DSA-SHAKE-128f, оскільки він базується на хеш-функціях і не потребує складної кільцевої арифметики або rejection sampling, залежного від секретних значень. Верифікована реалізація SHAKE з бібліотеки HACL* може бути інтегрована через OCaml-біндінги. На початковому етапі розробки обов'язковим є тестування проти KAT-векторів National Institute of Standards and Technology, сформованих на основі еталонних реалізацій проєкту pq-crystals.

Третя фаза (місяці 4–7) передбачає реалізацію ML-KEM, що має стратегічне значення для сучасних протоколів, включно з TLS 1.3. Запропоновано побудову модуля арифметики кільця поліномів виду $\mathbb{Z}_q[X]/(X^{256}+1)$ з використанням Number Theoretic Transform (NTT). Оптимальним є використання еталонного C-коду з проєкту pq-crystals, при цьому NTT-метелики та процедури редукції Барретта або Монтгомері доцільно винести у низькорівневі C-заглушки для запобігання переповненню та збереження властивостей constant-time виконання.

Четверта фаза (місяці 7–10) присвячена реалізації ML-DSA. Для цього необхідно розширити арифметичну інфраструктуру для підтримки модуля $q = 8\,380\,417$, що вміщується у 63-бітний нативний тип int OCaml на 64-бітних платформах. Особлива увага повинна приділятися контролю накопичення сум під час поліноміального множення та NTT-обчислень шляхом своєчасного застосування модульної редукції. Реалізація генерації challenge-поліномів, rejection sampling у циклі підписання та обчислення hint vector повинна виконуватися з урахуванням вимог constant-time семантики та мінімізації можливих таймінгових витоків. Для виявлення потенційних побічних каналів рекомендується інтеграція інструментів статистичного аналізу, зокрема dudect, а також інструментів динамічного аналізу пам'яті Valgrind та TIMECOP.

П'ята фаза (місяці 10+) орієнтована на реалізацію FN-DSA та формальну верифікацію компонентів. Дискретний гаусівський семплінг із використанням апаратної плаваючої коми у середовищі OCaml є практично небажаним з точки зору криптографічної безпеки, оскільки ускладнює гарантування constant-time поведінки та контроль режимів округлення у рантаймі. Рекомендується використовувати FFI-обгортку навколо перевіреної C-реалізації, зокрема роботи криптографа Thomas Pornin, яка використовує цілочисельну емуляцію плаваючої арифметики та оптимізована для constant-time виконання. Генерація ключів FN-DSA потре-

бує розв'язання рівняння NTRU та обчислення поліноміального НСД високої точності, тому OCaml доцільно використовувати як orchestration-рівень.

Додатково пропонується застосування coq-of-ocaml для трансляції чистих OCaml-компонентів у середовище Coq з метою доведення функціональної коректності криптографічних модулів. Перспективним напрямом дослідження є розширення Fiat-Crypto для автоматичної генерації верифікованого NTT-коду, параметризованої арифметики кільця $\mathbb{Z}_q[X]/(X^n + 1)$та подальшої інтеграції з OCaml-екосистемою.

Запропонована дорожня карта поєднує швидке практичне розгортання PQC через біндінги liboqs, створення нативних функціональних реалізацій SLH-DSA, ML-KEM та ML-DSA, а також дослідницький розвиток формально верифікованих компонентів FN-DSA. Такий багаторівневий підхід мінімізує криптографічні ризики, забезпечує поступове масштабування складності реалізації та створює основу для формування повноцінної постквантової криптографічної екосистеми в середовищі OCaml.

Висновки та перспективи подальших досліджень

Вперше запропоновано комплексний архітектурний підхід до реалізації фіналізованих стандартів постквантової криптографії NIST засобами функціональної мови програмування OCaml. Основні результати полягають у наступному.

Встановлено повну відсутність PQC-реалізацій в екосистемі OCaml на підставі систематичного аналізу понад 518 PQC-репозиторіїв на GitHub, реєстру opam та форуму discuss.ocaml.org. Водночас ідентифіковано математичний фундамент, а саме бібліотеки ff, polynomial, Zarith та перевірені архітектурні патерни mirage-crypto, hacl-star, що забезпечують реалістичність побудови PQC-бібліотеки.

Обґрунтовано трирівневу гібридну архітектуру OCaml+C, що поєднує типобезпечний функторний API OCaml із C-примітивами з постійним часом виконання. Показано, що Bigarray.Array1.t є оптимальним представленням даних для коефіцієнтів поліномів, забезпечуючи розміщення поза купою OCaml, стабільні C-покажчики та можливість явного обнулення секретних даних.

Визначено три паралельні шляхи формальної верифікації. Конвеєр coq-of-ocaml є найбільш перспективним з наукової точки зору, оскільки дозволяє верифікувати OCaml-код безпосередньо, без трансляції в іншу мову.

Розроблено рекомендації щодо реалізації. Найбільш впливовими короткостроковими внесками визначено: пакет ocaml-liboqs, що є першим OCaml PQC-пакет, чисту реалізацію SLH-DSA та ML-KEM.

Перспективи подальших досліджень включають, але не обмежуються, практичну побудову бібліотеки за запропонованою архітектурою. Цікавими напрямами є емпіричне порівняння продуктивності OCaml-реалізацій із еталонними C-реалізаціями та дослідження можливостей неборксованих типів OxCaml (int32#, int64#) від Jane Street для NTT-метеликів із постійним часом на чистому OCaml. Інтеграція PQC-бібліотеки з ocaml-tls для гібридного обміну ключами X25519 + ML-KEM-768 дозволила б продемонструвати використання запропонованого архітектурного підходу до практичної реалізації криптографічного протоколу. Застосування запропонованого архітектурного підходу до криптографії на неабелевих групах, зокрема логарифмічних підписів та криптосистем MST, створило б предпосилки до використання артефактів групової криптографії в побудові криптографічних застосунків нового покоління.

Список літератури:

1. Avanzi R., Bos J., Ducas L. et al. CRYSTALS-Kyber: Algorithm Specifications and Supporting Documentation (version 3.02). NIST PQC Submission. 2021. URL: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf

2. Ducas L., Kiltz E., Lepoint T. et al. CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation (version 3.1). NIST PQC Submission. 2021. URL: https://pq-crystals.org/dilithium/

3. Bernstein D. J., Hülsing A., Kölbl S. et al. The SPHINCS+ Signature Framework. Advances in Cryptology – CRYPTO 2019. Springer, 2019. P. 390–420. DOI: 10.1007/978-3-030-26948-7_16
4. Fouque P.-A., Hoffstein J., Kirchner P. et al. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. NIST PQC Submission. 2020. URL: https://falcon-sign.info/
5. Almeida J. B., Barbosa M., Barthe G. et al. A Machine-Checked Proof of the Correctness of Kyber // IACR Transactions on Cryptographic Hardware and Embedded Systems. 2023. Vol. 2023, No. 4. P. 94–117.
6. Almeida J. B., Barbosa M., Barthe G. et al. Formally Verifying Kyber: IND-CCA Security in EasyCrypt. Advances in Cryptology – CRYPTO 2024. Springer, 2024.
7. Barbosa M., Barthe G., Grégoire B. et al. Machine-Checked Security Proofs of Dilithium in EasyCrypt. Advances in Cryptology – CRYPTO 2023. Springer, 2023.
8. Barbosa M., Barthe G., Hülsing A. et al. A Tight Security Proof for SPHINCS+ in EasyCrypt. Advances in Cryptology – ASIACRYPT 2024. Springer, 2024.
9. Polubelova M., Protzenko J., Ho S. et al. libcrux: A Formally Verified, High-Assurance Cryptographic Provider. Cryspen Technical Report. 2024. URL: https://cryspen.com/libcrux/
10. Janssen D., Breitner J., Bhargavan K. KyberSlash: Timing Attacks on Kyber Implementations. 2024. URL: https://kyberslash.cr.yp.to/
11. Almeida J. B., Baritel-Ruet C., Barbosa M. et al. Machine-Checked Proofs for Cryptographic Standards: Indifferentiability of SPONGE and Secure High-Assurance Implementations of SHA-3 // Proceedings of the 2019 ACM SIGSAC. P. 1607–1622. DOI: 10.1145/3319535.3363211
12. Claret G., Zykin M. Formal Verification of OCaml Programs with coq-of-ocaml. Formal Land Technical Report. 2023. URL: https://formal.land/
13. Erbsen A., Philipoom J., Gross J. et al. Simple High-Level Code for Cryptographic Arithmetic – With Proofs, Without Compromises // IEEE Symposium on Security and Privacy. 2019. P. 1202–1219. DOI: 10.1109/SP.2019.00005
14. Stebila D., Mosca M. Post-Quantum Key Exchange for the Internet and the Open Quantum Safe Project. Selected Areas in Cryptography – SAC 2016. Springer, 2017. P. 14–37. DOI: 10.1007/978-3-319-69453-5_2
15. Willems D. ff: A Library for Finite Field Arithmetic in OCaml. opam package. 2022. URL: https://gitlab.com/nomadic-labs/cryptography/ocaml-ff
16. Lucas D. et al. mirage-crypto: Cryptographic Primitives for MirageOS. opam package. 2024. URL: https://github.com/mirage/mirage-crypto
17. Zinzindohoué J.-K., Bhargavan K., Protzenko J. et al. HACL*: A Verified Modern Cryptographic Library // Proceedings of the 2017 ACM SIGSAC. P. 1789–1806. DOI: 10.1145/3133956.3134043
18. Merigoux D. Constant-Time Operations in OCaml: Lessons from eqaf. OCaml Workshop 2020.
19. Khalimov, G., Kotukh, Y., Kolisnyk, M., & Khalimova, S., Sievierinov, O. (2024). LINE: Cryptosystem based on linear equations for logarithmic signatures // Cryptology ePrint Archive: Report 2024/697. https://ia.cr/2024/697
20. Khalimov G., Kotukh Y., Kolisnyk M., Khalimova S., Sievierinov O., & Korobchynskyi, M. Digital signature scheme based on linear equations // K. Arai (Ed.). Advances in Information and Communication. FICC 2025. Lecture Notes in Networks and Systems. 2025. Vol. 1285. Springer. https://doi.org/10.1007/978-3-031-84460-7_46
21. Khalimov G., Kotukh Y., Kolisnyk M., Khalimova S., Sievierinov O., & Volkov O. SIGNLINE: Digital signature scheme based on linear equations cryptosystem // 2024 4th International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME) (p. 1–9). IEEE. https://doi.org/10.1109/ICECCME62383.2024.10796704
22. Kotukh Y., Severinov E., Vlasov O., Tenytska A., & Zarudna E. Some results of development of cryptographic transformations schemes using non-abelian groups // Radiotekhnika. 2021. No 204. P.66–72.
23. Kotukh Y., & Khalimov G. Hard problems for non-abelian group cryptography // Fifth International Scientific and Technical Conference “Computer and Information Systems and Technologies”. 2021. https://doi.org/10.30837/csitic52021232176
24. Kotukh Y., Khalimov G., Dzhura I., & Hivrenko H. Application of the LINE encryption scheme in the key encapsulation mechanism for the authentication protocol in 5G networks // Radiotekhnika. 2025. No 219. P. 36–45. https://doi.org/10.30837/rt.2024.4.219.04
25. Kotukh Y., Khalimov G., Korobchynskyi M., Rudenko M., Liubchak V., Matsyuk S., & Chashchyn M. Research horizons in group cryptography in the context of post-quantum cryptosystems development // Radiotekhnika. 2024. No 216. P.62–72. https://doi.org/10.30837/rt.2024.1.216.05
26. Kotukh Y., & Khalimov G. Towards practical cryptoanalysis of systems based on word problems and logarithmic signatures // Information security: Problems and prospects. 2022. P. 55–60.
27. Khalimov G., & Kotukh Y. Cryptographic strengthening of MST3 cryptosystem via automorphism group of Suzuki function fields // arXiv preprint arXiv:2504.07318. 2025.https://arxiv.org/abs/2504.07318
28. Khalimov G., & Kotukh Y. MST3 encryption improvement with three-parameter group of Hermitian function field // arXiv preprint arXiv:2504.15391. 2025. https://arxiv.org/abs/2504.15391
29. Khalimov G., & Kotukh Y. Advanced MST3 encryption scheme based on generalized Suzuki 2-groups // arXiv preprint arXiv:2504.11804. 2025. https://arxiv.org/abs/2504.11804

30. Khalimov G., & Kotukh Y. Improved MST3 encryption scheme based on small Ree groups // arXiv preprint arXiv:2504.10947. 2025. https://arxiv.org/abs/2504.10947

31. Khalimov G., Kotukh Y., & Khalimova S. Encryption scheme based on the automorphism group of the Ree function field // IEEE 7th International Conference on Internet of Things: Systems, Management and Security (IOTSMS).2020. P. 1–8.

32. Khalimov G., Didmanidze I., Sievierinov O., Kotukh Y., & Shonia O. Encryption scheme based on the automorphism group of the Suzuki function field // IEEE International Conference on Problems of Infocommunications, Science and Technology (PIC S&T 2020). P. 383–387.

33. Khalimov G., Kotukh Y., & Khalimova S. Improved encryption scheme based on the automorphism group of the Ree function field // IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS). 2021.

34. Khalimov G., Kotukh Y., & Khalimova S. (2019). MST3 cryptosystem based on the automorphism group of the Hermitian function field // IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T 2019). P. 865–868.

35. Khalimov G., Kotukh Y., Didmanidze I., Sievierinov O., Khalimova S., & Vlasov A.. Towards three-parameter group encryption scheme for MST3 cryptosystem improvement // IEEE 5th World Conference on Smart Trends in Systems Security and Sustainability (WorldS4). 2021. P. 204–211.

36. Khalimov G., Kotukh Y., Didmanidze I., & Khalimova S. Encryption scheme based on small Ree groups // Proceedings of the 2021 7th International Conference on Computer Technology Applications (ICCTA ’21). 2021. P. 33–37.

Надійшла до редколегії 04.01.2026
Прийнята до друку після рецензування 23.04.2026
Публікація (оприлюднення) 30.04.2026

Відомості про автора:
Котух Євген Володимирович – канд. техн. наук, доцент, професор кафедри кібербезпеки; Національний технічний університет «Дніпровська політехніка»; Дніпро, Україна; e-mail: yevgenkotukh@gmail.com; ORCID: https://orcid.org/0000-0003-4997-620X