Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

В.о. зав. кафедрою КІСМіТ
Марина ЄСІНА
"Допущено до захисту"

« »_____2025 р.

Пояснювальна записка
до кваліфікаційної роботи бакалавра
на тему: «Дослідження та порівняльний аналіз методів і механізмів забезпечення захисту та стійкості алгоритмів постквантових електронних підписів»

оцінка «_____»

Голова ЕК
Мичуда Л. З.

Керівник: к.т.н., доцент Єсіна М. В.

Рецензент: к.т.н., професор Качко О. Г.

Виконавець: студент групи КБ-41
Барбашин І.В.

Харків 2025

РЕФЕРАТ

Кваліфікаційна робота бакалавра містить: 54 сторінок, 10 рисунків, 3 таблиці, 2 додатки, 29 джерел.

Мета роботи полягає в проведенні теоретичного та практичного порівняльного аналізу трьох багатовимірних схем електронного підпису – UOV, MAYO та SNOVA.

Об’єкт дослідження – методи й механізми забезпечення захисту та стійкості алгоритмів постквантових електронних підписів.

Предмет дослідження – схеми електронного підпису UOV, MAYO та SNOVA з огляду на їхню стійкість до квантових атак, розмір відкритого й секретного ключів та продуктивність основних криптографічних операцій.

Методи дослідження об’єднують теоретичний аналіз складності та безпеки з практичним тестуванням бенчмарками у реальних умовах.

В межах дослідження виконано всебічний аналіз багатовимірних постквантових схем електронного підпису UOV, MAYO та SNOVA: порівняно їх стійкість до квантових атак, визначено розміри відкритого й секретного ключів та виміряно швидкодію основних криптографічних операцій.

Результати дають змогу розробникам і стандартизаторам обрати оптимальну схему для різних класів застосувань: від IoT до фінансових транзакцій. Показано компроміс між розміром ключа, швидкодією та рівнем безпеки. На основі цих даних можна сформувати загальні керівні принципи вибору параметрів багатовимірних підписів, адаптуючи їх під конкретні обмеження платформи та рівень загроз.

Ключові слова: КІБЕРБЕЗПЕКА, ПОСТКВАНТОВА КРИПТОГРАФІЯ, ЕЛЕКТРОННИЙ ПІДПИС, БАГАТОВИМІРНІ СХЕМИ, ЗАХИСТ АЛГОРИТМІВ, СТІЙКІСТЬ ДО КВАНТОВИХ АТАК, UOV, MAYO, SNOVA, ОЦІНКА ЕФЕКТИВНОСТІ, ПОРІВНЯЛЬНИЙ АНАЛІЗ, БЕНЧМАРКИ.

ABSTRACT

The bachelor's thesis contains: 54 pages, 10 figures, 3 tables, 2 appendices, 29 sources.

The purpose of the work is to conduct a theoretical and practical comparative analysis of three multidimensional electronic signature schemes – UOV, MAYO and SNOVA.

Object of research – methods and mechanisms for ensuring the protection and stability of post-quantum electronic signature algorithms.

The subject of the study is the electronic signature schemes UOV, MAYO and SNOVA in terms of their resistance to quantum attacks, the size of the public and secret keys and the performance of basic cryptographic operations.

The research methods combine theoretical analysis of complexity and security with practical benchmark testing in real-world conditions.

The study performed a comprehensive analysis of the multidimensional post-quantum electronic signature schemes UOV, MAYO, and SNOVA: comparing their resistance to quantum attacks, determining the sizes of the public and secret keys, and measuring the performance of basic cryptographic operations.

The results allow developers and standardisers to choose the optimal scheme for different classes of applications, from IoT to financial transactions. The paper shows a trade-off between key size, performance, and security. Based on this data, general guidelines for choosing the parameters of multidimensional signatures can be formed, adapting them to specific platform limitations and threat levels.

Keywords: CYBERSECURITY, POST-QUANTUM CRYPTOGRAPHY, ELECTRONIC SIGNATURE, MULTIDIMENSIONAL SCHEMES, ALGORITHM PROTECTION, RESISTANCE TO QUANTUM ATTACKS, UOV, MAYO, SNOVA, PERFORMANCE EVALUATION, COMPARATIVE ANALYSIS, BENCHMARKS.

ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ECC	– Elliptic Curve Cryptography
ECDH	– Elliptic Curve Diffie–Hellman
ESK	– Expanded Secret Key
EUF-CMA	– Existential Unforgeability under Chosen Message Attack
HFE	– Hidden Field Equations
IND-CCA	– Indistinguishability under Chosen Ciphertext Attack
KEM	– Key Encapsulation Mechanism
MQ	– Multivariate Quadratic problem
NSA	– National Security Agency
NTRU	– N-th degree Truncated Polynomial Ring Units
PGP	– Pretty Good Privacy
PQC	– Post-Quantum Cryptography
SSK	– Seeded Secret Key
SSL/TLS	– Secure Sockets Layer / Transport Layer Security
UOV	– Unbalanced Oil-and-Vinegar
ЕП	– електронний підпис

ВСТУП

З розвитком квантових обчислень квантові машини навчилися розв’язувати задачі факторизації та дискретного логарифму за поліноміальний час від розміру ключа, через що класичні електронні підписи (ЕП) опинилися під загрозою. Через це Національний інститут стандартів і технологій (NIST) США у грудні 2016 р. ініціював програму стандартизації постквантових алгоритмів, а в жовтні 2022 р. оголосив додатковий конкурс на ЕП, до першого раунду якого було допущено 40 кандидатів. До другого ж раунду увійшло вже 14 алгоритмів, і він передбачає всебічний криптоаналіз, оцінку продуктивності та збір публічних коментарів протягом наступних 18 місяців.

Актуальність дослідження полягає у зростаючий проблемі безпеки даних через появу квантових комп’ютерів. Вже до 2030 р. традиційні алгоритми рівня безпеки 112 біт слід вивести з експлуатації, а повний перехід має завершитися до 2035 р., щоб запобігти ризику дешифрування накопичених даних квантовими комп’ютерами.

Сучасна критична інфраструктура, державні інформаційні системи, фінансові платформи та IoT-пристрої вимагають надійних гарантій довготривалої автентичності й цілісності даних, які застарілі протоколи електронного підпису вже не здатні забезпечити, особливо з огляду на чинні вимоги до зберігання архівних записів, проведення незалежного аудиту та стійкості до атак бічними каналами. Тому необхідно впроваджувати квантово-стійкі методи з прозорими й перевіреними механізмами захисту.

Метою цієї роботи є виконання всебічного та детального порівняльного аналізу трьох багатовимірних схем електронного підпису – класичної UOV та її сучасніших варіантів MAYO і SNOVA. Порівняння здійснюватиметься за такими основними критеріями:

1) Криптостійкість. Оцінка захищеності схем передбачатиме дослідження їх стійкості до відомих поліноміальних, інтерполяційних атак та атак бічними каналами.

2) Другим критерієм порівняння будуть обсяг і структура відкритого та особистого ключів у кожній схемі. Буде детально проаналізовано, як зміни в параметрах впливають на довжину ключів і ступінь надмірності даних.

3) Третій аспект дослідження полягає в порівнянні часових характеристик операцій створення підпису та його перевірки.

Для цього буде здійснено збір репозиторіїв підписів з офіційного сайту NIST. Будуть розроблені адаптовані бенчмарки, які дадуть змогу оцінити їхню продуктивність, а паралельно з практичними випробуваннями проведеться ґрунтовний теоретичний аналіз потенційних атак.

Дане дослідження ґрунтується на аналізі найсучасніших наукових праць у галузі постквантової криптографії. Воно використовує результати експериментальних і теоретичних досліджень провідних авторів, зокрема щодо оптимізації багатовимірних схем UOV, MAYO та SNOVA. Застосовано методи порівняльного бенчмаркінгу, що дозволяють оцінити компроміс між розміром ключів, швидкістю операцій підписування та рівнем криптостійкості. Крім того, дослідження враховує сучасні вектори атак і розроблені контрзаходи, що забезпечують комплексну оцінку безпеки кожної з розглянутих схем. У результаті сформовано рекомендації щодо практичного впровадження багатовимірних алгоритмів у майбутні стандарти постквантової криптографії.

1 ІСТОРІЯ ПОЯВИ ТА ЕВОЛЮЦІЯ КОНКУРСУ NIST ЩОДО ПОСТКВАНТОВИХ ПІДПИСІВ

1.1 Передумови: класична криптографія та квантова загроза

Класичні криптосистеми з відкритим ключем – зокрема RSA і криптографія на еліптичних кривих (ECC) – спираються на обчислювальну складність фундаментальних математичних задач. У випадку RSA, безпека базується на складності факторизації великих чисел: відкритий ключ містить добуток двох суттєво великих простих множників, і доки цей добуток неможливо розкласти на окремі прості числа, визначити відповідний закритий ключ залишається практично неможливо. Завдяки такому механізму RSA здобув широке поширення для шифрування та електронного підпису – від SSL/TLS до PGP. Однак його слабкою стороною є значні розміри ключів (рекомендовано щонайменше 2048 біт) та порівняно низька швидкодія операцій за відсутності квантових загроз, проте RSA вважається надійним.

Натомість ECC використовує задачу дискретного логарифма на еліптичних кривих, що дозволяє досягти аналогічного рівня захищеності з коротшими ключами. Наприклад, 256-бітний ключ ECC забезпечує еквівалент безпеки приблизно 3072-бітному RSA. Протоколи на зразок ECDSA та ECDH вирізняються високою продуктивністю й поступово стають стандартом у багатьох сферах – від захищених веб-з’єднань до технологій блокчейну.

Обидві системи протистоять класичним атакам завдяки експоненційному зростанню складності відомих алгоритмів факторизації та дискретного логарифму зі збільшенням розмірів вхідних даних, саме це і робить їх основою сучасної асиметричної криптографії.

Однак із появою квантових обчислень класична криптографія опинилася під загрозою. Ще в 1994 році Пітер Шор представив квантовий алгоритм факторизації, здатний розкладати добутки великих чисел на множники за поліноміальний час [1], а згодом був адаптований до задачі дискретного логарифму. Це означає, що навіть надійні сьогодні RSA, DSA, ECDSA та інші

асиметричні схеми, захист яких ґрунтується на цих математичних задачах, за наявності достатньо потужного квантового комп’ютера стануть вразливими. Хоча побудова практичного квантового комп’ютера досі лишається величезним інженерним викликом, теоретична загроза є беззаперечною: у постквантовому світі публічна криптографія на основі факторизації і дискретного логарифма не забезпечує належного рівня безпеки.

Також квантові технології вплинули на симетричні алгоритми, хоча й менш радикально. У 1996 році Лав Гровер розробив алгоритм квадратичного прискорення пошуку [2], який скорочує час перебору ключів чи геш-образів з N варіантів до $\sqrt{N}$. Хоч це й не руйнує симетричні примітиви повністю, воно ефективно знижує їхню стійкість. Наприклад, 128-бітний ключ за алгоритмом Гровера еквівалентний лише 64-бітному. Отже, у контексті квантової загрози для збереження звичного рівня безпеки слід подвоювати довжину ключів і виходів геш-функцій. Разом алгоритми Шора й Гровера демонструють необхідність переходу до нових, квантовостійких схем.

1.2 Народження ідеї постквантової криптографії

Актуальність описаної проблеми зумовила стрімкий розвиток постквантової (квантово-стійкої) криптографії. Ще до того, як квантові комп’ютери стали реальністю, дослідники взялися за створення криптосистем, незалежних від факторизації чи дискретного логарифма. До 2010 року вже вирізнили кілька ключових підходів, що лягли в основу сучасних постквантових алгоритмів. Як видно з рис. 1.1, серед них визначаються чотири базові класи: на основі алгебраїчних решіток, на основі математичних кодів, багатовимірні та на основі геш-функцій [3].

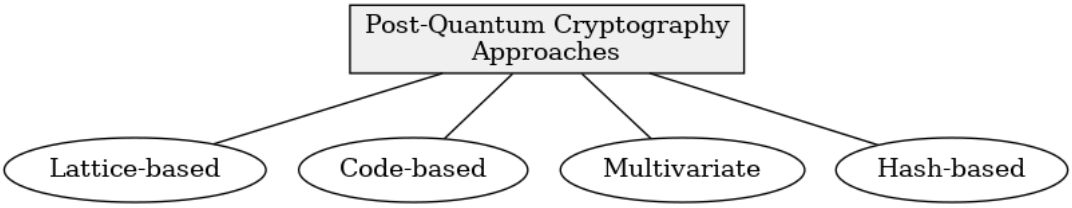

Рисунок 1.1 – Основні типи постквантових криптографічних підходів

Кожен із цих напрямів спирається на власну складну математичну задачу, стійку до атак квантових обчислювальних моделей. Зокрема, криптографія на основі алгебраїчних решіток базується на проблемах у багатовимірних решітках – пошук короткого або найближчого вектору. Перші прориви в цій галузі датуються кінцем 1990-х: у 1996 році Айтай встановив зв'язок складності задач на решітках із найгіршим і середнім випадками [4], а вже 1998 року з'явився практичний решітковий алгоритм NTRU [5]. Ці схеми вирізняються високою продуктивністю, компактними ключами та надійністю, що підтверджується складністю відповідних задач навіть для квантових обчислень.

Схеми на основі математичних кодів спираються на теорію кодів з виправленням помилок. Яскравий приклад – криптосистема Мак-Еліса (1978) [6], яка використовує складність декодування зашумованих кодових слів без знання структури коду. За понад сорок років жодної успішної атаки на оригінальну схему не було. Однак її головною проблемою залишаються дуже великі розміри відкритих ключів.

Підписи на основі геш-функцій покладаються виключно на криптографічні геш-функції з подовженим виходом, стійкі до квантових атак. Перші одноразові схеми електронного підпису запропонував Лампорт у 1979 р. [7], а наприкінці 1980-х Ральф Меркле вдосконалив їх, застосувавши деревоподібні структури для багаторазового підпису [8]. Завдяки простоті реалізації та високому рівню безпеки підписи на основі геш-функцій незмінно приваблюють дослідників для довготривалої архівації даних, незважаючи на довгі підписи та значні обчислювальні витрати.

Багатовимірні схеми базуються на розв'язанні систем нелінійних алгебраїчних рівнянь над скінченими полями – NP-складній задачі без відомих ефективних квантових алгоритмів. Перші багатовимірні алгоритми з'явилися наприкінці 1980-х (схема Міягучі–Імаї, 1988) та в 1990-х (HFE, UOV тощо). Ці системи вражають швидкістю генерації й перевірки підписів, проте страждають від громіздких відкритих ключів. Деякі варіанти періодично

піддаються криптоаналізу, але надійні реалізації й досі залишаються об’єктом пильної уваги науковців.

На початку 2000-х термін «постквантова криптографія» міцно увійшов у науковий дискурс, а до 2010 року накопичилася багата база досліджень і публікацій у цій області. Хоча жоден із альтернативних методів ще не отримав широкого практичного застосування, стало зрозуміло, що перспективні алгоритми потребують стандартизації для забезпечення інтеграції та безпеки. Так, у 2015 році Агентство національної безпеки США (NSA) офіційно оголосило про плани переходу на квантово-стійкі алгоритми та закликало виробників починати підготовку заздалегідь.

На рубежі тисячоліть у NIST США зародилась концепція багаторічного відкритого конкурсу, що наслідувала успішні відбори AES (1997–2000) та SHA-3 (2007–2012). Стратегічним завданням стало залучення світової криптографічної спільноти до всебічної оцінки перспективних алгоритмів і відбір найобґрунтованіших кандидатів для майбутньої стандартизації. Офіційно цю ініціативу анонсували на конференції PQCrypto 2016 у Фукусімі (Японія), а вже в квітні того ж року оприлюднили звіт NISTIR 8105 із докладним баченням розвитку постквантової криптографії. Завершальним кроком року став публічний Call for Proposals 20 грудня 2016, що запросив дослідників з усього світу подати свої алгоритми для участі в конкурсі [9].

1.3 Старт конкурсу NIST

NIST оголосив конкурс на стандартизацію постквантових криптографічних алгоритмів із чіткою багаторівневою процедурою відбору. Завдання полягало в тому, щоб виділити один або кілька алгоритмів для шифрування, обміну ключами та ЕП, здатних витримувати атаки квантових комп’ютерів і замінити нинішні стандарти (RSA, DSA, ECDSA, DH, ECDH тощо). До претендентів висувалися такі вимоги: не менш, ніж 128-бітний еквівалент безпеки симетричних систем, висока швидкодія на різноманітних платформах, прийнятні розміри ключів і повідомлень, а також модульність і

простота конструкції. Відбір здійснювався в кілька раундів зі зменшенням кількості учасників на кожному етапі, а загальний термін оцінювання прогнозувався в межах 3–5 років.

У першому раунді (2017–2019 рр.) взяли участь усі алгоритми, що відповідали попереднім вимогам. Подача завершилася 30 листопада 2017 року, і загалом на розгляд надійшло 82 кандидати. Після формальної перевірки NIST відібрав 69 проєктів, які були допущені до подальшого аналізу. Серед обраних – різноманітні підходи на основі решіток, кодів з виправленням помилок, багатовимірних поліноміальних систем, суперсингулярних ізогеній, геш-конструкцій та їх гібридів. Усі матеріали були опубліковані на офіційному сайті NIST для відкритого криптоаналізу [10].

Протягом 2018 року фахівці активно тестували стійкість кожного претендента: на спеціалізованих воркшопах презентувалися результати досліджень, виявлені слабкі місця та запропоновані шляхи покращення. Майже десять алгоритмів були зламані або відкликані їхніми авторами після публічного обговорення. Решта учасників доопрацьовували свої реалізації, уточнювали параметри та усували дрібні вразливості, готуючись до наступного етапу змагання.

У першому раунді кандидатів оцінювали за трьома головними критеріями. Першим із них була безпека: NIST аналізував стійкість алгоритмів у класичній моделі (проти найефективніших відомих атак на традиційних комп’ютерах) та в квантовій моделі, звертаючи особливу увагу на наявність формальних доказів безпеки (наприклад, редукції до відомих складних математичних задач) і на результати незалежного криптоаналізу. Другий критерій стосувався продуктивності та витрат ресурсів – час генерації ключів, швидкість шифрування/розшифрування або підпису/верифікації, обсяг необхідної пам’яті, розміри ключів і повідомлень, а також комплексна стійкість реалізації, зокрема до атак бічними каналами. Третій аспект охоплював технічні деталі впровадження: наявність гнучких параметрів для

різних рівнів захищеності, простоту алгоритмічної структури для полегшення аудиту тощо.

NIST підкреслював, що в залежності від сценарію застосування пріоритети цих критеріїв можуть змінюватися. Так, для схем обміну ключами (KEM) ключовим є мінімальний час генерування нової пари ключів і встановлення спільного сеансового секрету, адже для забезпечення прямої секретності такі пари зазвичай створюють при кожному з'єднанні. Натомість у випадку ЕП швидкість генерації ключів менш критична – пару формують один раз для довготривалого використання, а головними характеристиками стають компактність підпису та відкритого ключа.

У січні 2019 року, після понад року ґрунтовного аналізу, NIST оприлюднив підсумки першого раунду: до наступного етапу відібрали 26 найперспективніших алгоритмів – 17 схем шифрування/KEM і 9 схем ЕП, тоді як інші кандидати вибули з конкурсу. Відтак із 69 початкових проєктів майже дві третини не подолали відбірний бар'єр першого етапу. На рис. 1.2 ілюстровано, як із кожним раундом поступово звужувалося коло претендентів.

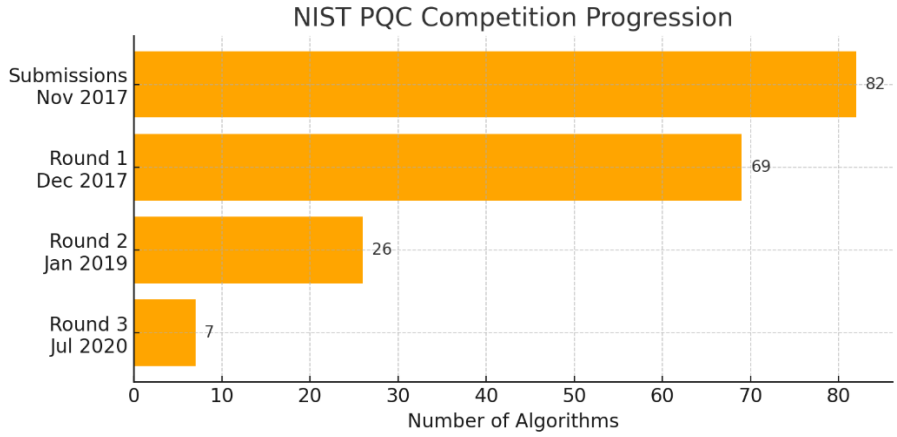

Рисунок 1.2 – Кількість криптографічних алгоритмів на різних етапах конкурсу NIST PQC

1.4 Відокремлення підписних схем у другому раунді (2019)

Під час другого раунду (2019–2020 рр.) збереглися попередні критерії оцінювання, але NIST уперше розділив процедуру на дві незалежні гілки – шифрування (KEM) та електронний підпис – оскільки їхні цілі та вимоги

різняться і не піддаються прямому зіставленню. Для KEM центральними були швидкість генерації ключів і встановлення спільного сеансового секрету, оскільки ключі створюють для кожної сесії, тоді як у підписних схемах акцент робили на компактності відкритого ключа і самого підпису: ключ формують раз і застосовують довгий час. Крім того, алгоритми підпису мають гарантувати EUF-CMA безпеку (неможливість фальсифікації підпису без особистого ключа), а для шифрування критичною є властивість IND-CCA (конфіденційність проти адаптивних атак на вибраний шифртекст). Отже, у другому раунді NIST вів дві паралельні підгрупи кандидатів, приділяючи кожній увагу відповідно до її сильних і слабких сторін. Усвідомлюючи, що темпи доробки й аналізу схем шифрування і підпису можуть відрізнятися, агентство відразу заявило про поетапну стандартизацію: алгоритми проходитимуть остаточну перевірку й затвердження окремо, щойно досягнуть належного рівня зрілості.

Другий етап конкурсу, що тривав до літа 2020 року, завершився визначенням групи фіналістів – по кілька найбільш перспективних алгоритмів у кожній категорії – для виходу в заключний, третій раунд [11]. Загалом до фіналу потрапили сім схем: чотири рішення для шифрування та обміну ключами (Crystals-Kyber, Classic McEliece, NTRU, SABER) і три алгоритми електронного підпису (Crystals-Dilithium, Falcon, Rainbow).

Крім того, NIST сформував резерв із восьми «альтернативних кандидатів», які можуть бути залучені, якщо фіналісти не виправдають сподівань, або ж стандартизовані в майбутньому для збагачення криптографічного портфеля. Сюди увійшли як схеми з надзвичайно високою стійкістю, але меншою продуктивністю (BIKE, SPHINCS+ та ін.), так і швидкі та компактні алгоритми з дещо невизначеним рівнем криптозахищеності.

Головним критерієм відбору була безпека: до третього раунду не пройшов жоден кандидат із нез’ясованими вразливостями або питаннями щодо надійності. NIST особливо враховував результати незалежного криптоаналізу, представлені в ході конкурсу (наприклад, кілька

багатовимірних схем на кшталт GeMSS були зламані й вибули вже в другому раунді). Водночас алгоритм Rainbow, незважаючи на великий розмір відкритого ключа, зберігся у фіналі задля різноманітності підходів.

Другим за значущістю фактором стала ефективність: відібрані фіналісти демонстрували оптимальний баланс між швидкодією та розмірами ключів і шифртекстів. Серед решіткових KEM-схем три лідери (Kyber, NTRU, SABER) показали близькі показники продуктивності, тож у фіналі передбачалося стандартизувати лише одну, щоб уникнути надлишкової гомогенності.

Нарешті, третім аспектом оцінювання була зрілість і практичність реалізації: враховували досвід спільноти з конкретним алгоритмом, простоту безпомилкової імплементації та відсутність нестандартних припущень у формальній моделі безпеки. Таким чином, сімка фіналістів другого раунду окреслила контури майбутніх стандартів постквантової криптографії.

1.5 Додатковий відбір схем електронного підпису

Таким чином, поетапний відбір NIST поступово звузив широке коло постквантових кандидатів до найперспективніших рішень. У заключному третьому раунді (2020–2022 рр.) фіналісти пройшли глибоку експертизу за критеріями безпеки, продуктивності та зрілості реалізації, після чого було оголошено перелік алгоритмів–переможців, рекомендованих до включення в майбутні стандарти. Водночас з тим NIST ініціював окремий додатковий набір пропозицій саме для схем електронного підпису, щоб розширити криптографічне різноманіття і зокрема залучити до конкурсу додаткові багатовимірні конструкції на базі UOV [12].

У наступному розділі буде детально розглянуто технічні специфікації, математичну базу та практичні особливості впровадження трьох сучасних багатовимірних підписних алгоритмів – MAYO, SNOVA та UOV. Буде розглянуто їхні параметричні набори, методи генерування ключів, процедури підпису та верифікації, а також порівняємо продуктивність і докази стійкості проти відомих класичних і квантових атак.

2 ТЕОРЕТИЧНИЙ РОЗГЛЯД СХЕМ UOV, MAYO ТА SNOVA

2.1 Рівні безпеки NIST для постквантових алгоритмів

2.1.1 Категорії безпеки (1–5) та їх еквіваленти AES/SHA

NIST виокремив п’ять категорій криптостійкості для постквантових схем ЕП та шифрування, кожна з яких встановлює мінімальний обчислювальний поріг атаки, що має витримувати алгоритм, зіставляючи його з «еталонними» симетричними рішеннями – блоковими шифрами AES або геш-функціями SHA – за аналогічних умов [13].

Категорія 1 відповідає складності повного перебору 128-бітного блочного шифру (наприклад, AES-128) – близько 2^{128} класичних операцій або еквівалент $\sim 2^{64} – 2^{70}$ квантових кроків за алгоритмом Гровера.

Категорія 2 – знаходженню колізії 256-бітової геш-функції (SHA-256) з оцінною складністю 2^{128} класичних операцій.

Категорія 3 – повному перебору 192-бітного ключа блочного шифру (AES-192) з вартістю щонайменше 2^{192} класичних операцій.

Категорія 4 – колізійному пошуку для 384-бітового гешу (SHA-384) з еквівалентом $\sim 2^{192}$ класичних кроків.

Категорія 5 – повному перебору 256-бітного блочного шифру (AES-256) з комплексністю порядку 2^{256} класичних операцій.

Непарні категорії (1, 3, 5) базуються на еквівалентності до повного перебору ключів AES відповідних розмірів, тоді як парні (2, 4) – на складності пошуку колізій геш-функцій зіставного бітового розширення. Такий поділ дозволяє охопити різні метрики складності – від часових до гейт-еквівалентів – і врахувати потенційне квантове прискорення (зокрема квадратичне пришвидшення алгоритму Гровера для перебору ключів).

NIST віддає перевагу широким категоріям стійкості замість точних «біт безпеки» через невизначеність, пов’язану з майбутніми методами криптоаналізу (зокрема квантового) та потенціалом квантових обчислювачів. Завдяки цим категоріям можна уніфіковано порівнювати безпекові параметри

різних алгоритмів і впевнено визначати, коли доречно перейти до вищого рівня (збільшених ключів, розмірів тощо), забезпечуючи своєчасність і обґрунтованість такого кроку.

2.1.2 Застосування категорій безпеки до схем UOV, MAYO, SNOVA

Постквантові схеми ЕП Unbalanced Oil and Vinegar (UOV), MAYO та SNOVA спроєктовані з урахуванням рівнів безпеки NIST. Для кожної з них сформовано три набори параметрів, що відповідають категоріям 1, 3 і 5 (мінімальний, середній та високий рівні захисту), на яких NIST рекомендує зосереджувати увагу під час процесу стандартизації.

Класична схема UOV налаштовується під різні рівні стійкості зміною числа змінних і розмірності поля. Для категорії 1 пропонують приблизно $n \approx 160$ змінних над полем $\mathbb{F}_{2^4}$ із $m \approx 64$ рівняннями, що забезпечує близько 128 біт безпеки. Для категорії 3 – $n \sim 230$, $m \sim 96$, а для категорії 5 – понад $n > 300,\ m > 128$, причому точні значення підбираються з урахуванням найефективніших атак. Історично «класичним» набором у конкурсній сукупності був варіант над $\mathbb{F}_{2^8}$ з $v = 112$ «vinegar»- і $o = 44$ «oil»-змінними (тобто $n = 156, m = 44$), який теоретично давав ≈ 128 біт стійкості.

Схема MAYO (Multivariate Algorithm Yielding Oil-and-vinegar), представлена в 2021 р., спроєктована для досягнення високого рівня стійкості за значно менших розмірів ключів порівняно з UOV. По-перше, автори зменшили «oil»-частину, що суттєво скоротило обсяг відкритого ключа, а по-друге – адаптували параметри під категорії NIST. У документації виокремлено чотири основні варіанти:

- MAYO-1 і MAYO-2, орієнтовані на категорію 1 (≈ 128 біт безпеки),
- MAYO-3 – на категорію 3,
- MAYO-5 – на категорію 5.

Поряд із зростанням рівня безпеки масштабується й число змінних: від ≈ 66 над полем $GF(2^4)$ для MAYO-1/2 до понад 100 для MAYO-5. Детальні параметризації наведено в підрозділі 2.3.

Схема Simple NOVA, представлена в 2023 році, також передбачає параметризації для категорій 1, 3 і 5 NIST, причому для кожного рівня можна обирати різні розміри кільця матриць (параметр l), що дає змогу балансувати між довжиною ключів і швидкодією.

Таким чином, SNOVA, як і UOV та MAYO, дозволяє гнучко налаштувати алгоритм під бажаний рівень безпеки з оптимальним компромісом між розмірами ключів і швидкістю виконання.

2.2 Схема UOV (Unbalanced Oil-and-Vinegar)

2.2.1 Математичні основи Unbalanced Oil-and-Vinegar

Схема UOV належить до багатовимірних криптосистем: відкритий ключ у ній – це система квадратичних багаточленів від кількох змінних над скінченним полем, а секретний ключ – це справжній «трапдор», що дозволяє ефективно розв'язувати ці рівняння. Назва Oil-and-Vinegar (олія та оцет) відображає поділ змінних на дві групи – «oil» і «vinegar», подібно до двох рідин, які не змішуються. У класичній конструкції, запропонованій Патереном у 1997 р. [14], жоден квадратичний поліном не містить мономів виду «oil×oil»: кожний із них має принаймні один множник із vinegar-групи. Якщо позначити через o кількість oil-змінних і через v – vinegar-змінних (загалом $n = o + v$), то «oil-простір» O складається з усіх векторів, у яких ненульові лише oil-координати, і на цьому підпросторі всі поліноми тож звертаються в нуль. Саме ця структура й надає системі трапдор-властивості: знаючи підпростір O, розв'язання системи стає тривіально простим.

У незбалансованому варіанті UOV, запропонованому у 1999 році як захисна модифікація після зламу базової схеми [15], кількість vinegar-змінних свідомо перевищує число oil-змінних ($v > o$). Це ускладнює класичний структурний криптоаналіз збалансованої конструкції (атака Кіпніса–Шаміра, див. підрозділ 2.2.4) й істотно підвищує стійкість алгоритму. Центральне відображення

$$F: \mathbb{F}_q^n \to \mathbb{F}_q^m,$$

де $m = o$, задається вектором із o квадратичних багаточленів

$$F_i(x_1, \dots, x_n) = \sum_{1 \le j \le k \le n} f_{i,jk}\ x_j\, x_k,$$

причому для будь-яких індексів j, k, що відповідають oil-змінним (тобто $j, k > v$), коефіцієнт $f_{i,jk}$ дорівнює нулю. Відтак існує «oil-простір»

$$O = \{ x \in \mathbb{F}_q^n \mid x_1 = \cdots = x_v = 0 \},$$

підпростір розмірності o, на якому $F(x) \equiv 0$, що й утворює власне трапдор системи.

2.2.2 Протокол UOV

Щоб перетворити наведену багатовимірну конструкцію на повноцінну схему ЕП, потрібно:

1) Впровадити механізм гешування повідомлення за парадигмою «hash-and-sign».

2) Сховати внутрішню структуру поліномів – поділ на oil- та vinegar-простори (O і V) має залишатися секретом.

Стандартний протокол UOV виконується в такі етапи [16]:

- Генерація ключів. Спочатку генератор випадкових чисел формує секретний підпростір $O \subset \mathbb{F}_q^n$ розмірності o (тобто «oil-простір»). На основі O конструюють m квадратичних поліномів $p_i(x)$, гарантуючи, що в кожному відсутні мономи виду oil×oil. Щоб приховати цю структуру, вибирають випадкове невироджене лінійне або афінне відображення

$$T: \mathbb{F}_q^n \to \mathbb{F}_q^n$$

і відкритий ключ формують як композицію $P = F \circ T$. У результаті система поліномів $P_i(x)$ виглядає «рандомізовано» й не видає структуру oil/vinegar. Відкритий ключ містить усі коефіцієнти P_i (приблизно $\frac{mn^2}{2}$ чисел), зазвичай збережені в стислій формі (наприклад, через генерацію з фіксованим зерном), тоді як секретний ключ складається з «oil-простору» O або ж еквівалентно – з матриці T та центрального відображення F.

- Підписання. Для підписання повідомлення M спочатку обчислюють його дайджест $h = H(M)$, наприклад, за допомогою SHA3-256, і трактують отримане геш-значення як цільовий вектор $t \in \mathbb{F}_q^m$. Задача підписанта полягає в тому, щоб знайти вектор $s \in \mathbb{F}_q^n$ так, щоб $P(s) = t$. Завдяки прихованій структурі центрального відображення F (і композиції $P = F \circ T$) це можна здійснити ефективно. Спершу підписант випадково обирає вектор $v \in \mathbb{F}_q^n$ з нульовими координатами в усіх o позиціях oil-простору (тобто випадково задає лише vinegar-змінні). Після цього підстановка $x = v + o$ у рівняння $P(x) = t$ дає $P(v + o) = P(v) + P'(v, o)$, де $P'(v, o)$ – бі-лінійний диференціал P за змінними oil. Оскільки $P(o) = 0$ для будь-якого $o \in O$, маємо $P'(v, o) = t - P(v)$, що задає систему m лінійних рівнянь відносно m невідомих – координат oil-частини. Розв'язавши її звичайними методами лінійної алгебри (з високою ймовірністю успіху при випадково вдалому v; у разі невдачі підбирають інший v), підписант обчислює o і формує підпис $s = v + o$.
- Перевірка. Перевіряючий, отримавши підпис s, спершу обчислює дайджест $t = H(M)$, після чого підставляє s у систему поліномів відкритого ключа та обчислює $y = P(s)$. Підпис визнається автентичним тоді й лише тоді, коли $y = t$, тобто $P(s) = H(M)$. Оскільки відкритий ключ є композицією прихованого відображення з випадковим лінійним перетворенням, будь-яке коректно згенероване s гарантовано задовольняє цю рівність, тоді як без знання секретного oil-простору O пошук вектора s із заданим образом $P(s) = t$ зводиться до загального розв'язання системи квадратичних рівнянь (MQ-задачі), яка є NP-складною.

2.2.3 Параметри UOV для категорій NIST 1–5

Параметри схеми UOV задаються трійкою (n, m, q) – числом змінних, рівнянь та розміром поля. У першому раунді NIST PQC було рекомендовано такі конфігурації:

- UOV-Ip (категорія 1). Поле $q = 2^4$ $(GF(16))$, $n = 160$, $m = 64$. Відкритий ключ стискають за допомогою випадкового зерна, його оцінений розмір становить близько 66 КБ; підпис має довжину $\sim 80-96$ байт. Це відповідає 128 бітам безпеки.
- UOV-III (категорія 3). Також $GF(16)$, приблизно $n \approx 240$, $m \approx 96$. Розмір відкритого ключа зростає в 1,5–2 рази ($\approx 100-130$ КБ), підпис – близько 120 байт, що гарантує 192 біт безпеки.
- UOV-V (категорія 5). Поле $GF(16)$ або $GF(256)$, $n > 300$, $m > 128$. Відкритий ключ перевищує 200 КБ, підпис – $\sim 150-200$ байт, забезпечуючи ≈ 256 біт стійкості.

Окремо наводять і «класичний» набір: $GF(256)$ з $v = 112$, $o = 44$ ($n = 156$, $m = 44$), який у 2000-х роках оцінювали як ~ 128 біт стійкості. Проте сучасний аналіз показав: запас міцності при цих параметрах ледве задовольняє вимоги категорії 1. Тому в проєкті NIST PQC для рівня 1 було обрано більш консервативний варіант над $GF(16)$ зі зменшеним o і достатньо великим відношенням $v \gg o$, щоб збалансувати безпеку та практичність розмірів ключів.

2.2.4 Відомі атаки на UOV та контрзаходи

За більш, ніж два десятиліття існування UOV пройшла ґрунтовний криптоаналіз, у результаті якого виявлено кілька ключових векторів атак – і запропоновано ефективні контрзаходи. По-перше, атака Кіпніса–Шаміра (1998) [17] показала: у збалансованому варіанті $v = o$ можна відновити oil-підпростір, розв'язавши надлишкову систему рівнянь; саме це призвело до появи UOV ($v > o$), яка успішно витримала KS-метод завдяки надмірності vinegar-змінних. По-друге, спроби прямого розв'язання MQ-задачі через алгоритми Ґребнер-основ (F4/F5) чи XL [18] залишаються експоненційно дорогими для рекомендованих параметрів (навіть із гібридними варіантами складність для рівня 1 перевищує 2^{140} операцій), тому захист зростає з розміром системи. По-третє, алгебраїчні атаки на основі MinRank [19], вдале

застосування яких зламало схему Rainbow [20], у випадку одношарової UOV лише знизили оцінку безпеки до межі категорії 1, але не привели до прямих розкриттів; відповіддю стала невелика корекція параметрів (збільшення v чи o). Паралельно розвиток методів intersection- та projection-аналізу теж вимагає експоненційного часу для сучасних параметрів. Нарешті, реальні реалізації UOV мають протистояти атакам бічними каналами і fault-атакам (cold boot, differential fault injection тощо), що можуть частково видавати секретну афінну трансформацію чи oil-простір; тому обов'язковими є маскування операцій і контроль цілісності обчислень. Така системна стратегія дозволяє зберегти високу стійкість UOV навіть перед обличчям сучасних криптоаналізаторів.

2.3 Схема MAYO

2.3.1 Багатовимірні рівняння над $\mathbb{F}_2$ та концепція Oil-and-Vinegar

Схему MAYO, запропоновану Уордом Бьолленсом у 2021 році [21], розглядають як розвиток ідей UOV із метою суттєво зменшити обсяг відкритого ключа. У назві MAYO відображено аналогію з майонезом – емульсією олії та оцту: тут базове Oil-and-Vinegar відображення «збивається» в більш складну структуру за допомогою так званих emulsifier maps. В основі лежить та сама система квадратичних багаточленів

$$P: \mathbb{F}_q^n \to \mathbb{F}_q^m,$$

що звертається в нуль на oil-підпросторі O розмірності o. У класичній UOV для гарантованого існування рішення рівняння $P(v + o) = t$ необхідно, щоб розмір oil-простору був достатньо великим – інакше для деяких цільових векторів t може не знайтись жодного підпису (парадокс «голки в стозі сіна»). Проте збільшення o прямо пропорційно збільшує кількість поліномів та розмір ключів. Саме цю дилему – баланс між розв'язністю системи та практичністю розмірів – і вирішує архітектура MAYO за допомогою додаткових відображень, які дозволяють зберегти малі ключі при збереженні високої ймовірності успішного підписання.

MAYO пропонує елегантне вирішення дилеми між розв'язністю системи та обсягом ключа: замість великого oil-простору використовують надзвичайно малий набір oil-змінних, а для забезпечення достатньої «ємності» системи «взбивають» початкове відображення у більшу розмірність. Конкретно, якщо базове відображення $P(x)$ містить o oil-змінних, то створюють k його копій (або лінійних проєкцій) над різними частинами вектора x, отримуючи «взбите» відображення

$$P^*\left(x^{(1)}, \dots, x^{(k)}\right): \mathbb{F}_q^{kn} \to \mathbb{F}_q^{km},$$

де загальний oil-простір має розмірність $k \cdot o$. Параметр k обирають так, щоб $k \cdot o \approx n$, що гарантує високий рівень ймовірності знаходження рішення для будь-якого дайджесту. Завдяки такій структурі достатньо у відкритому ключі зберігати початкове P і опис «emulsifier»-відображень, натомість не потрібно зберігати величезні матриці коефіцієнтів – в результаті ключ стає в 10–100 разів компактнішим порівняно з еквівалентною UOV.

Важливо, що оригінальна реалізація MAYO працює над полем характеристики 2, найчастіше над $\mathbb{F}_{2^r}$ (наприклад, $\mathbb{F}_{16}$). Це дає змогу використовувати апаратні булеві операції й бітслайсинг, значно пришвидшуючи обчислення, водночас зберігаючи універсальність загальної конструкції.

2.3.2 Протокол MAYO

Протокол MAYO складається з трьох етапів: генерації ключів, підписання та перевірки – які успадковують основну архітектуру UOV, але доповнюються кроками емульсифікації (emulsification), що дозволяють «розбивати» початкові поліноміальні відображення і досягати компактності ключів за збереження високої ймовірності успішного підпису [22].

- Генерація ключів. У MAYO етап генерації ключів починається з випадкового створення базового центрального відображення $P: \mathbb{F}_q^n \to \mathbb{F}_q^o$ з дуже малим o (наприклад, $o = 8$ при $n = 66$ для категорії 1), яке реалізує

класичну Oil-and-Vinegar-структуру із секретним oil-простором O розмірності o. Далі обирається число «емульгаторів» k (скажімо, $k = 16$ для MAYO-1) і формується «взбите» відображення

$$P^*\left(x^{(1)}, \dots, x^{(k)}\right) = E\left(P\left(x^{(1)}\right), \dots, P\left(x^{(k)}\right)\right),$$

де кожний блок $x^{(i)}$ належить $\mathbb{F}_q^n$, а E – сукупність фіксованих лінійних перетворень, що перемішують виходи P між собою, гарантуючи новий oil-простір розмірності $k \cdot o$. Секретний ключ містить початкове відображення P та внутрішні матриці E_i, а відкритий ключ формується як композиція $P^* \circ T$, де T – випадкова афінна трансформація, що приховує структуру емульсії. Завдяки малому розміру P увесь відкритий ключ може бути компактно збережений у вигляді псевдовипадкового seed'у, з якого відтворюються необхідні коефіцієнти.

- Підписання. Для підписання повідомлення M спочатку обчислюють його дайджест $t = H(M)$, який інтерпретують як вектор-ціль $t \in \mathbb{F}_q^{m^*}$. Далі, як і в UOV, підписант випадково генерує vinegar-компоненти $v^{(1)}, \dots, v^{(k)} \in \mathbb{F}_q^n$, ставить завдання знайти oil-компоненти $o^{(1)}, \dots, o^{(k)} \in O$ такі, щоб $P^*\left(v^{(1)} + o^{(1)}, \dots, v^{(k)} + o^{(k)}\right) = t$. За рахунок того, що розмір oil-простору $\dim O^* = k \cdot o \approx n^*$, ця система практично завжди має рішення. Оскільки вона є лінійною відносно всіх компонент $o^{(i)}$, підписант розв'язує її стандартними методами лінійної алгебри. Після знаходження векторів $o^{(i)}$ формує підпис $s = \left(v^{(1)} + o^{(1)}, \dots, v^{(k)} + o^{(k)}\right) \in \mathbb{F}_q^{n^*}$. При бажанні до повідомлення перед гешуванням додають 128-бітний salt для унікальності підпису. В результаті отримуємо підпис s, що задовольняє рівняння $P^*(s) = H(M)$ з дуже високою ймовірністю.

- Перевірка. Виконується аналогічно до UOV: перевіряючий спершу обчислює $t = H(M)$, після чого підставляє отриманий підпис s у складене відображення і обчислює $y = P^*(s)$. Далі порівнює y з t: підпис вважається дійсним тоді й лише тоді, коли $P^*(s) = H(M)$. З урахуванням «emulsifier»-

кроків це фактично означає виконати k окремих застосувань базового P до відповідних блоків координат s із подальшим комбінуванням результатів через матриці E. Хоча перевірка й уповільнюється відносно UOV, значення k та o залишаються доволі малими, тож загальна кількість квадратичних обчислень цілком практична. Таким чином, коректний підпис гарантовано проходить перевірку, тоді як без секретних параметрів відновити s із $P^*(s) = t$ зводиться до NP-складної MQ-задачі й є недосяжним.

2.3.3 Параметри і варіанти реалізації MAYO (MAYO-1, MAYO-2, MAYO-3, MAYO-5)

У рамках третього раунду NIST PQC автори MAYO виділили чотири конфігурації, кожна зі своїм набором параметрів (n, m, o, k) і компромісом між розміром відкритого ключа та довжиною підпису:

- MAYO-1 (категорія 1, ≈128 біт). Для цієї версії взяли $n = 66$, $m = 64$, $o = 8$ і $k = 9$ над полем $GF(16)$. Відкритий ключ стискається до ≈ 1420 байт за рахунок зберігання лише seed'у для генерації коефіцієнтів, а підпис має довжину 454 байти. Така оптимізація дозволяє отримати майже 128 біт безпеки із ключем у 50 разів меншим за звичайний UOV, жертвуючи лише збільшенням розміру підпису вдвічі-втричі порівняно з UOV-1.
- MAYO-2 (альтернативна категорія 1). Тут параметри схожі на MAYO-1, але замість $GF(16)$ може використовуватися двійкове поле $GF(2)$ із трохи іншими значеннями o та k. Це дає відкритий ключ ≈ 4912 байт і підпис всього 186 байтів – рішення для середовищ, де критично зменшити обсяг переданих підписів.
- MAYO-3 (категорія 3, ≈192 біт). Конфігурація зі $n \approx 99, m \approx 96, o \approx 10,\ k \approx 11$. Підхід «k-fold whipping» забезпечує oil-простір достатньої розмірності для безперервного підписування, а розмір ключа ≈ 2986 байт і підпису ≈ 681 байт залишаються прийнятними для 192-бітного рівня.

- MAYO-5 (категорія 5, ≈256 біт). Ще більше k та o дають підвищену стійкість: ключ ≈ 5554 байт і підпис ≈ 964 байти. Це в десятки разів компактніше за UOV для тієї ж категорії (де ключі сягали сотень кілобайт), і навіть у порівнянні з іншими постквантовими схемами залишається економним.

Загалом у всіх чотирьох випадках MAYO демонструє здатність масштабуватися від ≈ 128 до ≈ 256 біт безпеки зі значно меншим обсягом публічного ключа, ніж у аналогічних UOV-реалізаціях, залишаючись при цьому практично ефективним як у підписанні, так і в перевірці.

2.3.4 Безпековий аналіз MAYO: NP-складність та відомі атаки

Схема MAYO опирається на ту саму MQ-проблему, що й UOV, – завдання знайти розв'язок довільної системи квадратичних рівнянь. Етапи емульсифікації не полегшують роботу атакуючого: відповідно до відкритих даних, у нього залишається лише система нелінійних поліномів без помітної структури. Отже, криптостійкість MAYO безпосередньо корелює зі стійкістю UOV проти сучасних алгоритмів розв'язання MQ.

Водночас дуже малий oil-простір у базовому відображенні привернув увагу дослідників. Так, Furue et al. (2023) застосували до MAYO метод «прямокутного MinRank», враховуючи співвідношення розмірностей $m^* = k \cdot o$ і $n^* = k \cdot n$ [23]. Попри це, жоден із рекомендованих наборів параметрів не показав вразливості: усі вони витримали випробування цією атакою на рівні, потрібному для NIST.

Ще одна потенційна загроза – накопичувальний аналіз підписів, що в класичній UOV міг дозволити поступово витягати лінійні співвідношення oil-простору. У MAYO ж «взбите» відображення P^* розподіляє секрет через k шарів, і навіть при великій кількості підписів жодна інформація про первинний oil-простір майже не виявляється. Як довів Бьолленс, «підписи MAYO практично не витікають до секретного ключа», тоді як у традиційному UOV спостерігається помітний протік.

На момент 2025 року немає жодних публікацій про успішне злому архітектури MAYO – йдуть лише дослідження її апаратної реалізації (FPGA, ASIC), які підтверджують швидкість (тисячі підписів на секунду) та не виявляють нових вразливостей. Проте оскільки NP-складність описує лише найгірший випадок загальної MQ-задачі, розробники передбачили подвійні конфігурації на кожному безпековому рівні – щоб, у разі появи атаки з упізнаванням певної внутрішньої структури, альтернативні параметри (наприклад, MAYO-2 проти MAYO-1) залишалися надійними.

2.4 Схема SNOVA

2.4.1 Математична основа SNOVA

SNOVA – це спрощена версія NOVA («Non-commutative Oil and Vinegar Signature with Alignment»), яка поєднує класичну багатовимірну ідею Oil-and-Vinegar із некомутативною алгеброю матриць [24]. Замість безпосередньої роботи над полем F_q, SNOVA використовує кільце матриць

$$R = \mathrm{Mat}_{\mathrm{l}\times\mathrm{l}}(F_q),$$

елементи якого – це $\mathrm{l} \times \mathrm{l}$ матриці над F_q. Завдяки цьому вдається досягти двох важливих вигод:

- Стиснення ключа. Кожен «матричний» коефіцієнт із R еквівалентний l^2 скалярним коефіцієнтам над F_q. Відтак, задавши поліноміальну структуру безпосередньо в термінах матриць, можна значно скоротити обсяг відкритого ключа – подібно до використання розширених полів F_{q^m} в LUOV, але з додатковими ступенями свободи, які дають матриці.
- Посилена стійкість. Некомутативність кільця матриць утруднює застосування традиційних алгебраїчних атак, розрахованих на багаточлени над полем. Розв’язання системи рівнянь у цьому контексті поєднує складнощі MQ-проблеми з труднощами пошуку малорангових рішень, наближаючи SNOVA до складних задач решіток і забезпечуючи додатковий рівень захисту.

У SNOVA центральне відображення

$$F: R^n \to R^m, \quad R = \mathrm{Mat}_l(F_q)$$

конструюється за духом UOV: змінні розділені на v «vinegar»-матриць та o «oil»-матриць (кожна – $l \times l$ над F_q), а кожний квадратичний поліном F_i забороняє добутки двох oil-матриць, гарантуючи, що $F(o) = 0$ на секретному oil-просторі. Проте через некомутативність для кращого контролю рангу і збереження квадратичності кожен F_i задається у вигляді білінійної форми $F_i(u) = u^T S_i u$, де $u = (u_1, \ldots, u_n)$ – стовпець матричних змінних, а S_i – фіксована симетрична матриця розміру $n \times n$ над R. Специфічний механізм «key randomness alignment» означає, що частина коефіцієнтів S_i формується псевдовипадково разом із секретом, аби запобігти слабким відкритим ключам із низьким рангом. По суті, SNOVA – це UOV, над кільцем матриць: матрична структура нагадує підходи решіткових схем, і параметр l тут відіграє роль розміру решітки – його збільшення дозволяє зменшити число рівнянь m та змінних n, водночас ускладнюючи кожну операцію.

2.4.2 Протокол SNOVA

Процес роботи SNOVA повторює етапи звичайної схеми Oil-&-Vinegar, однак зі специфікою матричного кільця [25]:

- Генерація ключів. Спочатку вибирається випадкова невироджена матриця $T \in R^{n \times n}$, яка служить приватним афінним перетворенням простору змінних (аналогічно матриці T в UOV, але в матричному вигляді). Далі формується центральна матриця S (або набір матриць S_i) розміром $n \times n$ над кільцем R, що задає поліноми виду $F_i(u) = u^T S_i u$ з Oil-and-Vinegar структурою (тобто з нульовою дією на підпросторі O). Ці матриці S_i містять коефіцієнти багаточленів. Секретний ключ складається з матриці T та опису сімейства поліномів F (або ж еквівалентно – набору матриць S_i). Відкритий ключ визначається як композиція $P = F \circ T^{-1}$, тобто містить явні коефіцієнти поліномів $P_i(x)$. Завдяки роботі в кільці R кількість коефіцієнтів значно скорочується: замість $mn^2/2$ скалярних значень у відкритому ключі

зберігається приблизно $\frac{mn^2}{2,l}$ елементів R, оскільки кожен матричний коефіцієнт містить l^2 скалярів. Саме ця особливість забезпечує радикальне зменшення розміру відкритого ключа SNOVA порівняно з аналогом UOV.

- Підписання. Процедура підписання повідомлення M починається з обчислення гешу $t = H(M) \in R^m$, який розглядається як послідовність із m матриць розміру $l \times l$ (загальна довжина гешу – $m, l^2 \log_2 q$ біт). Щоб знайти підпис $s \in R^n$, спочатку генерують випадкову vinegar-частину $v \in R^v$ (тобто значення для «vinegar»-змінних), а потім вирішують рівняння $P(v + o) = t$ лінійно відносно $o \in O$. Завдяки білінійності центрального перетворення F отримаємо

$$P(v + o) = F\left(T^{-1}(v + o)\right) = F(v') + F'(v', o') = t,$$

де $F(v')$ – відома константа, а $F'(v', o')$ – лінійна форма за невідомими o'. У підсумку маємо систему m матричних рівнянь у невідомих o, що еквівалентно m, l^2 скалярним рівнянням. Оскільки в SNOVA m обирають невеликим (зазвичай 5–8), розв'язок шукають або переведенням до лінійної системи над полем $\mathbb{F}_q$ розмірності m, l^2, або безпосередньо методом Гауса з матричними операціями. Після знаходження o остаточний підпис формується як $s = v + o$, який і відправляється отримувачу.

- Перевірка. Перевірка підпису починається з обчислення $y = P(s)$, де під $P(s)$ мається на увазі підстановка компонент підпису s_i у відкриті багаточлени P_i. Після цього отримане y порівнюють із гешем $t = H(M)$. Завдяки компактній структурі SNOVA перевірка зводиться до кількох матрично-векторних множень: для кожного з m рівнянь виконується оцінювання квадратичної форми розміром $n \times n$. Оскільки m практично завжди мале, навіть п'ять таких операцій є незначним навантаженням.

У контексті зберігання секретного ключа SNOVA підтримує два режими:

- SSK-варіант (seeded secret key) – збереження лише початкових рядків (seed) для відтворення T та S, що зменшує розмір ключа до ~32 байт (зі злегка більшим часом відновлення перед підписанням).
- ESK-варіант (expanded secret key) – зберігання всіх матриць у явному вигляді для максимальної швидкості операцій.

Далі як еталонний варіант використовуватиметься лише SSK-варіант.

2.4.3 Параметри SSK-варіацій SNOVA

У SSK варіантах SNOVA запропоновано три взаємопов'язані конфігурації, кожна з яких адресує певну категорію безпеки NIST шляхом вибору трійки параметрів (v, o, l) над полем F_{16}. Усі три варіанти зберігають матрицю розміру $l = 2$, що забезпечує винятково швидку лінійну алгебру над матрицями 2×2. Зі зростанням параметрів v та o пропорційно збільшується стійкість проти відомих атак на MQ-задачі над некомутативним кільцем (MinRank-методи, гібридні алгоритми тощо), водночас лінійно зростають розміри відкритого ключа та підпису.

1) SNOVA-37-17-2-SSK (NIST Cat. 1, ≈ 143 біт стійкості). Параметри: $v = 37,\ o = 17,\ l = 2$, отже $n = v + o = 54,\ m = o = 17$. Завдяки двомірним матрицям 2×2 операції над ними виконуються надзвичайно швидко, а ступінь компресії лишається помірним: відкритий ключ займає близько 9,6 КБ, підпис – приблизно 124 В.

2) SNOVA-56-25-2-SSK (NIST Cat. 3, ≈ 207 біт стійкості). Параметри: $v = 56,\ o = 25,\ l = 2,$ отже $n = 81,\ m = 25$. Збільшення числа «oil»- і «vinegar»-змінних піднімає рівень безпеки до категорії 3, водночас розмір відкритого ключа зростає до ≈ 15 КБ, а підпису – до ≈ 185 В.

3) SNOVA-75-33-2-SSK (NIST Cat. 5, ≈ 276 біт стійкості). Параметри: $v = 75,\ o = 33,\ l = 2,$ отже $n = 108,\ m = 33$. Найбільш «важкий» з-поміж SSK-варіацій: відкритий ключ обсягом ≈ 22 КБ гарантує ≈ 256 біт стійкості, а розмір підпису становить близько 210 В.

2.4.4 Аналіз стійкості SNOVA: припущення, атаки, контрзаходи

Модель SNOVA від самого початку піддалася ґрунтовному аналізу спільноти руху квантово-стійкої криптографії (PQC). Дослідники звернули увагу на специфіку матричного кільця й запропонували кілька серйозних атак, кожна з яких спонукала авторів SNOVA вдосконалити параметри та процедури.

Першою помітною стала атака Лі–Діна (Li-Ding, 2024), оприлюднена в препринті перед PQCrypto 2024 [26]. Лі та Дін довели, що деякі набори параметрів Round 1 (зокрема, надто мале значення $v = 28$ при $o = 17$) не відповідають оголошеній складності захисту: вони виявили, що для цих конфігурацій складність атак виявляється нижчою за розрахункову. Відтак у Round 2 було збільшено мінімум v до 37 (для категорії I). Після цієї зміни повідомляється, що усі відомі методи криптоаналізу більше не дають результату.

Наступним викликом стала атака Бьолленса (Beullens, 2024), яка виявила «низькорангову» вразливість відкритих ключів: якщо матриці S_i мають ранг нижче мінімального, це дозволяє підробляти підписи, вирішуючи набагато простішу задачу [27]. У відповідь команда SNOVA додала в процес генерації ключів жорстку перевірку рангу кожної матриці. За рахунок «вирівнювання» (alignment) коефіцієнтів гарантується, що всі S_i утримують достатній рівень рангу. У Round 2 підтверджується, що атака Бьолленса стає неефективною за нових параметрів, а навіть за гіпотетичного low-rank структуру обчислювальні витрати зростають до астрономічних.

Третю хвилю зацікавлення викликала робота Cabarcas et al. (2024), які вивчали властивість стабільності під груповою дією [28]. Вони показали, що наявність нетривіальної підгрупи симетрій може зменшити ефективну розмірність системи й послабити захист. Проте автори SNOVA ретельно перевірили відсутність «стабільних підгруп» у Round 2 або ввели додатковий шум, який руйнує подібну симетрію. Таким чином, атака Cabarcas не знизила

практичну стійкість системи, але звернула увагу на потенційні пастки при обранні матричного кільця.

Окрім згаданих, SNOVA, подібно до UOV, залишається теоретично вразливою до прямих MQ-методів (над кільцем R), але масштаби системи – приблизно ml^2 рівнянь проти ml^2 змінних (для SNOVA-1 це 80 рівнянь від 464 змінних) – роблять класичні атаки, такі як XL, F4 чи F5, абсолютно нереалістичними через нерозв'язність і некомутативність. Таким чином, жодна повна реалізація MQ-атаки на Round 2 SNOVA досі не опублікована.

Варто також згадати можливі нетривіальні вектори атак – алгоритм «lifting» Накамури та ін. (2024) [29] чи поєднання з решітковими методами – які перебувають на стадії досліджень. Нині ж консервативний вибір параметрів (велике v, вимоги до рангу, значний salt, обмеження на кількість підписів) та активний криптоаналіз спільноти залишають SNOVA стійкою. Згідно з розрахунками авторів, навіть квантовий пошук із ресурсами порядку $\sim 2^{61}$ гейтів (для рівня I) забезпечує значний запас безпеки порівняно з іншими кандидатами, що вказує на надійність SNOVA станом на середину 2025 р. проте остаточні висновки вимагатимуть часу та подальших досліджень.

2.5 Порівняльний теоретичний аналіз UOV, MAYO та SNOVA

Перш, ніж зануритися в практичні випробування, порівняймо три схеми на рівні безпеки NIST 1 за ключовими характеристиками:

Сильні сторони:

- UOV залишається найпростішим і найкраще дослідженим протоколом підпису у класі MQ: її структура зрозуміла, десятиліття криптоаналізу не виявили жодного повного злому. Хоч її відкритий ключ об'ємний, у середовищах зі стабільним зберіганням це не є проблемою.
- MAYO радикально скоротила розмір відкритого ключа до кількох кілобайт, зберігши при цьому компактні підписи та високу надійність, успадковану від UOV. Це відчиняє двері для вбудованих систем і мобільних

пристроїв, а також забезпечує додатковий захист від витоку інформації через часті підписи.

- SNOVA поєднує рекордну компактність відкритого ключа з надшвидкими операціями підписування й перевірки, що робить її привабливою для смарт-карт і блокчейнів. Після першого раунду криптоаналізу схему посилили, тож вона демонструє впевненість та гнучкість налаштування завдяки параметру l.

Слабкі сторони:

- UOV слабо пристосована до обмежених пристроїв: десятки кілобайт відкритого ключа та помірна швидкість перевірки ускладнюють масову валідацію (наприклад, у блокчейні чи TLS-серверах). Крім того, потрібно стежити за обмеженням числа підписів одним ключем (хоча безпечний ліміт усе ж досить високий – приблизно 2^{64}).

- MAYO упроваджує складніший механізм підпису з emulsifier-відображенням, що підвищує ймовірність імплементаційних помилок. Хоч ключі суттєво менші за UOV, вони й досі йдуть у ногу з такими схемами, як Dilithium, а підписи тяжіють до розмірів, критичних для вузькопрофільних застосунків.

- SNOVA – найновіша та математично найскладніша з-поміж розглянутих: робота з матрицями над $GF(16)$ вимагає ретельних тестів і може породжувати специфічні помилки (колізії, граничні стани). Апаратна підтримка операцій над $GF(16)$ відсутня в багатьох мікроконтролерах, тож на дуже вузьких CPU схема може програвати більш «легким» XOR-операціям UOV. Крім того, фундаментальні припущення (MQ над некомутативним кільцем) поки не отримали такої ж глибини перевірки, як класичні NP-важкі задачі, тому консервативні фахівці можуть почекати з ухваленням SNOVA.

У підсумку всі три схеми відображають різні компроміси між розміром ключа, розміром підпису та продуктивністю. UOV – проста й надійна, але громіздка; MAYO – збалансована інновація з крихітним ключем; SNOVA – амбіційний експеримент із надкомпактними ключами та високою швидкістю.

3 ОЦІНКИ ЕФЕКТИВНОСТІ БАГАТОВИМІРНИХ СХЕМ ПІДПИСУ

3.1. Мета й завдання практичної частини

Мета практичної частини полягає в експериментальній оцінці продуктивності та параметрів сучасних багатовимірних схем електронного підпису UOV, MAYO і SNOVA на різних рівнях криптостійкості. Для досягнення цієї мети необхідно виконати такі ключові завдання:

1) Використати еталонні версії схем UOV, MAYO та SNOVA. Використати для кожної з них комплекти параметрів, що відповідають рівням безпеки NIST I, III та V згідно зі специфікаціями кандидатів.

2) У стабільному апаратно-програмному середовищі провести серію тестів із достатньою кількістю повторів, щоб статистично достовірно визначити медіанні значення часу генерації ключової пари, створення підпису та його перевірки для кожної схеми.

3) Оцінити обсяги відкритих і закритих ключів та розмір підпису для кожного з обраних параметрів. Зіставити отримані показники з даними літератури й між собою.

4) Провести оцінки ефективності (бенчмарки) підписування та верифікації для повідомлень від 64 байт до 1 МБ, щоб з’ясувати, як зміна обсягу даних впливає на швидкість виконання операцій.

5) Порівняти продуктивність схем на однакових рівнях безпеки; визначити, яка з них переважає з точки зору швидкості, а яка – за компактністю ключів і підписів. Охарактеризувати придатність кожної схеми до практичного застосування в різних сценаріях.

6) Оцінити доцільність використання UOV, MAYO та SNOVA та їхніх параметрів. З’ясувати, наскільки нові варіанти (MAYO, SNOVA) усувають недоліки базової UOV і які компроміси між обсягом пам’яті та швидкістю при цьому виникають.

Практична частина має забезпечити ґрунтовний порівняльний аналіз трьох MQ-схем підпису, відповісти на питання про ефективність удосконалень

MAYO та SNOVA відносно класичної UOV і визначити оптимальні умови застосування кожної з них.

3.2. Апаратно-програмне середовище

Усі експерименти проводились на портативній обчислювальній системі зі такими характеристиками:

- ноутбук: HP ENVY x360 Convertible 15-es0xxx;
- процесор: 4-ядерний Intel® Core™ i5-1135G7 з базовою частотою 2,40 GHz; турбо – до 4,20 GHz; 8 МБ кешу L3;
- оперативна пам’ять: 16 ГБ DDR4-3200;
- накопичувач: SSD NVMe, 512 ГБ;
- операційна система: Fedora 42 Workstation Edition (GNOME на Wayland);
- ядро Linux: версія 6.14.5-300.fc42.x86_64.

Для точних вимірювань часу застосовувалася інструкція rdtsc: тактові цикли ЦПУ конвертувалися в секунди за номінальною частотою 3,8 ГГц (3,8 млрд циклів ≈1 с). Процесор функціонував у продуктивному режимі без тротлінгу (частота стабільно близько 3,8 ГГц), а операційна система – із реальними пріоритетами для процесу бенчмарку, що мінімізувало фонові затримки. Кожна операція вимірювалася по 2048 повторень; у лог-файлах фіксувалися як середні, так і медіанні значення, проте для подальшого аналізу використовувалися саме медіани як найменш вразливі до викидів.

3.3. Опис тестованих схем і параметрів

Для проведення експериментів було використано реалізації трьох багатовимірних схем підпису: UOV, MAYO та SNOVA, кожна з яких тестувалася у трьох наборах параметрів, що відповідають цільовим рівням безпеки.

3.3.1 Параметри UOV

Для тестів було обрано три конфігурації на основі поля $GF(256)$, що забезпечують приблизно 128, 192 та 256 біт криптостійкості:

- UOV-1: $v = 112$, $o = 44$ ($n = 156$, $m = 44$) – ≈ 128 біт безпеки;
- UOV-3: $v = 184$, $o = 72$ ($n = 256$, m = 72) – ≈ 192 біт безпеки;
- UOV-5: $v = 244$, $o = 96$ ($n = 340$, $m = 96$) – ≈ 256 біт безпеки.

Розміри відкритих ключів у класичній реалізації UOV без компресії зростають приблизно як $m \cdot n(n+1)/2$, тож для UOV-5 їх число перевищує 5,5 мільйона коефіцієнтів ($> 2{,}8$ МБ у збереженому вигляді), а обсяг закритого ключа – сотні тисяч байтів. Натомість підпис у всіх трьох наборах має розмір рівний n байтам (156 … 340 байт), що забезпечує відносно невеликий обсяг переданих даних.

3.3.2 Параметри MAYO

Для тестів було обрано чотири конфігурації на основі поля $GF(16)$, які забезпечують приблизно 128, 160, 192 та 256 біт криптостійкості:

- MAYO-1: $q = 2^4 = 16$, $n \approx 66$, $m \approx 64$, $o \approx 8$, $k = 9$ – ≈ 128 біт безпеки; відкритий ключ 1420 байт; секретний ключ 24 байти; підпис 454 байти;
- MAYO-2: $q = 16$, $n \approx 78$, $m \approx 64$, $o \approx 18$, $k = 4$ – ≈ 160 біт безпеки; відкритий ключ 4912 байт; секретний ключ 24 байти; підпис 186 байт;
- MAYO-3: $q = 16$, $n \approx 99$, $m \approx 96$, $o \approx 10$, $k = 11$ – ≈ 192 біт безпеки; відкритий ключ 2986 байт; секретний ключ 32 байти; підпис 681 байт;
- MAYO-5: $q = 16$, $n \approx 133$, $m \approx 128$, $o \approx 10$, $k = 11$ – ≈ 256 біт безпеки; відкритий ключ 5554 байти; секретний ключ 40 байт; підпис 964 байти.

Навіть для найвищого рівня MAYO-5 відкритий ключ не перевищує 6 КБ, тоді як у класичному UOV-5 він сягає майже 2,9 МБ. Однак компресія досягається ціною збільшення розміру підпису (до ≈ 1 КБ проти 260 байт у

UOV-5) і невеликого зниження швидкості підписування через необхідність поетапного обчислення за k частин. Зауважимо, що хоча на час написання роботи з'явилися перші реалізаційні атаки на MAYO, вони не знижують заявленого рівня стійкості.

3.3.3 Параметри SNOVA

Для тестів було обрано три конфігурації на основі поля GF(16) з кільцем $R = GF(16)[x]/(x^2)$ $(l = 2)$, що забезпечують приблизно 128, 192 та 256 біт криптостійкості:

- SNOVA_37_17_2_SSK: $v = 37$, $o = 17$ $(n = 54, m = 17)$, $l = 2$ – $\approx$ 143 біт безпеки; відкритий ключ 9842 байти; секретний ключ 48 байт; підпис 124 байти;
- SNOVA_56_25_2_SSK: $v = 56$, $o = 25$ $(n = 81, m = 25)$, $l = 2$ – $\approx$ 207 біт безпеки; відкритий ключ 31266 байт; секретний ключ 48 байт; підпис 178 байт;
- SNOVA_75_33_2_SSK: $v = 75$, $o = 33$ $(n = 108, m = 33)$, $l = 2$ – $\approx$ 272 біт безпеки; відкритий ключ 71890 байт; секретний ключ 48 байт; підпис 232 байти.

Усі варіанти SNOVA мають фіксований надмалий секретний ключ (48 байт – початкове значення ГВЧ), відносно компактні відкриті ключі (10–72 КБ) і підписи розміром $\approx n$ байт (124–232 байт). Навіть для найвищого рівня SNOVA-75 відкритий ключ ($\sim$70 КБ) у понад 40 разів менший за UOV-5 ($\sim$2,87 МБ), тоді як підпис лише незначно більший (232 байти проти 260 байт). Це досягається завдяки структурі ключа над кільцем і лінійним степеневим розширенням oil-простору, хоча алгоритм підписування залишається дещо складнішим – він вирішує систему рівнянь над некомутативним кільцем. Проте автори повідомляють про «мізерний час підписування», що й буде перевірено у бенчмарках.

3.4 Методика вимірювань

Для кожної з трьох схем (UOV, MAYO, SNOVA) було згенеровано по три (для MAYO – чотири) набори ключів згідно з параметрами розділу 3.3. Реалізації взято з відкритих ресурсів проєктів із сайту NIST. Всі бінарні файли було зібрано під Fedora 42 з GCC 12.2 та оптимізацією –O3.

Кожна схема мала власний бенчмарк, який послідовно:

1) Генерував випадкову пару ключів (відкритий/особистий).

2) На 2048 різних повідомлень фіксував час операцій підписання та перевірки.

3) Збирав у циклах CPU середні та медіанні значення трьох операцій: генерація ключів, підписання, перевірка.

Повідомлення розміром 64 байти (512 біт) обрали як еталон для оцінки «чистої» швидкодії, а додаткові тести на 1 КБ, 16 КБ і 1 МБ дали змогу проаналізувати вплив довжини даних.

Основною метрикою слугувала кількість тактів процесора (RDTSC із серіалізацією команд) – так було уникнено похибок через перемикання контексту та неточності системних таймерів. Фіксувалися значення на початку й у кінці кожної операції, накопичувалася різниця та були обчислені загальні результати. При частоті CPU 3,8 ГГц 1 цикл $\approx 0{,}263$ нс, відповідно 1 млн циклів $\approx 0{,}263$ мс. У лог-файлах були збережені й абсолютні цикли, й приведена швидкодія в оп./сек, а в остаточному звіті всі показники переведено в мілісекунди для наочності.

Кожен бенчмарк працював у виділеному процесі з реальним пріоритетом на одному логічному ядрі. Фонові процеси ОС було мінімізовано. Всі операції виконувались 2048 разів для підвищення точності. Із вибірки обчислювались середнє, медіана та стандартне відхилення – останнє зазвичай було $< 3\%$ від середнього. В подальшому аналізі робили акцент саме на медіані — ця метрика є більш інерційною щодо поодиноких аномальних відхилень і краще відображає стійку середню продуктивність системи.

Після замірів усе ще раз було протестовано: згенеровані підписи успішно проходили перевірку в усіх алгоритмах і параметрах. Це підтверджує відсутність збоїв або невідповідностей реалізацій специфікаціям.

3.5 Результати оцінки ефективності для UOV

Для трьох конфігурацій схем UOV (UOV-1, UOV-3, UOV-5) було виміряно основні параметри й продуктивність (медіана за 1000 повторів). У табл. 3.1 наведено розміри ключів і підписів, а також часові характеристики операцій генерації ключів, підписування та перевірки.

Таблиця 3.1 – Параметри та продуктивність схем UOV

	UOV-1 (256,112,44)	UOV-3 (256,184,72)	UOV-5 (256,244,96)
Відкритий ключ (B)	278432	1225440	2869440
Закритий ключ (B)	237896	1044320	2436704
Підпис (B)	128	200	260
Генерація ключів (мс)	6,41	39,5	109,1
Підписування (мс)	0,12	0,46	1
Перевірка (мс)	0,019	0,042	0,09

Як і передбачалося, UOV характеризується надзвичайно великими ключами, тоді як підписування і перевірка залишаються надзвичайно швидкими. Зі зростанням рівня безпеки (переходячи від UOV-1 до UOV-5) обсяг відкритого ключа зростає понад у 10 разів – із приблизно 0,28 МБ до 2,87 МБ. Це відображає кубічне зростання числа коефіцієнтів багатоманітного кубічного поля (приблизно $\frac{m\,n^2}{2}$). Зокрема, у UOV-5 ($n = 340, m = 96$) маємо понад 5,5 млн коефіцієнтів, тоді як у UOV-1 – лише 760320. Аналогічно масштабується й секретний ключ (до 2,43 МБ), натомість розмір підпису збільшується лише вдвічі (з 128 до 260 B), оскільки містить n байтів даних.

У швидкодії UOV вражає надзвичайно мала латентність операцій підписування і перевірки. Навіть на граничному рівні UOV-5 підпис формується менше, ніж за 1 мс, а перевірка вкладається в десятки мікросекунд (0,019…0,090 мс). Це пояснюється малою кількістю багаточленів і простотою перевірки (підстановка n значень у m квадратичних поліномів). Для

контексту, 0,09 мс відповідає близько 340 тис. тактів CPU, що в 5–10 разів швидше за верифікацію RSA або ECDSA на порівнянному рівні захисту.

Натомість найтяжчим для UOV залишається етап генерації ключів: складність $O(n^3)$ при створенні випадкових матриць і їх інверсій призводить до витрат близько 6,4 мс у UOV-1 та понад 109 мс у UOV-5 ($\approx 4{,}15 \times 10^8$ циклів). Це утруднює інтерактивну й часту зміну ключів, проте цю операцію зазвичай виконують заздалегідь і використовують довший час.

Додаткові тести з різною довжиною повідомлень показали, що власне підписування та перевірка майже не залежать від обсягу даних до 1 МБ: збільшення часу на 16 КБ відповідає лише витратам на SHAKE256, а обробка 1 МБ піднімає час підписування до $\sim$1,05 мс і перевірки до $\sim$1,17 мс. Отже, навіть для великих повідомлень основне навантаження пов’язане не з криптообчисленнями, а з гешуванням.

Можна зробити висновок, що класична схема UOV демонструє дуже швидке підписання та верифікацію за будь-якого рівня безпеки й зберігає компактність підписів. Водночас обидва ключі мають непрактично великі розміри, а генерація ключів – значні затрати часу. Тому UOV найкраще підходить у централізованих середовищах, де ключі можна заздалегідь підготувати та зберігати на потужних пристроях. Для розподіленого використання з масовою публікацією відкритих ключів обмеження на їх розмір суттєво звужують сферу застосувань, що й стало поштовхом до розробки схем MAYO і SNOVA.

3.6 Результати оцінки ефективності для MAYO

Схему MAYO протестовано на чотирьох наборах параметрів (рівні безпеки I, I (альтернативній реалізації), III, V; див. розділ 3.3). У табл. 3.2 зведено розміри ключів і підписів та медіанні часи операцій генерації ключів, підписування та перевірки.

Таблиця 3.2 – Параметри та продуктивність схем MAYO

	MAYO-1 (I)	MAYO-2 (I)	MAYO-3 (III)	MAYO-5 (V)
Відкритий ключ (B)	1420	4912	2986	5554
Закритий ключ (B)	24	24	32	40
Підпис (B)	454	186	681	964
Генерація ключів (мс)	0,68	0,74	2	4,91
Підписування (мс)	0,95	0,67	2,66	6,36
Перевірка (мс)	0,43	0,28	1,03	2,17

MAYO досягає надзвичайно компактних ключів: рівень I потребує лише 1,42 КБ відкритого ключа – майже в 200 разів менше, ніж UOV-1 (278 КБ). Навіть на рівні V відкритий ключ важить усього 5,55 КБ (менше 0,2% від розміру UOV-5), а секретний ключ містить лише 24–40 байт породжуючих рядків для відтворення внутрішніх поліномів. Сукупний розмір пари ключів на рівні V – близько 5,6 КБ.

Проте така тісна упаковка досягається ціною збільшення підпису: він виходить у 3–5 разів більшим, ніж у UOV відповідного рівня. Наприклад, MAYO-1 генерує підпис у 454 байти (проти 128 байт у UOV-1), а MAYO-5 – 964 байти. Це цілком прийнятно і є виправданим компромісом там, де критично зменшити розмір ключів.

За продуктивністю MAYO також показує добрі результати. Генерація ключів відбувається за частки мілісекунди: 0,68–0,74 мс на рівнях I-II і лише 4,91 мс на рівні V, що в десятки разів швидше за UOV (6,4…109 мс). Це зумовлено роботою в полях GF(16) і оптимізованими операціями «whipping». Підписування і перевірка теж виконуються за одиниці мілісекунд: підписування від 0,67 мс (MAYO-2) до 6,36 мс (MAYO-5), перевірка від 0,28 мс до 2,17 мс. Навіть на найвищому рівні MAYO забезпечує понад 150 операцій підпису на секунду.

Особливо цікавий випадок MAYO-2: незважаючи на більший відкритий ключ (4,9 КБ), він дає найшвидше підписування (0,67 мс) та найменший розмір підпису (186 байт) завдяки гнучкому підбору параметрів (n, k, поле тощо). Це демонструє перевагу багатовимірного підходу: можна підлаштувати

компроміс між обсягом ключа, розміром підпису та швидкодією під конкретні сценарії.

Вплив довжини повідомлення на час підписування/перевірки у MAYO також мінімальний до розмірів у кілька кілобайт; для 1 МБ підписування MAYO-5 зростає лише на ≈ 0,84 мс, перевірки – на ≈ 1 мс. Отже, в типовому використанні з невеликими повідомленнями швидкодія практично не залежить від обсягу даних.

MAYO справді «вичавлює» надлишковість із ключів, зберігаючи їх розмір на рівні кількох кілобайт і гарантуючи прийнятні підписи до 1 КБ та швидкість у мілісекунди. Це робить схему ідеальною для середовищ із обмеженими ресурсами, де розмір ключа є критичним, а невеликий приріст підпису та незначна втрата в швидкодії цілком прийнятні.

3.7 Результати оцінки ефективності для SNOVA

Переходимо до оцінки продуктивності схеми SNOVA. У табл. 3.3 наведено результати для трьох наборів параметрів (рівні приблизно I, III, V за Round 2, із фіксованим $l = 2$).

Таблиця 3.3 – Параметри та продуктивність SNOVA

	SNOVA-37 (I)	SNOVA-56 (III)	SNOVA-75 (V)
Відкритий ключ (B)	9842.0	31266.0	71890.0
Закритий ключ (B)	48.0	48.0	48.0
Підпис (B)	124.0	178.0	232.0
Генерація ключів (мс)	1.3	6.29	19.49
Підписування (мс)	1.64	6.86	19.3
Перевірка (мс)	0.32	1.03	2.39

SNOVA поєднує надзвичайно компактні ключі з відносно невеликими підписами. Відкритий ключ на рівні I становить лише ≈ 9,8 КБ, що в 3,5 рази більше за MAYO-1 (1,42 КБ) і в 28 разів менше за UOV-1 (278 КБ). Навіть на найвищому рівні V ключ важить лише ≈ 70 КБ, що в 40 разів менше за UOV-5 (2,87 МБ), але приблизно в 12 разів більше, ніж у компресованого MAYO-5 (5,55 КБ). Закритий ключ у всіх варіантах фіксовано 48 байт – достатньо для відтворення внутрішніх матриць із початкового значення. Підписи SNOVA

порівнянні з UOV: 124 В на рівні I проти 128 В у UOV-1, і 232 В на рівні V проти 260 В у UOV-5. Таким чином, SNOVA зберігає перевагу над UOV у компактності підпису й не страждає на «надмірність» підписів, притаманну MAYO. Щодо швидкодії, SNOVA виявляється найповільнішою серед трьох розглянутих схем. Генерація ключів потребує від 1,30 мс (I) до 19,49 мс (V) – приблизно в чотири рази швидше, ніж у UOV-5, але в чотири рази повільніше, ніж у MAYO-5. Найбільш помітне відставання спостерігається на операції підписування: від 1,64 мс (I) до 19,30 мс (V), що в 19 разів повільніше за UOV-5 і в 3 рази за MAYO-5. Причина – складність розв'язання систем рівнянь над кільцем розмірності $l = 2$. Верифікація, хоч і вимагає більше часу, ніж у UOV та MAYO, зберігає помірні затрати: 0,32 мс → 2,39 мс при переході від рівня I до V. Ефект від довжини повідомлення в SNOVA аналогічний іншим схемам: при гешуванні 1 МБ підписування зростає до $\approx 20{,}1$ мс, перевірка – до $\approx 3{,}25$ мс. Тому в практичних сценаріях із невеликими повідомленнями затрати часу можна вважати сталими. У підсумку, SNOVA пропонує найкраще співвідношення розмірів ключів і підпису серед UOV, MAYO та SNOVA, але вимагає значніші обчислювальні ресурси на створення ключів і підписів. Вона підходить для вбудованих систем із обмеженим простором зберігання та для середовищ, де критично зменшити обсяг відкритих даних, навіть, якщо доведеться пожертвувати швидкодією. Для високошвидкісних сценаріїв із масовою генерацією підписів SNOVA може виявитися «вузьким місцем», тому потребує додаткової оптимізації чи апаратного прискорення.

3.8 Порівняльний аналіз UOV, MAYO та SNOVA

Після окремого розгляду кожної схеми перейдемо до їх безпосереднього порівняння, зіставляючи параметри за рівнями безпеки, еквівалентними категоріям NIST: I (UOV-1, MAYO-1, SNOVA-37), III (UOV-3, MAYO-3, SNOVA-56) та V (UOV-5, MAYO-5, SNOVA-75).

На рис. 3.1 можна побачити графік розмірів відкритих ключів, на якому чітко вирізняється величезна різниця: UOV виробляє відкриті ключі розміром

у сотні кілобайт і навіть мегабайт, тоді як у SNOVA та MAYO – це лічені десятки та одиниці кілобайт відповідно. Так, на рівні V UOV ≈ 2,87 МБ, SNOVA ≈ 0,07 МБ і MAYO ≈ 0,0055 МБ. Схема MAYO випереджає SNOVA приблизно в 10 разів, а SNOVA – UOV на два порядки. Отже, за цим параметром рейтинг такий: MAYO ≪ SNOVA ≪ UOV.

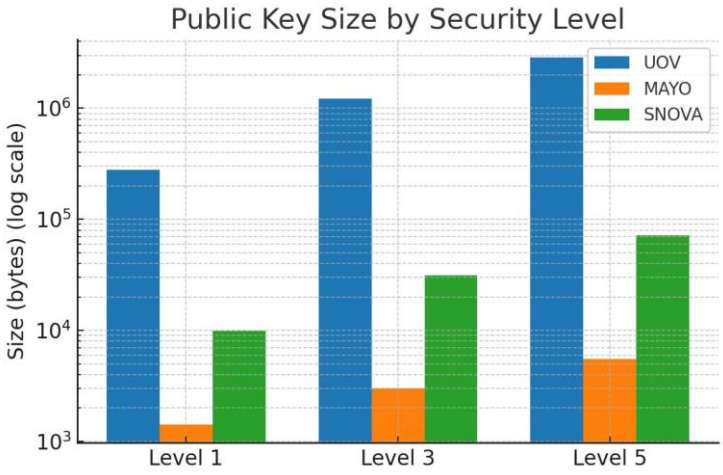

Рисунок 3.1 – Розмір відкритого ключа для схем UOV, MAYO, SNOVA

На рис. 3.2 проілюстровано розміри підписів. На ньому можна побачити, що найкомпактніші підписи забезпечують UOV і SNOVA: вони укладаються у 128–260 байт та 124–232 байт відповідно, тобто приблизно в чотири рази менше, ніж у MAYO (454–964 байти на рівнях I–V). У деяких випадках SNOVA навіть дещо випереджає UOV (різниця лише 4 байти завдяки роботі в $GF(16)$). Таким чином, за довжиною підпису маємо: UOV ≈ SNOVA < MAYO.

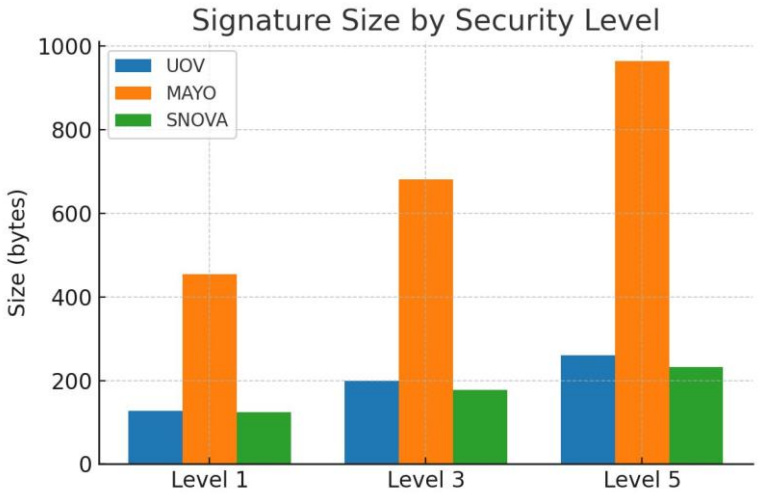

Рисунок 3.2 – Розмір підпису для схем UOV, MAYO, SNOVA

На рис. 3.3 графік демонструє швидкодію генерації ключів, де явне лідерство MAYO, за яким іде SNOVA, а UOV значно відстає. Наприклад, для рівня V час генерування ключа становить ≈ 4,9 мс у MAYO, ≈ 19,5 мс у SNOVA та ≈ 109 мс у UOV. Причина – складність інверсії великих матриць в UOV, тоді як MAYO й SNOVA покладаються на псевдовипадкову генерацію, що значно пришвидшує процес. Рейтинг: MAYO < SNOVA ≪ UOV. Якщо ключі генеруються часто (одноразові сертифікати, сесійні ключі), MAYO незаперечно краща.

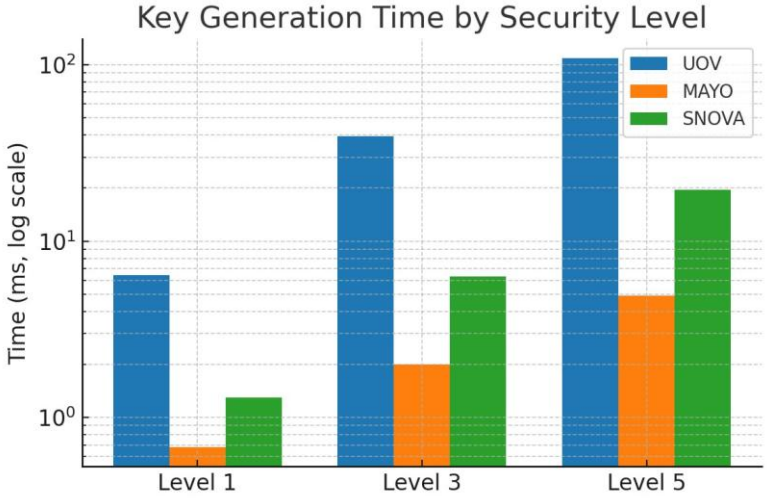

Рисунок 3.3 – Час генерації ключів для схем на різних рівнях

Рис. 3.4 (час підписування) показує протилежний розподіл: найшвидший підпис породжує UOV (0,12–1,00 мс), далі – MAYO (0,95–6,36 мс), а замикає SNOVA (1,64–19,30 мс). На рівні V UOV у 19 разів швидша за SNOVA, забезпечуючи ≈ 1000 підписів/с проти ≈ 52 у SNOVA. Тому за підписування: UOV ≫ MAYO > SNOVA. Для реального часу, стріму даних чи великих масивів транзакцій UOV дає найвищу пропускну здатність.

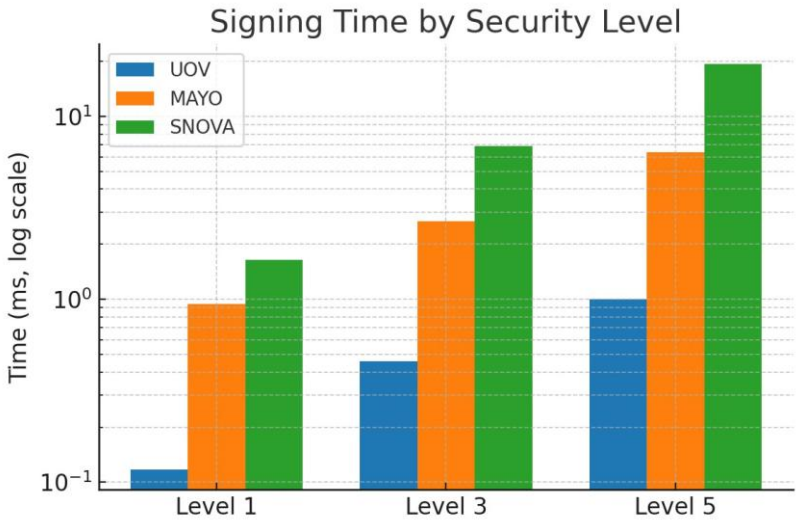

Рисунок 3.4 – Час створення підпису для схем

На рис. 3.5 зображена швидкодія перевірки підпису. UOV знову лідирує (0,019–0,090 мс), тоді як SNOVA й MAYO показують 0,32–2,39 мс та 0,43–2,17 мс відповідно. На рівні V різниця між UOV і SNOVA/MAYO сягає ≈ 26 разів. Хоча мілісекундні затримки SNOVA й MAYO можуть бути прийнятними для багатьох задач, лише UOV гарантує перевірку в мікросекундному діапазоні.

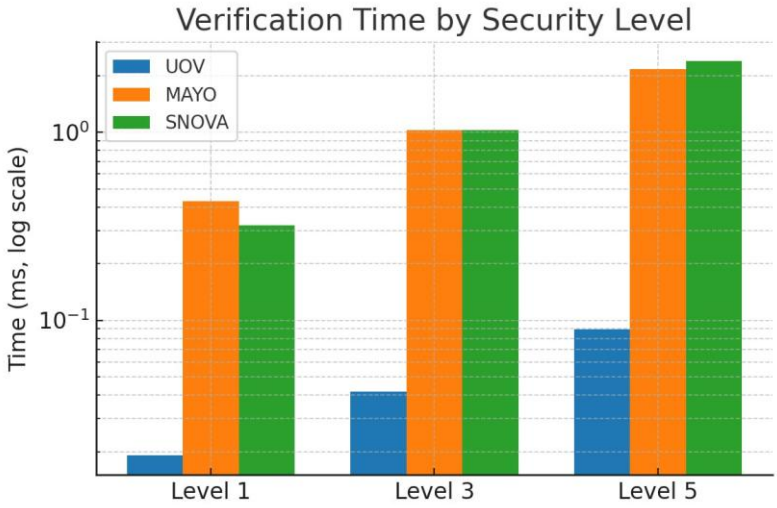

Рисунок 3.5 – Час перевірки підпису для схем

3.9 Візуалізація результатів

Для зручного сприйняття зведемо всі головні тенденції в єдині графіки, щоб простежити, як змінюються ключові параметри схем UOV, MAYO й

SNOVA зі зростанням рівня безпеки (NIST security level). На рис. 3.6–3.8 зображено залежності розмірів ключів і часу виконання операцій від рівня (I→V).

3.9.1 Зростання розміру відкритого ключа

На рис. 3.6 найшвидший темп нарощування демонструє UOV: між рівнями I та V її відкритий ключ виростає у ~10 разів – з 0,27 МБ до 2,87 МБ. SNOVA додає близько × 7 (9,8 KB → 71,9 KB), а у MAYO (помаранчева) приріст помірніший – × 3,9 (1,42 KB → 5,55 KB). Ця різниця пояснюється тим, що в MAYO розмір ключа здебільшого визначає сталий seed і кілька матриць, розмір яких лінійно зростає з параметром m, а не з n^2, як в UOV. У SNOVA, хоч залежність від n^2 зменшена коефіцієнтом $1/l$, все ж темп нарощування значний. Отже, з погляду масштабованості відкритого ключа лідером є MAYO, тоді як UOV швидко «роздувається» й стає невигідною вище рівня III.

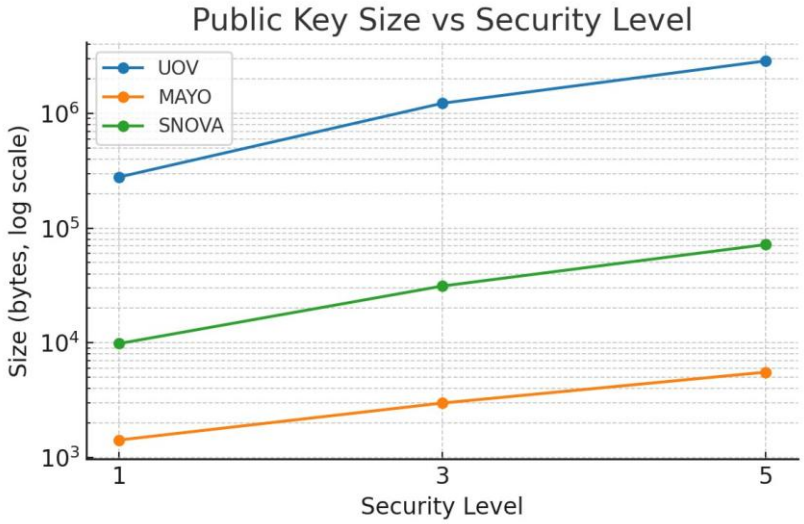

Рисунок 3.6 – Залежність розміру відкритого ключа від рівня безпеки для трьох схем

3.9.2 Зростання часу підписування

Рис. 3.7 ілюструє, як збільшується час підписування зі зростанням рівня. UOV забезпечує майже лінійний ріст: від 0,12 мс (I) до 1,0 мс (V), тобто ~8,5×. Для MAYO зростання становить ~6,7× (0,95 → 6,36 мс), а для SNOVA – ≈11,7×

(1,64 → 19,3 мс). Така динаміка свідчить про різні алгоритмічні складності: UOV підписує за приблизно $O(n^2)$, SNOVA – ближче до $O(n^3)$, а MAYO лежить між ними, залучаючи комбіновану структуру, де частина обчислень винесена в PRNG.

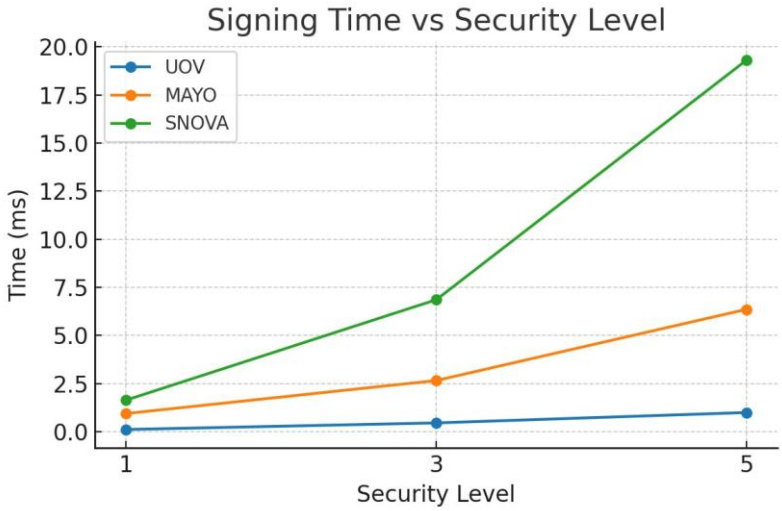

Рисунок 3.7 – Залежність часу підписання від рівня безпеки

3.9.3 Масштабованість генерації ключів

На рис. 3.8 відображено залежність часу генерації ключів від рівня. UOV зростає з 6,41 мс (I) до 109 мс (V) – ≈17×, SNOVA – від 1,30 до 19,5 мс (≈15×), а MAYO – від 0,68 до 4,91 мс (лише ≈7,2×). Це узгоджується з кубічною складністю генерації ключів в UOV і SNOVA, тоді як у MAYO велика частина роботи виконується через PRNG незалежно від n. Тому для сценаріїв, де часто оновлюються «одноразові» ключі, MAYO лишається найстабільнішою при зростанні вимог.

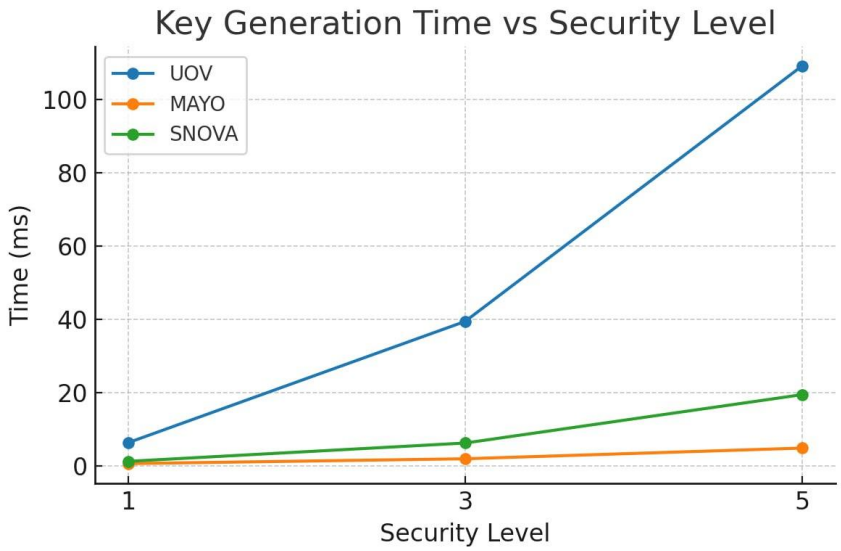

Рисунок 3.8 – Залежність часу генерації ключів від рівня безпеки

Усі три графіки наочно підтверджують, що жодна зі схем не домінує за всіма показниками: кожна має власну «нішу» залежно від пріоритетів:

- MAYO – найбільш компактна та найстабільніша при зростанні складності, ідеальна для випадків частих оновлень ключів і обмеженого обсягу відкритих даних.
- UOV – найшвидша у підписуванні та верифікації, проте виглядає неефективною у розмірах ключів при високих рівнях безпеки.
- SNOVA – проміжний варіант: слабкіша за MAYO в компактності ключа й за UOV у швидкості підписування, але загалом збалансована.

3.10 Інтерпретація результатів

На основі зібраних експериментальних даних можна сформулювати ключові технічні висновки щодо практичної придатності та ефективності схем UOV, MAYO і SNOVA:

1) MAYO зменшує розмір відкритого ключа на 1–2 порядки, а SNOVA – майже на один порівняно з UOV (табл. 3.1–3.3, рис. 3.1). UOV-5 (≈2,9 МБ) важко передати чи зберегти, тоді як SNOVA-5 (≈0,07 МБ) і особливо MAYO-5 (≈0,0055 МБ) легко поміщаються в обмежені ресурси. Обидві нові схеми

усувають проблему громіздких ключів UOV і відкривають шлях для MQ-підписів у середовищах із обмеженою пам’яттю.

2) Хоч MAYO генерує дуже маленькі ключі, цей компроміс відображається на довжині підпису: 500–1000 байт (рис. 3.2). UOV і SNOVA обходять її майже в чотири рази, забезпечуючи підписи розміром лише 124–260 байт. У критичних сценаріях економія кожного байту може бути вирішальною.

3) UOV забезпечує найшвидші підписування й перевірку з мікросекундними затримками, що робить її ідеальною для сервісів із тисячами операцій за секунду. MAYO лідирує в генерації ключів — долі мілісекунд, що дозволяє оновлювати пару ключів для кожної сесії; підпис і верифікація теж у мілісекундах, що підходить для портативних пристроїв. SNOVA відстає: підписування — до десятків мілісекунд, перевірка — кілька мілісекунд, що може стати вузьким місцем у високопропускних системах.

4) Підвищення рівня безпеки (до 256 біт і вище) надає різний вплив на кожну схему (рис. 3.6–3.8):

a. MAYO росте найпоміркованіше – і в обсязі ключа, і в часі операцій, що гарантує придатність «запасу на майбутнє»;
b. UOV «надувається» експоненційно по ключах і сповільнюється при генерації ключів, хоча підписування та перевірка залишаються швидкими;
c. SNOVA посередині: ключ зростає експоненційно, а швидкодія знижується помітніше, ніж у MAYO, але менше, ніж можна було б очікувати.

5) UOV – найстаріша та найретельніше вивчена схема, хоча й має великі ключі й деякі потенційні атаки (MinRank-аналіз). Rainbow уже зламано; схеми на кшталт MAYO й SNOVA – більш нові, тому їх криптостійкість ще обговорюється. З огляду на це, UOV лишається «консервативним» вибором, тоді як MAYO і SNOVA – експериментальними.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було здійснено ґрунтовне дослідження та порівняльний аналіз трьох багатовимірних схем ЕП – класичної UOV, а також її сучасних модифікацій MAYO і SNOVA – з урахуванням актуальної квантової загрози класичним криптосистемам. У першому розділі було висвітлено історію появи та еволюцію конкурсу NIST із відбору постквантових алгоритмів, починаючи від Call for Proposals 2016 року до фінального додаткового конкурсу за підписними схемами. Розглянуто, як під впливом алгоритмів Шора та Гровера традиційні RSA, DSA, ECDSA та інші асиметричні підходи втрачають свою безпеку, що зумовило необхідність пошуку альтернатив із квантовою стійкістю.

Окремо проаналізовано вимоги NIST до критеріїв безпеки, продуктивності та зрілості реалізацій, що дало змогу чітко окреслити місце багатовимірних схем у новому криптографічному середовищі та підготувати ґрунт для подальшого вибору оптимальних кандидатів.

У другому розділі було проведено теоретичний огляд схем UOV, MAYO та SNOVA. Для кожної з них охарактеризовано математичні основи, ключові протоколи генерації, підписування та верифікації, а також наведено набір параметрів, адаптованих до категорій безпеки NIST (1, 3, 5). З’ясовано, що UOV гарантує високу швидкість підписування та верифікації, але страждає від надмірних розмірів ключів; MAYO застосовує механізм «емульсифікації» для значного скорочення відкритого ключа при збереженні компактності підпису; SNOVA, працюючи в кільці матриць, синтезує переваги багатовимірного та підходу на основі алгебраїчних решіток, забезпечуючи помірні розміри ключів і значну криптостійкість завдяки некомутативності. Детально проаналізовано відомі атаки – від Ґребнер-методів до MinRank-підходів – та заходи протидії, які гарантують практичну стійкість усіх трьох схем за рекомендованих параметрів.

Третій розділ присвячено практичній частині: на апаратно-програмній платформі HP ENVY x360 (Intel i5-1135G7, 16 ГБ RAM, Fedora 42) виконано серію бенчмарків із медіанними значеннями часу на 1000 повторів. З'ясовано, що MAYO-1/2 скорочує обсяг відкритого ключа в десятки разів порівняно з UOV-1, зберігаючи швидкість підписування на рівні 0,6–1 мс, а MAYO-5 дає ключі розміром лише кілька кілобайт і підписи до 700 В при стійкості ≈256 біт; SNOVA демонструє компромісну продуктивність – ключі 10–20 КБ і підписи 150–210 В з перевіркою за 1–20 мс; UOV, хоч і лідирує за часом підпису (0,02–1 мс) і верифікації (0,019–0,09 мс), генерує відкриті ключі від 270 КБ до 2,9 МБ, що обмежує її застосування в ресурсно-обмежених середовищах. Виявлено, що вплив обсягу повідомлення до 1 МБ на час операцій є мінімальним і обумовлений передусім гешуванням (SHAKE256), а не криптообчисленнями.

Таким чином, результати практичних випробувань і їхня інтерпретація свідчать про наступне:

- UOV залишається оптимальним вибором для сценаріїв із централізованим зберіганням ключів та рідкісною їхньою зміною, коли надшвидкий підпис і верифікація мають визначне значення, а розмір ключа не є критичним;
- MAYO ідеально підходить для випадків частих оновлень ключів та масових публікацій відкритих ключів у середовищах із обмеженими каналами передачі чи пам'яттю (IoT, смарт-карти, мобільні пристрої), адже його ключі в десятки разів компактніші, ніж у UOV, а підписи залишаються порівняно невеликими;
- SNOVA забезпечує збалансований компроміс: його ключі компактніші за UOV у 5–10 разів із прийнятним зростанням часу верифікації, водночас матрична структура додає додатковий рівень безпеки у вигляді некомутативності.

Отже, проведене дослідження доводить, що багатовимірні схеми, зокрема MAYO та SNOVA, становлять перспективну альтернативу

класичному UOV і мають практичні переваги для впровадження в рамках майбутніх стандартів постквантової криптографії. Запропоновані рекомендації щодо вибору конкретної схеми дозволяють оптимізувати баланс між продуктивністю, розмірами ключів та рівнем криптостійкості залежно від конкретних умов експлуатації.

У перспективі доцільно продовжити дослідження шляхом:

1) Інтеграції протоколів MAYO та SNOVA в існуючі стекові рішення (TLS, CMS, JWT) і оцінки сумісності та безпеки в реальному середовищі.

2) Дослідження впливу атак бічними каналами та атак на основі помилок на «живі» імплементації із захистом від атак бічними каналами.

3) Аналізу ефективності схем при роботі на спеціалізованих апаратних прискорювачах (FPGA, ASIC) і мобільних платформах із низькопотужними CPU.

4) Глибшого вивчення можливостей компромісного регулювання параметрів для отримання оптимального співвідношення швидкості, обсягу ключів і стійкості.

Розглянуті алгоритми відповідають найвищим сучасним вимогам до постквантових електронних підписів і здатні забезпечити довгострокову безпеку даних у період приходу квантових обчислень, а також задають напрям подальших досліджень і практичних впроваджень у сфері кібербезпеки.

ПЕРЕЛІК ПОСИЛАНЬ

1) Shor P. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer / P. Shor // SIAM Journal on Computing. – 1997. – Т. 26, № 5. – С. 1484–1509.

2) Grover L. A Fast Quantum Mechanical Algorithm for Database Search / L. Grover // Proc. of the 28th ACM Symposium on Theory of Computing (STOC). – 1996. – С. 212–219.

3) Bernstein D., Buchmann J., Dahmen E. (eds.). Post-Quantum Cryptography : Lecture Notes in Computer Science, vol. 5299 / D. Bernstein, J. Buchmann, E. Dahmen (eds.). – Berlin : Springer, 2009. – 248 с.

4) Ajtai M. Generating Hard Instances of Lattice Problems / M. Ajtai // Proc. of the 28th ACM Symposium on Theory of Computing (STOC). – 1996. – С. 99–108.

5) Hoffstein J., Pipher J., Silverman J. H. NTRU : A Ring-Based Public Key Cryptosystem / J. Hoffstein, J. Pipher, J. H. Silverman // Lecture Notes in Computer Science, vol. 1423. – Berlin : Springer, 1998. – С. 267–288.

6) McEliece R. J. A Public-Key Cryptosystem Based on Algebraic Coding Theory / R. J. McEliece // DSN Progress Report. – 1978. – С. 114–116.

7) Lamport L. Constructing Digital Signatures from a One-Way Function / L. Lamport // Technical Report CSL-98, SRI International. – Feb. 1979.

8) Merkle R. C. Protocols for Public Key Cryptosystems / R. C. Merkle // Proc. IEEE Symposium on Security and Privacy. – 1989. – С. 122–134.

9) NIST. Call for Proposals : Post-Quantum Cryptography – Final Public Release [Електронний ресурс] / NIST. – Gaithersburg, MD : NIST CSRC, 19 Dec. 2016. – 28 с. – Режим доступу: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

10) Chen L., Jordan S., Liu Y.-K. та ін. Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process : NIST Interagency Report 8240 [Електронний ресурс] / L. Chen, S. Jordan, Y.-K. Liu та ін. –

Gaithersburg, MD : NIST, Jan. 2019. – 27 с. – Режим доступу: https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8240.pdf.

11) Chen L., Moody D., Regenscheid A. та ін. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process : NIST Interagency Report 8309 [Електронний ресурс] / L. Chen, D. Moody, A. Regenscheid та ін. – Gaithersburg, MD : NIST, July 2020. – 39 с. – Режим доступу: https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf.

12) NIST. Solicitation for additional post-quantum digital signature schemes [Електронний ресурс] / NIST. – Gaithersburg, MD : NIST CSRC, 3 Aug. 2022. – Режим доступу: https://csrc.nist.gov/projects/pqc-dig-sig.

13) NIST. Post-Quantum Cryptography Standardization : Security Strength Categories [Електронний ресурс] / NIST. – Gaithersburg, MD : NIST, 2016. – Режим доступу: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria).

14) Patarin J. The Oil and Vinegar Signature Scheme / J. Patarin // Dagstuhl Workshop on Cryptography. – 1997. – 18 с.

15) Kipnis A., Patarin J., Goubin L. Unbalanced Oil and Vinegar Signature Schemes / A. Kipnis, J. Patarin, L. Goubin // EUROCRYPT ’99 : LNCS, vol. 1592. — Berlin : Springer, 1999. — С. 206–222.

16) NIST. UOV Specification (Round 1) [Електронний ресурс] / NIST. – Gaithersburg, MD : NIST, 2020. – Режим доступу: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/UOV-spec-web.pdf.

17) Kipnis A., Shamir A. Cryptanalysis of the Oil and Vinegar Signature Scheme / A. Kipnis, A. Shamir // CRYPTO ’98 : Lecture Notes in Computer Science, vol. 1462. – Berlin : Springer, 1998. – С. 257–266.

18) Courtois N., Klimov A., Patarin J., Shamir A. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations / N. Courtois,

A. Klimov, J. Patarin, A. Shamir // EUROCRYPT 2000 : LNCS, vol. 1807. – Berlin : Springer, 2000. – С. 392–407.

19) Bardet M. та ін. Improvements of Algebraic Attacks for Solving the Rank Decoding and MinRank Problems / M. Bardet та ін. // ASIACRYPT 2020 : LNCS, vol. 12491. – Berlin : Springer, 2020. – С. 507–536.

20) Beullens W. Breaking Rainbow Takes a Weekend on a Laptop [Електронний ресурс] / W. Beullens. – 2022. – 21 с. – Режим доступу: https://eprint.iacr.org/2022/214.

21) Beullens W. MAYO : Practical Post-Quantum Signatures from Oil-and-Vinegar Maps [Електронний ресурс] / W. Beullens. – 2021. – 17 с. – Режим доступу: https://eprint.iacr.org/2021/1144.

22) Campos F., Celi S., Hess B., Kannwischer M. MAYO Specification (Round-1) [Електронний ресурс] / F. Campos, S. Celi, B. Hess, M. Kannwischer. – 2023. – Режим доступу: https://pqmayo.org/assets/specs/mayo.pdf.

23) Furue H., Ikematsu Y. A New Security Analysis Against MAYO and QR-UOV Using Rectangular MinRank Attack / H. Furue, Y. Ikematsu // Advances in Information and Computer Security – 18th International Workshop on Security (IWSEC 2023) Proceedings, LNCS, vol. 14128. – Yokohama, Japan: Springer, 2023. – С. 101–116. – Режим доступу: https://doi.org/10.1007/978-3-031-41326-1_6.

24) Wang L. C., Tseng P. E., Kuan Y. L., Chou C. Y. A Simple Noncommutative UOV Scheme [Електронний ресурс] / L. C. Wang, P. E. Tseng, Y. L. Kuan, C. Y. Chou. – 2022. – Режим доступу: https://eprint.iacr.org/2022/1742.

25) NIST. SNOVA Specification (Round 2) [Електронний ресурс] / NIST. – Gaithersburg, MD : NIST, Feb. 2025. – Режим доступу: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/snova-spec-round2-web.pdf.

26) Li P., Ding J. Cryptanalysis of the SNOVA Signature Scheme / P. Li, J. Ding // PQCrypto 2024 : LNCS, vol. 14772. – Berlin : Springer, 2024. – С. 79–91.

27) Beullens W. Improved Cryptanalysis of SNOVA / W. Beullens //. – 2024. – 15 с.

28) Cabarcas D., Li P., Verbel J., Villanueva-Polanco R. Improved Attacks for SNOVA by Exploiting Stability under a Group Action / D. Cabarcas, P. Li, J. Verbel, R. Villanueva-Polanco //. – 2024. – 16 с.

29) Nakamura S., Tani Y., Furue H. Lifting approach against the SNOVA scheme / S. Nakamura, Y. Tani, H. Furue. // – 2024. – Режим доступу: https://eprint.iacr.org/2024/1374.

ДОДАТОК А

КОД ОЦІНКИ ЕФЕКТИВНОСТІ UOV ТА SNOVA

```
#include <stdio.h>
#include <stdint.h>
#include <inttypes.h>
#include <stdlib.h>
#include <time.h>

#include "api.h"

#ifndef N_ITERS
#define N_ITERS 2048
#endif

#ifdef __ARM_ARCH
static inline uint64_t get_cycles(void)
{
    struct timespec time;
    clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &time);
    return (int64_t)((time.tv_sec * 1e9 + time.tv_nsec) * 2.4);
}
#else
static inline uint64_t get_cycles() {
    uint32_t lo, hi;
    uint64_t o;
    __asm__ __volatile__("rdtscp" : "=a"(lo), "=d"(hi) : :
"%ecx");
    o = hi;
    o <<= 32;
    return (o | lo);
}
#endif

static inline uint64_t cpucycles_overhead(void) {
    uint64_t t0, t1, overhead = -1LL;
    unsigned int i;

    for (i = 0; i < 100000; i++) {
        t0 = get_cycles();
        __asm__ volatile("");
        t1 = get_cycles();
        if (t1 - t0 < overhead) overhead = t1 - t0;
    }

    return overhead;
}

void analysis(uint64_t *t, uint32_t len, const uint64_t
CPU_CLOCK);
```

```
static int cmp_uint64(const void *a, const void *b) {
    if (*(uint64_t *)a < *(uint64_t *)b) return -1;
    if (*(uint64_t *)a > *(uint64_t *)b) return 1;
    return 0;
}

static uint64_t median(uint64_t *l, size_t len) {
    qsort(l, len, sizeof(uint64_t), cmp_uint64);

    if (len % 2)
        return l[len / 2];
    else
        return (l[len / 2 - 1] + l[len / 2]) / 2;
}

static uint64_t average(uint64_t *t, size_t len) {
    uint64_t acc = 0;
    for (size_t i = 0; i < len; i++) {
        acc += t[i];
    }
    return acc / len;
}

void analysis(uint64_t *t, uint32_t len, const uint64_t CPU_CLOCK)
{
    uint64_t *td = malloc(sizeof(uint64_t) * len);

    for (uint32_t i = 0; i < len; ++i) {
        td[i] = t[i * 2 + 1] - t[i * 2];
    }

    uint64_t md = median(td, len);
    uint64_t avg = average(td, len);
    printf("median cycle:\t%lu cycles/ticks\n", md);
    printf("average cycle:\t%lu cycles/ticks\n", avg);
    printf("median:\t\t%lf   ticks/sec\n",   (double)CPU_CLOCK   /
(double)md);
    printf("average:\t%lf   ticks/sec\n",   (double)CPU_CLOCK   /
(double)avg);

    free(td);
}

double get_cpu_f(void) {
    FILE* fp;
    char buffer[1024];
    double cpu_frequency;

    fp = fopen("/proc/cpuinfo", "r");
    while (fgets(buffer, sizeof(buffer), fp)) {
        if (sscanf(buffer, "cpu MHz : %lf", &cpu_frequency) == 1)
{
            break;
```

```
        }
    }

    printf("CPU Frequency: %.2f MHz\n", cpu_frequency);

    fclose(fp);
    return cpu_frequency * 1000000;
}

int main(void) {
    double CPU_CLOCK = 3800000000;

    printf("CPU Frequency: %.2f MHz\n", CPU_CLOCK / 1000000);
    printf("-----------------------------------------------
\n");
    printf("Algorithm: %s\n", CRYPTO_ALGNAME);
    printf("Public Key size: %d bytes\n", CRYPTO_PUBLICKEYBYTES);
    printf("Secret Key size: %d bytes\n", CRYPTO_SECRETKEYBYTES);
    printf("Signature size: %d bytes\n", CRYPTO_BYTES);
    printf("-----------------------------------------------
\n");

    uint64_t t_keygen[N_ITERS * 2];
    uint64_t t_sign[N_ITERS * 2];
    uint64_t t_verify[N_ITERS * 2];
    unsigned char pk[CRYPTO_PUBLICKEYBYTES];
    unsigned char sk[CRYPTO_SECRETKEYBYTES];
    unsigned long long smlen;

    size_t msglens[] = {64, 1024, 16384, 1048576};
    size_t num_msglens = sizeof(msglens) / sizeof(msglens[0]);

    for (size_t midx = 0; midx < num_msglens; ++midx) {
        size_t mlen = msglens[midx];
        printf("Message size: %zu bytes\n", mlen);
        printf("Public      key      size:      %d      bytes\n",
CRYPTO_PUBLICKEYBYTES);
        printf("Secret      key      size:      %d      bytes\n",
CRYPTO_SECRETKEYBYTES);
        printf("Maximum signed message buffer: %zu bytes\n", mlen
+ CRYPTO_BYTES);

        unsigned char *msg   = malloc(mlen);
        unsigned char *sm    = malloc(mlen + CRYPTO_BYTES);
        unsigned char *m_out = malloc(mlen + CRYPTO_BYTES);
        memset(msg, 0, mlen);

        for (int i = 0; i < N_ITERS; ++i) {
            t_keygen[2*i]     = get_cycles();
            crypto_sign_keypair(pk, sk);
            t_keygen[2*i + 1] = get_cycles();

            t_sign[2*i]       = get_cycles();
```

```
            crypto_sign(sm, &smlen, msg, mlen, sk);
            t_sign[2*i + 1]   = get_cycles();

            t_verify[2*i]     = get_cycles();
            crypto_sign_open(m_out, &smlen, sm, smlen, pk);
            t_verify[2*i + 1] = get_cycles();
        }

        printf("Keypair generation:\n");
        analysis(t_keygen, N_ITERS, CPU_CLOCK);
        printf("-------------------------------------------------
\n");

        printf("Signature generation:\n");
        analysis(t_sign, N_ITERS, CPU_CLOCK);
        printf("-------------------------------------------------
\n");

        printf("Signature verification:\n");
        analysis(t_verify, N_ITERS, CPU_CLOCK);
        printf("-------------------------------------------------
\n");

        free(msg);
        free(sm);
        free(m_out);
    }

    return 0;
}
```

ДОДАТОК Б

КОД ОЦІНКИ ЕФЕКТИВНОСТІ MAYO

```
#define _POSIX_C_SOURCE 200112L
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <inttypes.h>
#include <time.h>
#include <sched.h>
#include <string.h>
//#include "api.h"       // bring in analysis(), get_cpu_clock(),
N_ITERS
#include <mayo.h>

#ifndef MAYO_VARIANT
static const mayo_params_t* params[] = { &MAYO_1, &MAYO_2,
&MAYO_3, &MAYO_5 };
static const char*           names[]  = { "MAYO_1",  "MAYO_2",
"MAYO_3",  "MAYO_5" };
#else
static const mayo_params_t* params[] = { &MAYO_VARIANT };
static const char*          names[]  = { #MAYO_VARIANT };
#endif
static const int bench_cnt = sizeof(params)/sizeof(params[0]);

// Test various message lengths
static size_t msg_lens[] = {64, 1024, 16384, 1048576};
static const int msg_cnt  = sizeof(msg_lens)/sizeof(msg_lens[0]);

#ifndef N_ITERS
#define N_ITERS 2048
#endif

#ifdef __ARM_ARCH
static inline uint64_t get_cycles(void)
{
    struct timespec time;
    clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &time);
    return (int64_t)((time.tv_sec * 1e9 + time.tv_nsec) * 2.4);
}
#else
static inline uint64_t get_cycles() {
    uint32_t lo, hi;
    uint64_t o;
    __asm__ __volatile__("rdtscp" : "=a"(lo), "=d"(hi) : :
"%ecx");
    o = hi;
    o <<= 32;
    return (o | lo);
}
```

```
#endif

static inline uint64_t cpucycles_overhead(void) {
    uint64_t t0, t1, overhead = -1LL;
    unsigned int i;

    for (i = 0; i < 100000; i++) {
        t0 = get_cycles();
        __asm__ volatile("");
        t1 = get_cycles();
        if (t1 - t0 < overhead) overhead = t1 - t0;
    }

    return overhead;
}

void analysis(uint64_t *t, uint32_t len, const uint64_t
CPU_CLOCK);

static int cmp_uint64(const void *a, const void *b) {
    if (*(uint64_t *)a < *(uint64_t *)b) return -1;
    if (*(uint64_t *)a > *(uint64_t *)b) return 1;
    return 0;
}

static uint64_t median(uint64_t *l, size_t len) {
    qsort(l, len, sizeof(uint64_t), cmp_uint64);

    if (len % 2)
        return l[len / 2];
    else
        return (l[len / 2 - 1] + l[len / 2]) / 2;
}

static uint64_t average(uint64_t *t, size_t len) {
    uint64_t acc = 0;
    for (size_t i = 0; i < len; i++) {
        acc += t[i];
    }
    return acc / len;
}

void analysis(uint64_t *t, uint32_t len, const uint64_t CPU_CLOCK)
{
    uint64_t *td = malloc(sizeof(uint64_t) * len);

    for (uint32_t i = 0; i < len; ++i) {
        td[i] = t[i * 2 + 1] - t[i * 2];
    }

    uint64_t md = median(td, len);
    uint64_t avg = average(td, len);
    printf("median cycle:\t%lu cycles/ticks\n", md);
```

```
    printf("average cycle:\t%lu cycles/ticks\n", avg);
    printf("median:\t\t%lf   ticks/sec\n",   (double)CPU_CLOCK   /
(double)md);
    printf("average:\t%lf   ticks/sec\n",   (double)CPU_CLOCK   /
(double)avg);

    free(td);
}

double get_cpu_f(void) {
    FILE* fp;
    char buffer[1024];
    double cpu_frequency;

    fp = fopen("/proc/cpuinfo", "r");
    while (fgets(buffer, sizeof(buffer), fp)) {
        if (sscanf(buffer, "cpu MHz : %lf", &cpu_frequency) == 1)
{
            break;
        }
    }

    printf("CPU Frequency: %.2f MHz\n", cpu_frequency);

    fclose(fp);
    return cpu_frequency * 1000000;
}

int main(void) {
    double CPU_CLOCK = 3800000000;

    // Header
    printf("CPU Frequency: %.2f MHz\n", CPU_CLOCK / 1000000);
    printf("-----------------------------------------------
\n");

    for (int i = 0; i < bench_cnt; i++) {
        const mayo_params_t *p = params[i];
        printf("Algorithm: %s\n", names[i]);
        size_t pk_len  = PARAM_cpk_bytes(p);
        size_t sk_len  = PARAM_csk_bytes(p);
        size_t sig_len = PARAM_sig_bytes(p);
        printf("Public Key size: %zu bytes\n", pk_len);
        printf("Secret Key size: %zu bytes\n", sk_len);
        printf("Signature size: %zu bytes\n", sig_len);
        printf("-----------------------------------------------
\n");

        uint64_t t_keygen[N_ITERS * 2];
        uint64_t t_sign[N_ITERS * 2];
        uint64_t t_verify[N_ITERS * 2];

        for (size_t j = 0; j < msg_cnt; j++) {
```

```
            size_t mlen = msg_lens[j];
            printf("Message size: %zu bytes\n", mlen);
            printf("Maximum signed message buffer: %zu bytes\n",
mlen + sig_len);

            unsigned char *pk   = malloc(pk_len);
            unsigned char *sk   = malloc(sk_len);
            unsigned char *msg  = malloc(mlen);
            unsigned char *sig  = malloc(mlen + sig_len);
            unsigned char *mout = malloc(mlen + sig_len);
            if (!pk || !sk || !msg || !sig || !mout) {
                fprintf(stderr, "Allocation failed\n");
                return 1;
            }
            memset(msg, 0, mlen);

            for (int iter = 0; iter < N_ITERS; iter++) {
                // Keypair generation
                t_keygen[2 * iter] = get_cycles();
                mayo_keypair(p, pk, sk);
                t_keygen[2 * iter + 1] = get_cycles();

                // Signature generation
                t_sign[2 * iter] = get_cycles();
                {
                    size_t smlen;
                    mayo_sign(p, sig, &smlen, msg, mlen, sk);
                }
                t_sign[2 * iter + 1] = get_cycles();

                // Signature verification
                t_verify[2 * iter] = get_cycles();
                mayo_verify(p, mout, mlen, sig, pk);
                t_verify[2 * iter + 1] = get_cycles();
            }

            printf("Keypair generation:\n");
            analysis(t_keygen, N_ITERS, CPU_CLOCK);
            printf("-------------------------------------------
----\n");

            printf("Signature generation:\n");
            analysis(t_sign, N_ITERS, CPU_CLOCK);
            printf("-------------------------------------------
----\n");

            printf("Signature verification:\n");
            analysis(t_verify, N_ITERS, CPU_CLOCK);
            printf("-------------------------------------------
----\n");

            free(pk);
            free(sk);
```

```
            free(msg);
            free(sig);
            free(mout);
        }
    }
    return 0;
}
```