**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Західноукраїнський національний університет**
**Факультет комп'ютерних інформаційних технологій**
Кафедра кібербезпеки

**ШУХМАНН Вадим Александерович**

# Алгоритми цифрового підпису для блокчейн-технологій / Digital signature algorithms for blockchain technologies

спеціальність: 125 – Кібербезпека та захист інформації

освітньо-професійна програма – Кібербезпека

Кваліфікаційна робота

Виконав студент групи
КБм -21
В. А. Шухманн

Науковий керівник
д.т.н., професор В. В.Яцків

Кваліфікаційну роботу
Допущено до захисту:

«____» _____ ___2024 р.

Завідувач кафедри
_____ **В.В.Яцків**

**ТЕРНОПІЛЬ - 2024**

**Факультет комп'ютерних інформаційних технологій**
Кафедра кібербезпеки
Освітній ступінь «магістр»
спеціальність: 125 – Кібербезпека та захист інформації
освітньо-професійна програма – Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ В.В.Яцків
«____» _____ 2023 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

**ШУХМАНН Вадим Александерович**
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
**Алгоритми цифрового підпису для блокчейн-технологій / Digital signature algorithms for blockchain technologies**
керівник роботи д.т.н., професор В.В. Яцків

затверджені наказом по університету від 12 грудня 2023 року № 753

2. Строк подання студентом закінченої кваліфікаційної роботи 12 грудня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
– провести аналіз криптографічних примітивів, які використовуються у блокчейн-технологіях;
– дослідити алгоритми цифрового підпису, які застосовуються для захисту транзакцій;
– провести порівняння алгоритмів цифрового підпису із розширеними схемами, такими як мультипідписи, порогові та агреговані підписи;
– розробити програмну реалізацію агрегованого цифрового підпису для оптимізації процесів верифікації підписів у децентралізованих мережах;
– дослідження характеристики алгоритмів цифрового підпису в блокчейн-системах, включаючи час генерації та перевірки підпису.

5. Перелік графічного матеріалу у роботі:
– схема мультипідпису;

– порогова схема підпису;
− кільцевий підпис;
– схема агрегованого підпису;
– алгоритм агрегованого цифрового підпису;
– результати дослідження постквантових алгоритмів цифрового підпису.

6. Консультанти розділів кваліфікаційної роботи

| | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| | | | |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строки виконання етапів кваліфікаційної роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз видів цифрового підпису в блокчейн | 12.2023 р. – 03.2024 р. | |
| 2 | Алгоритми цифрового підпису в технології блокчейн | 03.2024 р. – 06.2024 р. | |
| 3 | Реалізація та дослідження алгоритмів цифрових підписів в блокчейн | 06.2024 р. – 11.2024 р. | |

Студент _____ Шухманн В.А.
( підпис )

Керівник роботи _____ д.т.н., професор В.В. Яцків
( підпис )

# АНОТАЦІЯ

Кваліфікаційна робота на тему «Алгоритми цифрового підпису для блокчейн-технологій» на здобуття освітнього ступеня «Магістр» зі спеціальності 125 «Кібербезпека та захист інформації» освітньо-професійної програми «Кібербезпека» написана обсягом 67 сторінки і містить 5 ілюстрації, 7 таблиць, 2 додатки та 36 джерел за переліком посилань.

Метою кваліфікаційної роботи є підвищення ефективності алгоритмів цифрових підписів в технології блокчейн.

Методи досліджень. Для розв’язання поставлених задач у даній кваліфікаційній роботі використано: теоретичний аналіз, експериментальні методи, порівняльний аналіз, алгоритмічний синтез, статистичний аналіз.

Результати дослідження: Удосконалено агрегований метод цифрового підпису на основі алгоритму EdDSA, що підвищило швидкість виконання операцій і стійкості до атак завдяки використанню криптографічної кривої Curve25519.

У роботі виконано алгоритмічну та програмну реалізацію агрегованих підписів, які є ефективним рішенням для зменшення розміру блоків у блокчейні. Це забезпечує оптимізацію використання ресурсів і підвищенню масштабованості мережі.

Результати роботи можуть бути використані при розробці інформаційних систем на основі технології блокчейн.

Ключові слова: ЦИФРОВІ ПІДПИСИ, ПОСТКВАНТОВІ АЛГОРИТМИ, БЛОКЧЕЙН, ЦІЛІСНІСТЬ, АВТЕНТИЧНІСТЬ.

# ABSTRACT

Qualification thesis on the topic "Digital Signature Algorithms for Blockchain Technologies" for obtaining the educational degree "Master" in the specialty 125 "Cybersecurity and Information Protection" under the educational-professional program "Cybersecurity" comprises 67 pages and includes 5 illustrations, 7 tables, 2 appendices, and 36 references in the bibliography.

Objective of the Qualification Paper. The objective is to enhance the efficiency of digital signature algorithms in blockchain technology.

Research Methods. To solve the tasks set in this qualification paper, the following methods were applied: theoretical analysis, experimental methods, comparative analysis, algorithmic synthesis, statistical analysis.

Research Results. An improved aggregated digital signature method based on the EdDSA algorithm has been developed, which enhances the operational speed and resistance to attacks by utilizing the Curve25519 cryptographic curve.

The paper includes algorithmic and software implementations of aggregated signatures, which provide an efficient solution for reducing blockchain block sizes. This ensures resource optimization and improves network scalability.

The results of the research can be applied in the development of information systems based on blockchain technology.



**Keywords**: DIGITAL SIGNATURES, POST-QUANTUM ALGORITHMS, BLOCKCHAIN, INTEGRITY, AUTHENTICITY

ЗМІСТ

# ВСТУП

**Актуальність роботи.** Розвиток блокчейн-технологій докорінно змінює підхід до забезпечення безпеки цифрових даних і транзакцій. Центральним елементом цих технологій є алгоритми цифрового підпису, які гарантують автентичність, цілісність і незмінність інформації. Широке впровадження криптовалют, смарт-контрактів і децентралізованих фінансових систем підвищує вимоги до криптографічних методів, зокрема щодо стійкості до нових атак, швидкості виконання та ефективності використання ресурсів.

Актуальність теми зумовлена необхідністю аналізу сучасних алгоритмів цифрового підпису, таких як ECDSA, EdDSA, та новітніх схем, включаючи мультипідписи, порогові й агреговані підписи. Водночас постквантові загрози стимулюють дослідження алгоритмів, стійких до квантових обчислень. Це особливо важливо для забезпечення довготривалої безпеки децентралізованих мереж.

Цифрові підписи відіграють центральну роль у технології Blockchain, оскільки кожна транзакція має бути підписана, щоб бути дійсною. Відповідно, цифрові підписи мають потрійну мету в протоколах блокчейну [1, 2]:

– вони підтверджують право власності та надають дозвіл на використання коштів;

– вони підтверджують неспростовність, тобто доказ авторизації є незаперечним;

– вони доводять, що трансакція не була і не може бути змінена.

Поточні схеми підпису, де один користувач створює підписи для повідомлень, можуть постраждати від потенційної загрози, оскільки це єдина точка збою, яка може зруйнувати схему, якщо зловмисний користувач отримає контроль над секретним ключем.

Вищезазначеної проблеми можна уникнути шляхом впровадження схем мультипідпису або порогових підписів. Обидві схеми досить схожі в тому сенсі, що вони є багатосторонніми протоколами та вимагають мінімум $m$ із $n$

користувачів, щоб підписати файл, щоб підтвердити автентичність. Тим не менш, між ними важливі відмінності.

**Мета і завдання дослідження.** Метою роботи є підвищення ефективності алгоритмів цифрових підписів в технології блокчейн.

Досягнення визначеної мети передбачає вирішення таких завдань:

– провести аналіз криптографічних примітивів, які використовуються у блокчейн-технологіях, з метою визначення їх впливу на безпеку та ефективність систем;

– дослідити алгоритми цифрового підпису, які застосовуються для захисту транзакцій у блокчейн-мережах, таких як ECDSA та EdDSA, з оцінкою їх криптографічної стійкості та продуктивності;

– провести порівняння традиційних алгоритмів цифрового підпису із розширеними схемами, такими як мультипідписи, порогові та агреговані підписи, для оцінки їх придатності до різних сценаріїв використання в блокчейнах;

– розробити програмну реалізацію агрегованого цифрового підпису для оптимізації процесів верифікації підписів у децентралізованих мережах;

– дослідження впливу різних алгоритмів цифрового підпису на продуктивність блокчейн-систем, включаючи час генерації та перевірки підпису, а також використання обчислювальних ресурсів і обсягів пам’яті.

**Об’єкт дослідження** – процеси забезпечення автентичності, цілісності та незмінності транзакцій;

**Предмет дослідження –** алгоритми та схеми цифрового підпису в технології блокчейн.

**Методи досліджень.** Для розв’язання поставлених задач у даній кваліфікаційній роботі використано: теоретичний аналіз, експериментальні методи, порівняльний аналіз, алгоритмічний синтез, статистичний аналіз.

**Наукова новизна одержаних результатів.** Удосконалено агрегований метод цифрового підпису на основі алгоритму EdDSA, що підвищило

швидкість виконання операцій і стійкості до атак завдяки використанню криптографічної кривої Curve25519.

**Практичне значення отриманих результатів.** У роботі виконано алгоритмічну та програмну реалізацію агрегованих підписів, які є ефективним рішенням для зменшення розміру блоків у блокчейні. Це забезпечує оптимізацію використання ресурсів і підвищенню масштабованості мережі.

**Публікації та апробація КР**.

1. Шухманн В.А., Яцків Н. Г. Алгоритми цифрового підпису для захисту транзакцій в мережі біткоїн. Матеріали XVI Всеукраїнська науково-практична конференція “Актуальні проблеми комп’ютерних наук (АПКН – 2024)”, м.Хмельницький, ХНУ, 15-16 листопада 2024. – С. 559-561.

2. Шухманн В., Смирнов Д., Кондратюк В. Порогова схема цифрового підпису для захисту анонімності в блокчейні. Матеріали проблемно-наукової міжгалузевої конференції «Кібербезпека та комп’ютерно-інтегровані технології» (КБКІТ - 2024), Тернопіль, 2024. – С. 78-81.

# 1 АНАЛІЗ ВИДІВ ЦИФРОВОГО ПІДПИСУ В БЛОКЧЕЙН

## 1.1 Криптографічні примітиви в блокчейні

Криптографічні примітиви є основою, на якій побудовано структуру технології блокчейн. Ці фундаментальні алгоритми та протоколи є основою, яка забезпечує безпеку, автентичність і довіру в екосистемі блокчейну. Розглянемо криптографічні примітиви та дослідимо їх роль у технології блокчейн.

Криптографічні примітиви – це базові алгоритми, які лежать в основі більш складних криптографічних систем. У технології блокчейн ці примітиви використовуються для захисту даних, автентифікації транзакцій і підтримки цілісності всієї мережі.

До основних примітивів в блокчейні необхідно віднести [3, 4].

Хеш-функції: такі функції, як SHA-256, перетворюють будь-які вхідні дані в рядок фіксованого розміру, який виглядає випадковим. Хеш функції використовуються для створення унікального відбитка даних, забезпечення цілісності та можливості зв’язування блоків у блокчейні.

Криптографія з відкритим ключем: передбачає використання пари ключів (відкритого та закритого) для безпечного зв’язку. Відкритий ключ відкритий, а закритий ключ зберігається в таємниці.

Цифрові підписи: такі методи, як ECDSA (алгоритм цифрового підпису на основі еліптичних кривих), дозволяють користувачам підписувати цифрові транзакції, підтверджуючи право власності та автентичність без розкриття закритого ключа.

Криптографічні хеш-функції: вони використовуються в процесі майнінгу, особливо в блокчейнах Proof of Work (PoW), де майнери вирішують складні математичні головоломки на основі хеш-функцій.

Криптографічні примітиви у блокчейні забезпечують [1, 5]:

1) захист транзакцій:

– кожна транзакція в блокчейні захищена за допомогою криптографічних методів, що гарантує, що тільки законний власник може ініціювати транзакції;

2) підтримка цілісності ланцюга:

– хеш-функції використовуються для створення унікального хешу кожного блоку, пов’язуючи його з попереднім блоком і таким чином зберігаючи цілісність блокчейну;

3) підтримка децентралізації:

– криптографічні примітиви мають вирішальне значення для децентралізації, дозволяючи безпечну та надійну взаємодію між сторонами, яким не потрібно довіряти одна одній, але довіряти алгоритму.

4) перспективний блокчейн:

– у міру розвитку криптографії технологія блокчейн адаптується, інтегруючи більш просунуті криптографічні методи для підвищення безпеки та ефективності.

Криптографічні примітиви є суттю, яка вливає на довіру та надійність у технології блокчейн. Розуміння цих примітивів має вирішальне значення для розробників. По мірі просування вперед еволюція криптографічних примітивів продовжуватиме формувати ландшафт блокчейну, створюючи більш безпечні, ефективні та стійкі системи. У цифрову епоху криптографічні примітиви є інструментами безпеки заснованими на довірі.

## 1.2 Алгоритми цифрового підпису для захисту транзакцій в мережі біткоїн

Цифрові підписи є важливим елементом безпеки в блокчейн-технологіях, оскільки вони забезпечують надійний захист транзакцій, аутентифікацію користувачів і захист від шахрайства. Їх актуальність обумовлена такими ключовими причинами [2, 6]:

1) забезпечення аутентифікації та авторизації**.** Цифровий підпис дозволяє підтвердити, що транзакція була ініційована дійсним власником криптовалютного гаманця. Усі транзакції в блокчейні підписуються особистим (приватним) ключем користувача, що гарантує, що жодна стороння особа не може відправити транзакцію від імені власника без доступу до цього ключа;

2) цілісність даних. Цифрові підписи дозволяють верифікувати цілісність транзакцій, забезпечуючи захист від змін після підписання. Будь-які модифікації даних призведуть до недійсності підпису, і транзакція буде відхилена, що гарантує точність інформації в блокчейні;

3) захист від несанкціонованих змін та шахрайства**.** Цифрові підписи знижують ризик несанкціонованих змін транзакцій, забезпечуючи криптографічний захист від підробки. Це особливо важливо у випадку великих фінансових транзакцій, коли необхідно захистити активи від шахрайства чи кібератак;

4) масштабованість і зниження витрат**.** Сучасні алгоритми цифрового підпису, такі як Шнорра, дозволяють об’єднувати кілька підписів у один. Це зменшує розмір транзакцій у блокчейні, оптимізує використання пам’яті, знижує комісії та підвищує ефективність, що особливо актуально для блокчейнів із великою кількістю користувачів;

5) Підвищення конфіденційності та гнучкості. Деякі алгоритми цифрового підпису, як-от мультипідпис (multisig) та порогові підписи, дозволяють гнучко розподіляти контроль над активами, що є важливим для спільного управління активами. Завдяки мультипідпису або підписам Шнорра транзакції можна налаштувати так, щоб забезпечити конфіденційність і підтвердження від кількох учасників;

6) довіра до децентралізованої мережі. Цифрові підписи створюють криптографічно захищене середовище, де користувачі можуть довіряти один одному без потреби в централізованому управлінні. Завдяки цифровим

підписам кожен вузол мережі може верифікувати транзакції, що підтримує чесність і прозорість системи.

В криптовалютах використовують різні види цифрових підписів, зокрема: мультипідписи, порогові, агреговані та кільцеві підписи [3].

На практиці застосовують також різні алгоритми цифрового підпису, такі як RSA, DSA (Digital Signature Algorithm) і схеми цифрового підпису на основі ECDSA (Elliptic Curve Digital Signature Algorithm) та EdDSA (Edwards-curve Digital Signature Algorithm). RSA є найпоширенішим, однак із розвитком ECC схеми на основі ECDSA стали досить популярними. Це має переваги використання в блокчейнах, оскільки ECC забезпечує той самий рівень безпеки, що й RSA, але займає менше місця. Крім того, генерація ключів відбувається набагато швидше в ECC порівняно з RSA, тому це сприяє загальній продуктивності блокчейну.

У мережі Bitcoin основним алгоритмом цифрового підпису є ECDSA. Зокрема, він використовує еліптичну криву secp256k1. ECDSA є стандартним методом цифрового підпису в мережі Bitcoin, забезпечуючи: аутентифікацію відправника; цілісність транзакцій; захист від несанкціонованих змін транзакцій.

З 2021 року в мережу Bitcoin було введено оновлення Taproot**,** яке додало підтримку підписів Шнорра. Це стало можливим завдяки активації BIP-340 (Bitcoin Improvement Proposal), який офіційно впровадив підписи Шнорра для транзакцій у Bitcoin [7, 8].

До особливостей підписів Шнорра у Bitcoin необхідно віднести:

– агрегація підписів. Підпис Шнорра дозволяє об’єднувати кілька підписів у один, що зменшує розмір транзакцій;

– конфіденційність. Завдяки можливості агрегування, підписи Шнорра підвищують конфіденційність, оскільки мультипідписні транзакції виглядають як звичайні, без явного вказування кількості учасників;

– оптимізація продуктивності. Підпис Шнорра підвищує ефективність обробки та перевірки підписів.

Таким чином, в даний час в Bitcoin використовуються: ECDSA для традиційних транзакцій та підписи Шнорра для транзакцій з Taproot, які забезпечують кращу масштабованість та конфіденційність.

При цьому мультипідписи мають багато ширше застосування в технології блокчейн, зокрема [2, 9, 10]:

1) спільні гаманці. Для компаній, які мають спільні фонди, мультипідписний гаманець забезпечує безпечний доступ до коштів. Наприклад, благодійний фонд може налаштувати гаманець "2 з 3", щоб витрачати кошти лише за наявності підписів двох членів ради;

2) смарт-контракти. В блокчейнах із підтримкою смарт-контрактів (наприклад, Ethereum) мультипідпис використовується для управління контрактами, які потребують схвалення кількох сторін. Це дозволяє реалізувати складніші сценарії, де витрачання коштів вимагає консенсусу;

3) забезпечення конфіденційності та контролю в DeFi. Мультипідпис часто використовується в децентралізованих фінансах (DeFi) для створення безпечних та спільних скарбниць (treasuries). Це дозволяє спільно керувати фінансами, одночасно забезпечуючи прозорість і контроль на рівні платформи;

4) децентралізовані автономні організації (DAO). DAO використовують мультипідпис для управління спільними активами. Наприклад, для затвердження пропозицій і витрачання коштів потрібен підпис кількох уповноважених учасників DAO, що забезпечує консенсус і захист від односторонніх рішень;

5) управління особистими фінансами для підвищення безпеки. Користувачі можуть застосовувати мультипідпис для власних гаманців для підвищення особистої безпеки. Наприклад, зберігати свої кошти в гаманці з мультипідписом з налаштуванням "2 з 3", де ключі розміщені на різних пристроях (наприклад, телефон, ноутбук, апаратний гаманець);

6) запобігання одностороннім рішенням у командній роботі. Наприклад, у стартапі, де кілька співзасновників мають спільний контроль над капіталом,

можна використовувати мультипідпис, щоб ухвалювати фінансові рішення лише за згодою кількох співвласників.

Отже, мультипідпис це важливий інструмент в екосистемі блокчейну для забезпечення спільного управління активами та безпечної взаємодії кількох користувачів з коштами. Завдяки можливості налаштування різних схем підпису, він забезпечує гнучкість і адаптивність до вимог різних користувачів.

Загалом, цифрові підписи є фундаментальним компонентом у захисті цифрових транзакцій, забезпечуючи надійний рівень безпеки, довіри та ефективності в технології блокчейн.

Цифрові підписи забезпечують зв’язок повідомлення з об’єктом, від якого надійшло повідомлення. Цифрові підписи використовуються для автентифікації джерела даних і неспростування [11, 12].

У таблиці 1.1 показано, що ECC може забезпечити такий самий рівень криптографічної стійкості, як і система на основі RSA з меншим розміром ключа.

Таблиця 1.1 – Розмір ключа для підпису RSA та еліптичної кривої

| Розмір ключа RSA (біт) | Розмір ключа еліптичної кривої (біт) |
|---|---|
| 1,024 | 160 |
| 2,048 | 224 |
| 3,072 | 256 |
| 7,680 | 384 |
| 15,360 | 521 |

На практиці використовуються різні схеми, такі як RSA, DSA і схеми цифрового підпису на основі ECDSA. RSA є найпоширенішим; однак із поширенням ECC схеми на основі ECDSA стали досить популярними. Це має переваги використання в блокчейнах, оскільки ECC забезпечує той самий рівень безпеки, що й RSA, але займає менше місця. Крім того, генерація

ключів відбувається набагато швидше в ECC порівняно з RSA, тому це сприяє загальній продуктивності блокчейну.

### 1.2.1 Цифровий підпис ECDSA

Алгоритм цифрового підпису ECDSA є однією з найпопулярніших схем цифрового підпису, що базується на криптографії з використанням еліптичних кривих. Цей алгоритм забезпечує автентичність, цілісність і незаперечність цифрових даних.

Основні етапи алгоритму ECDSA [13, 14]:

1 Генерація ключів:

– Вибір еліптичної кривої та її параметрів, таких як a, b, порядок точки генератора G та модуль p.

– Генерація випадкового приватного ключа k у межах від 1 до n−1, де n – порядок точки G.

Обчислення публічного ключа $Q$ як $Q = kG$, де $G$ – точка генератора на еліптичній кривій.

2. Підписання повідомлення:

Обчислення хешу повідомлення m за допомогою криптографічної хеш-функції (наприклад, SHA-256).

Вибір випадкового числа $k$ для кожного підпису, яке не повинно повторюватися.

Обчислення координати точки на кривій: $R = k\ G$, а потім взяття його $x$ – компоненти для отримання

$$r = R.x\ (mod\ n).$$

Обчислення значення $s$ за формулою:

$$s = k^{-1}(h + d\ r)\ mod\ \text{n},$$

де $h$ – хеш повідомлення, а $d$ – приватний ключ.

Перевірка підпису:

Отримання публічного ключа Q та перевірка, що він є дійсною точкою на еліптичній кривій.

Обчислення значення:

$$w = s^{-1}\ mod\ n.$$

Обчислення:

$$u_1 = h\ w\ (mod\ n)\ ,$$

$$u_2 = r\ w\ (mod\ n).$$

Обчислення точки на кривій:

$$P = u_1 G + u_2 Q.$$

Витягування x-компоненти точки $P$:

$$R' = P.x(mod\ n).$$

Підпис вважається дійсним, якщо

$$R' = r.$$

Серед переваг ECDSA необхідно виділити: безпеку, ефективність та широке застосування [15, 16].

Висока безпека. ECDSA забезпечує високий рівень безпеки при коротших довжинах ключів у порівнянні з іншими алгоритмами, такими як RSA.

Ефективність. Менші розміри підписів і швидше виконання операцій роблять ECDSA ефективним для використання в обмежених середовищах.

Широке застосування. Використовується в різних сферах, включаючи криптовалюти (Bitcoin, Ethereum), протоколи безпеки (TLS/SSL) та цифрові сертифікати.

До недоліки ECDSA слід віднести: управління ключами, вразливість до атак.

Управління ключами. Необхідність у належному управлінні приватними ключами для запобігання компрометації.

Вразливість до атак. Неправильна реалізація (наприклад, використання статичного значення для випадкового числа k) може призвести до витоку приватного ключа.

Отже, ECDSA є потужним і ефективним алгоритмом цифрового підпису, який забезпечує високу безпеку та швидкість. Розуміння його роботи є важливим для впровадження безпечних систем аутентифікації та захисту даних у сучасному цифровому середовищі.

Для обчислення R, ми використовуємо формули для множення точки на кривій. Для спрощення, використаємо малі числа та спрощені параметри.

Параметри алгоритму.

Еліптична крива. Використовуємо просту еліптичну криву, задану рівнянням:

$$y^2 = x^3 + ax + b \ (mod\ p),$$

де $p = 17, a = 2, b = 2$.

Точка генератора. Нехай точка генератора $G = (5, 1)$.

Припустимо, що порядок точки $n = 19$.

Тоді, нехай приватний ключ d = 7.

Нехай повідомлення, яке ми плануємо підписати, буде “Vadym”.

Алгоритм підписання складається з наступних кроків.

1. Обчислення хешу повідомлення. Для спрощення, нехай хеш повідомлення $h = 13$ (використаємо просту хеш-функцію).

2. Вибираємо випадкового числа $k$. Нехай k=10 (це число має бути випадковим і не повторюватися).

3. Обчислюємо координати точки $R = k\ G$:

Для обчислення R, використовуємо формули для множення точки на кривій.

Нехай після обчислень ми отримуємо точку $R = (6, 3)$.

4. Обчислюємо $r = R.x(mod\ n) = 6.$

Обчислення значення $s$:

Обчислюємо обернене значення $k^{-1}(mod\ n)$:

Припустимо, що $k^{-1} = 12$ (можна знайти за допомогою розширеного алгоритму Евкліда).

Отже,

$$s = k^{-1}(h + dr) mod\ n$$

Підставляємо значення:

$$s = 12(13 + 7 \cdot 6) mod 19 = 17\ ,$$

Отже, підпис на повідомленні "Vadym" буде:

$$(r, s) = (6, 17).$$

Перевірка підпису.

Обчислюємо $w$:

$$w = s^{-1}(mod\ n).$$

Припустимо, що $s^{-1} = 11$ (знову ж таки, можна знайти за допомогою розширеного алгоритму Евкліда).

Обчислення $u_1$ і $u_2$:

$$u_1 = h\ w(mod\ n).$$

Підставляємо значення:

$$u_1 = 13 \cdot 11(mod\ 19) = 10.$$

Тепер обчислюємо $u_2$:

$$u_2 = r \cdot w\ (mod\ n).$$

Підставляємо значення:

$$u_2 = 6 \cdot 11(mod\ n) = 9.$$

Обчислюємо точку на кривій:

$$P = u_1\ G + u_2\ Q.$$

Перевірка.

Знаходимо $x$ – компоненту точки $P$.

Перевіряємо, чи дорівнює $x$ – компонента $r$.

Таким чином, ми пройшли через всі етапи підписання та перевірки підпису в ECDSA на простих числах. У реальних сценаріях використовуються більші числа та складніші еліптичні криві для забезпечення безпеки.

### 1.2.2 Цифровий підпис EdDSA

EdDSA – це алгоритм цифрового підпису, який базується на еліптичних кривих, зокрема на кривих у формі Едвардса. Він розроблений для забезпечення високої продуктивності та необхідного рівня безпеки. Основними варіантами EdDSA є Ed25519 і Ed448, які використовують 255-бітну та 448-бітну криві відповідно.

Основні етапи алгоритму EdDSA [17, 18].

1. Генерація ключів.

Вибір еліптичної кривої (наприклад, Ed25519).

Генерація приватного ключа як випадкового 32-байтового числа.

Обчислення публічного ключа як $Q = dG$, де $d$ – приватний ключ, $G$ – точка генератора на кривій.

2. Підписання повідомлення:

Обчислення хешу повідомлення $m$ за допомогою криптографічної хеш-функції (наприклад, SHA-512).

Генерація випадкового числа $r$ (один раз для кожного підпису).

Обчислення точки на кривій: $R = r\,G$.

Обчислення значення $h = H(R \,||\, Q \,||\, m)$, де $H$ – хеш-функція, а "||" означає конкатенацію.

Обчислення значення підпису:

$$s = r + d\,h\ (mod\ L),$$

де $L$ – порядок групи.

3. Перевірка підпису.

Отримання публічного ключа Q і підпису (R, s).

Обчислення хешу:

$$h' = H(R \,||\, Q \,||\, m).$$

Обчислення точки:

$$V = s^{-1} \cdot G + h'^{-1}\,Q.$$

Перевірка. Підпис вважається дійсним, якщо координата $x$ точки $V$ дорівнює $x$ – компоненті точки $R$.

До переваг EdDSA можна віднести високу швидкість, так як EdDSA зазвичай швидший за ECDSA завдяки оптимізації обчислень. Також алгоритм EdDSA має простішу структуру, що знижує ймовірність помилок у реалізації.

Використання детермінованого випадкового числа для підпису зменшує ризик компрометації приватного ключа.

Серед недоліків EdDSA є обмежена підтримка в деяких системах. Хоча EdDSA стає все більш популярним, ще не всі системи його підтримують.

EdDSA є ефективним алгоритмом цифрового підпису, який забезпечує високий рівень безпеки та швидкості. Його використання рекомендоване для сучасних криптографічних застосувань завдяки простоті реалізації та стійкості до атак.

Розглянемо приклад підпису та перевірки цифрового підпису на основі EdDSA з використанням бібліотеки ed25519.

Крок 1. Встановлення бібліотеки.

Спочатку потрібно встановити бібліотеку ed25519. Встановити програму можна зробити за допомогою команди: pip install ed25519.

Крок 2: Генерація ключів.

Згенеруємо пару ключів (приватний і публічний) використовуючи мову Python:

```
import ed25519
# Генерація приватного та публічного ключа
privKey, pubKey = ed25519.create_keypair()
print("Приватний ключ:", privKey.to_ascii(encoding='hex'))
print("Публічний ключ:", pubKey.to_ascii(encoding='hex'))
```

Крок 3. Підписання повідомлення.

```
# Повідомлення, яке потрібно підписати
msg = b “Message for Ed25519 signing”
# Підписування повідомлення
signature = privKey.sign(msg, encoding= “hex”)
print("Підпис (64 байти):", signature)
```

Крок 4. Перевірка підпису.

Перевіримо, чи дійсний підпис для даного повідомлення.

```
try:
    # Перевірка підпису
    pubKey.verify(signature, msg, encoding='hex')
    print("Підпис дійсний.")
except ed25519.BadSignatureError:
    print("Недійсний підпис!")
```

Після виконання коду отримаєте наступні значення:

Приватний ключ (32 байти):

b'1498b5467a63dffa2dc9d9e069caf075d16fc33fdd4c3b01bfadae6433767d93'

Публічний ключ (32 байти):

b'b7a3c12dc0c8c748ab07525b701122b88bd78f600c76342d27f25e5f92444cde'

Підпис (64 байти):

b'6dd355667fae4eb43c6e0ab92e870edb2de0a88cae12dbd8591507f584fe4912babff497f1b8edf9567d2483d54ddc6459bea7855281b7a246a609e3001a4e08'

Підпис дійсний.

## 1.3 Порівняння алгоритмів цифрового підпису ECDSA та EdDSA

При порівнянні алгоритмів цифрового підпису ECDSA та EdDSA можна врахувати кілька параметрів для оцінки їхніх особливостей, безпеки та продуктивності, зокрема [19, 20]:

1. Тип кривої.

ECDSA. Використовує еліптичні криві у формі Вейєрштрасса. Ця форма широко використовується, але може мати певні вразливості, якщо не реалізована правильно.

EdDSA. Використовує скручені криві Едвардса, які розроблені для забезпечення кращої продуктивності та функцій безпеки, включаючи стійкість до певних типів атак.

2. Безпека.

ECDSA. Загалом вважається безпечним, але його безпека сильно залежить від міцності генератора випадкових чисел, що використовується під

час генерації ключів і підпису. Якщо випадкове число (nonce) повторюється або передбачуване, це може призвести до розкриття приватного ключа.

EdDSA. Забезпечує більш високий рівень безпеки завдяки детермінованій генерації nonce, яка виводиться з повідомлення та приватного ключа. Це зменшує ризики, пов'язані з повторним використанням nonce, і підвищує загальну безпеку від бічних каналів атак.

3. Генерація ключів.

ECDSA. Передбачає вибір випадкового приватного ключа та обчислення відповідного публічного ключа, що може бути обчислювально інтенсивним.

EdDSA. Процес генерації ключів є більш ефективним, що дозволяє швидше обчислювати публічні ключі безпосередньо з приватних ключів без додаткових кроків.

4. Алгоритм генерації підпису.

ECDSA. Процес генерації підпису включає кілька кроків, включаючи генерацію випадкового числа, що може бути обчислювально дорогим.

EdDSA. Генерація підпису, як правило, швидша і вимагає менше обчислювальних ресурсів завдяки спрощеному процесу.

5. Використання хеш-функції.

ECDSA. Вимагає додаткової хеш-функції для хешування повідомлення перед підписом. Це додає складність і потенційні вразливості, пов'язані з вибором хеш-функції.

EdDSA. Включає повідомлення безпосередньо в процес підпису без необхідності окремої хеш-функції, спрощуючи реалізацію і зменшуючи можливі проблеми з безпекою.

6. Розміри ключів.

ECDSA. Зазвичай вимагає більших розмірів ключів для еквівалентних рівнів безпеки порівняно з EdDSA. Наприклад, ECDSA може використовувати 256-бітний ключ для схожої безпеки, як менший ключ EdDSA.

EdDSA. Зазвичай забезпечує еквівалентну безпеку з меншими розмірами ключів (наприклад, Ed25519 використовує 256-бітний ключ), що робить його більш ефективним з точки зору зберігання та пропускної здатності.

7. Продуктивність.

ECDSA. Хоча ефективний, він може не відповідати швидкості EdDSA в практичних застосуваннях, особливо в умовах високого навантаження.

EdDSA. Відомий високою продуктивністю на різних платформах, що робить його придатним для ресурсно обмежених середовищ.

8. Складність реалізації.

ECDSA. Більш складний через свою залежність від генерації випадкових чисел і додаткових кроків у створенні підпису.

EdDSA. Простіша реалізація завдяки детермінованій генерації nonce і меншій кількості обчислювальних кроків (таблиця 1.2).

Таблиця 1.2 – Порівняння ECDSA та EdDSA

| Параметр | ECDSA | EdDSA |
|---|---|---|
| Тип кривої | Форма Вейерштрасса | Криві Едвардса |
| Рівень безпеки | Вимагає більших ключів для еквівалентної безпеки (наприклад, 256 біт для 128-бітної безпеки) | Забезпечує еквівалентну безпеку з меншими ключами (наприклад, 256 біт для 128-бітної безпеки) |
| Розмір ключа | Зазвичай 256 біт для secp256k1 | Зазвичай 256 біт для Ed25519 |
| Розмір підпису | 64 байти (для 256-бітних ключів) | 64 байти (для Ed25519) |

Продовження таблиці 1.2.

| Параметр | ECDSA | EdDSA |
|---|---|---|
| Швидкість генерації підпису | Повільніша через кілька етапів і генерацію випадкового числа | Швидша, оскільки не вимагає генерації випадкового числа |
| Швидкість перевірки підпису | Зазвичай повільніша в порівнянні з EdDSA | Швидша завдяки меншій кількості обчислень |
| Генерація нонсу | Генерується випадковим чином (вразливість при повторному використанні) | Детермінована, похідна від повідомлення та приватного ключа |
| Стійкість до атак через бічні канали | Вразливий, якщо не реалізований належним чином | Має вбудований захист від атак на час і атак через бічні канали |
| Вимога до хеш-функції | Вимагає додаткової хеш-функції | Інтегрує повідомлення безпосередньо в процес підпису, зменшуючи складність |
| Складність реалізації | Більш складна через випадковість і кілька етапів | Простішою реалізацією з детермінованими підписами |

Отже, хоча ECDSA і EdDSA є ефективними алгоритмами цифрового підпису на основі еліптичної криптографії, EdDSA, як правило, пропонує переваги з точки зору безпеки, продуктивності та простоти реалізації. Вибір між ними повинен враховувати конкретні випадки використання, вимоги до безпеки та потенційні вразливості, пов'язані з конструкцією та реалізацією кожного алгоритму.

# 2 АЛГОРИТМИ ЦИФРОВОГО ПІДПИСУ В ТЕХНОЛОГІЇ БЛОКЧЕЙН

## 2.1 Осліплений підпис

Сліпі підписи – це розширення цифрових підписів, які забезпечують конфіденційність, дозволяючи користувачеві отримати підпис від підписувача на повідомленні, не маючи можливості побачити вміст закритого повідомлення. Якщо підписувачу пізніше буде представлено підписаний документ, він не зможе пов’язати його ні з сеансом підписання, ні з користувачем, від імені якого він підписав повідомлення [21].

Схему сліпого підпису винайшов Девід Чаум у 1982 році. Вона заснована на схемах цифрового підпису з відкритим ключем, наприклад RSA. Це досягається маскуванням або засліпленням повідомлення перед його підписанням, звідси й назва сліпих підписів. Потім цей сліпий підпис можна перевірити, як і звичайний цифровий підпис. Сліпі підписи були запроваджені як механізм, що дозволяє розробку схем цифрових готівкових грошей. Зокрема, сліпий підпис може надавати послуги незв’язності та анонімності в розподіленій системі, такій як блокчейн.

Схема сліпого підпису також використовується для створення Blindcoin, який використовується в протоколі Mixcoin для Bitcoin, щоб приховати адреси користувачів від служби змішування. Протокол Mixcoin дозволяє анонімні платежі в мережі біткойн, але адреси користувачів все ще видимі для служби змішування. Blindcoin – це протокол, який усуває це обмеження.

Алгоритм шифрування RSA можна використати для створення цифрового підпису $(s)$ повідомлення $(m)$, який обчислюється з використанням секретного ключа за формулою [22]:

$$s = m^d (mod\ n).$$

Перевірка правильності цифрового підпису здійснюється за формулою:

$$m = s^e (mod\ n).$$

Використовуючи сліпий підпис Боб може підписати повідомлення, не знаючи зміст цього повідомлення.

Метод сліпого підпису використовує шифрування RSA, де Боб створює ключі RSA зазначеним вище способом, тобто він вибирає два прості числа ($p$ і $q$) і значення відкритого ключа шифрування ($e = 65537$), а потім обчислює:

$$n = p * q,$$

$$\varphi = (p - 1)(q - 1),$$

$$d = e^{-1} (mod\ \varphi),$$

відповідно, $(n,\ e)$ – відкритий ключ Боба, $(n,\ d)$ – закритий ключ Боба.

Тепер Аліса хоче, щоб Боб сліпо підписав її повідомлення $(m)$. Для цього Боб надсилає Алісі свій відкритий ключ $(n,\ e)$.

Спочатку вона генерує випадкове значення $(k)$, а потім обчислює:

$$m' = m \cdot k^e (mod\ n)$$

Аліса надсилає це Бобу, який використовує свій закритий ключ $(d)$ для обчислення:

$$s' = m'^{\,d} (mod\ n)$$

Боб надсилає це назад, і Аліса обчислює справжній підпис Боба:

$$s = k^{-1}\ s' (mod\ n)$$

Підпис, яким Боб підписав повідомлення, відповідає підпису, якщо б Боб використав свій закритий ключ:

$$s = m^d (mod\ n).$$

Це працює тому, що:

$$s = k^{-1} \cdot s' = k^{-1} \cdot m'^d = k^{-1} \cdot (m\, k^e)^d =$$
$$= k^{-1} \cdot m^d \cdot k^{ed} = k^{-1} \cdot m^d \cdot k^1 = m^d (mod\ n).$$

Розглянемо приклад. Нехай $p = 101$, $q = 197$, $e = 65537$,

$n = 101 * 197 = 19897$,

$\varphi(n) = (101 - 1)(197 - 1) = 19600$,

$d = 65537^{-1} (mod\ 19600) = 13473$.

Нехай повідомлення Аліси $m = 125$, випадкове число $k = 37183$, $k^{-1} (mod\ n) = 12147$

Тоді:

$m' = 125 *\ 37183^{65537} (mod\ 19897) = 16878$,

$s' = 16878^{13473} (mod\ 19897) = 2979$.

Підпис Боба (осліплений):

$s = (12147 * 2979) mod\ 19897 = 13167$.

Підпис повідомлення Бобом з використанням закритого ключа:

$$s = 125^{13473} (mod\ 19897) = 13167.$$

Отже, підпис, яким Боб підписав повідомлення, відповідає підпису, з використанням його закритого ключа.

## 2.2 Мультипідпис

У схемі мультипідпису група об’єктів підписує одне повідомлення. Іншими словами, для підпису одного повідомлення використовується кілька унікальних ключів, які належать їхнім власникам [23, 24].

У літературі мультипідписи також іноді називають багатосторонніми підписами. У реалізаціях блокчейну мультипідписи надають можливість дозволяти підписувати транзакції кільком користувачам, що призводить до підвищення безпеки. Це також називається multi-sig і реалізовано в Bitcoin. Ці схеми можна використовувати таким чином, щоб можна було встановити вимогу щодо кількості підписів для авторизації транзакцій. Наприклад, мультипідпис 1 із 2 може представляти спільний рахунок, де один із власників спільного рахунку має авторизувати транзакцію, підписавши її.

В іншому варіанті, наприклад, мультипідпис 2 з 2 можна використовувати в сценарії, де для підписання транзакції потрібні підписи обох власників спільних рахунків. Це поняття також узагальнюється як m з n підписів, де $m$ – мінімальна кількість необхідних підписів, а $n$ – загальна кількість підписів.

Мультипідпис (multisig) означає необхідність використання кількох ключів для авторизації транзакції Bitcoin, а не одного підпису з використанням одного ключа.

Мультипідпис має ряд застосувань:

– розподіл відповідальності за володіння біткойнами між кількома користувачами;

– уникнення єдиної точки відмови, що значно ускладнює компрометацію гаманця;

– резервне копіювання, коли втрата одного початкового коду не призводить до втрати гаманця.

Концепцію мультипідпису можна візуалізувати за допомогою діаграми (рисунок 2.1).

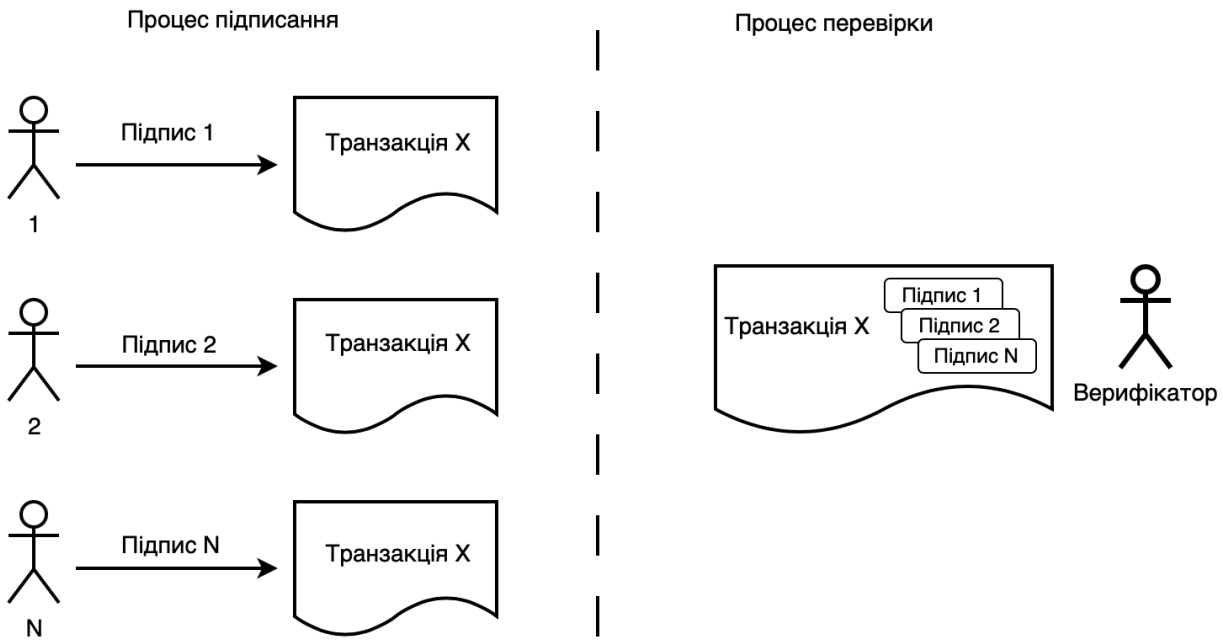


Рисунок 2.1 – Схема мультипідпису

На попередній діаграмі ліворуч показано процес підписання, де m – це кількість різних користувачів, а наявність m унікальних підписів підписує одну транзакцію. Коли верифікатор отримує його, усі підписи в ньому потрібно перевірити окремо.

Розглянемо приклад підпису Боб і Віктором повідомлення для Аліси. За допомогою методу підпису Шнорра можна просто виконати операцію додавання. Завдяки цьому Боб згенерує свій закритий ключ ($sk_1$) і отримає свій відкритий ключ із:

$$P_1 = sk_1 \cdot G.$$

Тепер обчислюємо:

$$R_1 = k_1 \cdot G,$$

$$s_1 = k_1 + H(P||M||R) \cdot sk_1,$$

де $H()$ – хеш пов'язаних байтових значень, а $(P||M||R)$ – це додані байтові значення з $M$ і $R$.

Загальний відкритий ключ $(P)$ буде обчислено шляхом додавання значень Боба та Віктора відкритий ключ. Віктор згенерує свій закритий ключ $(sk_2)$ і отримає відкритий ключ із:

$$P_2 = sk_2 \cdot G.$$

Потім згенеруємо:

$$R_2 = k_2 \cdot G,$$
$$s_2 = k_2 + H(P||M||R) \cdot sk_2,$$

і це буде відкритий ключ для підпису повідомлення:

$$P = P_1 + P_2.$$

Потім Боб надсилає повідомлення $(M)$ і підпис $(R, s)$ Алісі.

Підпис обчислюємо:

$$R = R_1 + R_2,$$

$$s = s_1 + s_2.$$

Коли Аліса отримала підпис $(R, s)$ і повідомлення $(M)$, вона перевіряє:

$$R + H\big(P\big||M\big||R\big) * P = sG.$$

Якщо вони збігаються, підпис вважається перевіреним (рисунок 2.2 ).

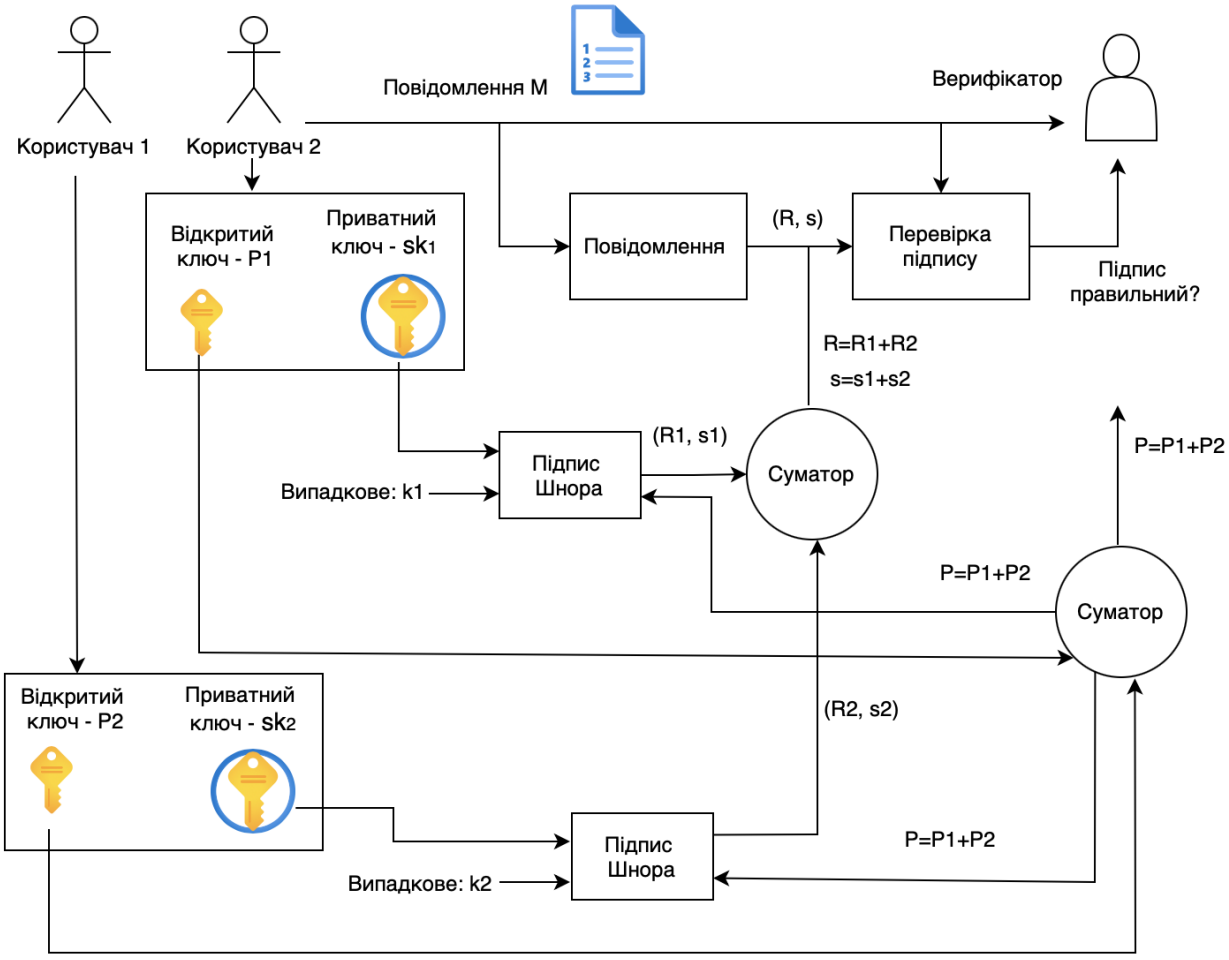


Рисунок 2.2 – Схема мультипідпису з еліптичною кривою

Розглянемо приклад мультипідпису з еліптичною кривою NIST P-256.

Крива NIST P-256, яка має вигляд:

$$y^2 = x^3 - 3x +$$

$$+41058363725152142129326129780047268409114441015993725554835256314039467401291\ (mod\ p)$$

і яка є модулем простого числа (p):

$$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1.$$

Нехай повідомлення:  «Шухман Вадим».

Приватний ключ Боба: (x1) =
7881274680735287887105368914550251638328484278672554986864622650040873737240;

Публічний ключ Боба: (X1) =
(1203091424751182717086592989829291771296520708208684033420333243169238739499 0,213087572128440665750078063607860182770193394569616451372698606608687241767 06);

Приватний ключ Аліси: (x2) =
3337232136802067821775550187757425881148377071751449077296517450853159681944 5;

Публічний ключ Боба: (X2) =
(9095040917463085440650651204457375567846048647230846933248374942886488264400 3,851817837556476000898312040174760724290851328929663721743536091303577288157 28)’

Публічний ключ Віктора: (X) =
(2492606787753014278339412172588025649906459953293359252908448139490570057394 2,1052994280879899533376189583943079556308617866912680368583186442852456004347 6);

Боб: s1 =
8431705947156982203183385108513256252342508173873108861540006842513841895733 8;

Аліса: s2 =
1040787953442540712218146644902562752624828126568013397747902898991896540874 17;

Віктор: s =
7260376560546764449095106862598126425591093917139666804776809926325956100038 6;

Віктор: R =
10828873898067207254417987156203028757284179320639132715503341913764762545709;

sG=(0x88fd38902cd3af6f13a129d878184348bc49eaa55d33712e35000227c0e5d258,
0x682e8b9f86f12a6accf198b5aa8e30ed0ec6be742e7c00198a8dcf634a825853);

$R + H(X \,||\, M)\, X \; =$
(0x88fd38902cd3af6f13a129d878184348bc49eaa55d33712e35000227c0e5d258 ,
0x682e8b9f86f12a6accf198b5aa8e30ed0ec6be742e7c00198a8dcf634a825853)

Так як $s\,G \; = \; R + H(X \,||\, M)\, X$ то перевірка пройшла успішно.

## 2.3 Порогові підписи

Ця схема є специфічним типом мультипідпису. Вона не покладається на те, що користувачі підписуватимуть повідомлення своїми унікальними ключами; натомість для цього потрібен лише один відкритий ключ і один закритий ключ, що призводить до лише одного цифрового підпису. У режимі мультипідпису кінцеве повідомлення містить цифрові підписи всіх підписувачів і потребує індивідуальної перевірки стороною, що перевіряє, але в порогових підписах перевіряльник перевіряє лише один цифровий підпис. Основна ідея схеми полягає в тому, щоб розділити закритий ключ на кілька частин, і кожен підписувач зберігає свою власну частину закритого ключа. Процес підписання вимагає від кожного користувача використання відповідної частини закритого ключа для підпису повідомлення. У даній схемі лише підмножина підписувачів може створити підпис зі своєю часткою, і немає необхідності, щоб усі учасники співпрацювали для створення підпису.

Зв’язок між підписантами регулюється спеціальним протоколом зв’язку. На відміну від мультипідписів, порогові підписи призводять до меншого розміру транзакції та їх швидше перевірити. Пороговий підпис підвищує доступність і стійкість схеми, оскільки частки підпису зберігаються окремими підписувачами (серверами) [25, 26].

Однак невелике обмеження полягає в тому, що для роботи порогових підписів усі підписувачі, залучені до процесу підписання, повинні залишатися онлайн, щоб брати участь в інтерактивному протоколі для генерації підпису, тоді як у мультипідписах підпис можна надавати асинхронно; тобто користувачі можуть надавати підписи, коли вони доступні.

Існують також варіанти, які не є інтерактивними, де кожен користувач обчислює свою частку підпису без будь-якого онлайн-спілкування з іншими користувачами (серверами). З іншого боку, у схемах мультипідпису може бути сценарій, коли користувач може зловмисно приховати свій підпис, що може призвести до відмови в обслуговуванні. За допомогою порогових підписів можливо побудувати необхідний підпис лише з будь-якою авторизованою підмножиною всієї групи, тобто $t + 1 < n$, але принаймні $t + 1$ учасників мають бути онлайн.

Порогові підписи також можна використовувати для надання послуг анонімності в мережі блокчейн (рисунок 2.3).

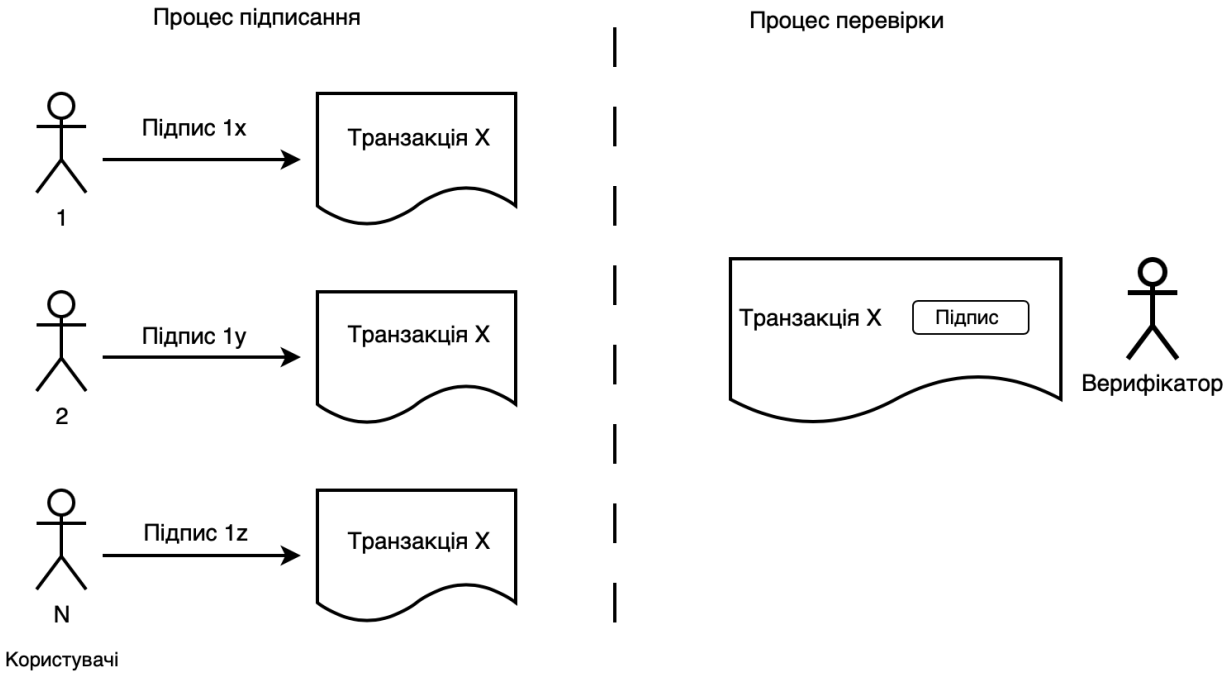


Рисунок 2.3 – Порогова схема підпису

Це відбувається тому, що схеми порогового підпису не розкривають членів порогової групи, які підписалися для створення підпису. Це відрізняється від схем мультипідпису, які розкривають особи всіх підписантів. Крім того, у дозволених мережах порогові підписи можна використовувати в сценаріях, де для узгодження операції потрібен поріг користувачів

На попередній діаграмі ліворуч показано процес підписання, де $m$ різних користувачів, які мають різні частини (частки) цифрового підпису, підписують одну транзакцію. Коли валідатор або верифікатор отримує його, потрібно перевірити лише один підпис.

Алгоритм цифрового підпису на основі кривої Едвардса (EdDSA) використовується для створення цифрового підпису з використанням покращення підпису Шнорра за допомогою скручених кривих Едвардса. Загалом він швидший, ніж багато інших методів цифрового підпису, і надійний з точки зору безпеки [27].

Одним із прикладів EdDSA є Ed25519, який базується на кривій 25519. Він генерує 64-байтове значення підпису (R, s) і має 32-байтові значення для відкритого та закритого ключів. У цьому випадку розділимо закритий ключ на

секретну політику спільного доступу 2 із 3 і розподілимо його між кількома сторонами. Далі кожна зі сторін може створити частину підпису, а потім вони об’єднуються разом для створення загального підпису. Таким чином закритий ключ ніколи не потрібно перебудовувати, щоб створити підпис для об’єкта даних.

За допомогою Ed25519 Боб спочатку генерує 256-бітне випадкове число для свого закритого ключа (priv), а потім створює відкритий ключ за допомогою [28]:

$$pub = H(priv)[:32] \cdot B,$$

де $B$ – базова точка кривої, а $H()$ – хеш-функція (SHA-512).

У цьому випадку ми беремо молодші 32 байти хешу SHA-512. Зазвичай ми просто визначаємо це як координату точки $y$.

Таким чином, відкритий ключ має 32 байти (256 біт). Щоб обчислити підпис, ми генеруємо:

$$r = H_{int}\ (H(priv)[32:] \mid\mid M)(mod\ l),$$

де $H_{int}$ – хеш-функція (SHA-512), перетворена на ціле значення.

У цьому випадку для розрахунку ми беремо старші 32 байти. Значення $l$ є порядком кривої. Тоді $M$ – байтове значення повідомлення. З '||' ми додаємо байти разом. Потім ми перетворюємо це на точку на кривій за допомогою:

$$R = r \cdot B$$

Далі обчислюємо:

$$h = H_{int}(R||pub||M)(mod\ l)$$

і

$$s = H_{int}(priv))[:32]$$

$$S = (r + h \cdot s)(mod\ l)$$

Тоді підпис $(R, S)$. Боб надсилає Алісі $M$, $R$, $S$ і $pub$. Потім Аліса обчислює:

$$k = H(R \,||\, pub \,||\, M) (\text{mod } l),$$

$$P_1 = S \cdot B,$$

$$P_2 = R + k \cdot pub,$$

і якщо $P_1$ дорівнює $P_2$, підпис перевіряється. Це працює тому, що:

$$P_1 = S \cdot B = (r + h \cdot s(modl)) \cdot B = r \cdot B + h \cdot s \cdot B = R + h \cdot pub = P_2.$$

Зауважте, що $l$ є порядком кривої $l$:

$$(l = 2^{252} + 27742317777372353535851937790883648493).$$

Значною перевагою використання Ed25519 над ECDSA є те, що ми можемо об’єднувати підписи та відкриті ключі.

Спочатку кожна сторона створює секрет $(s_i)$ і передає відкритий ключ

$$A_i = s_i \cdot B,$$

і де $B$ є базовою точкою на кривій.

Сторона також генерує випадкове значення $r_i$ та зобов’язується:

$$R_i = r_i \cdot B.$$

Нарешті кожна сторона генерує:

$$S_i = r_i + h \ \cdot s_i (mod\ l),$$

де $q$ - порядок кривої. Потім кожна сторона зобов'язується $(A_i, R_i, S_i)$ і надсилає іншим сторонам та чекає на відповідь про зобов'язання. З усіма отриманими зобов'язаннями ми можемо відновити:

$$R = R_1 + R_2 + \cdots R_2,$$

$$S = S_1 + S_2 + \ldots S_n.$$

Тоді відкритий ключ $(A)$:

$$A = A_1 + A_2 + \ldots A_n,$$

який дорівнює:

$$A = s_1 \cdot B + s_2 \cdot B + \ldots s_n \cdot B.$$

Потім ми просто перевіряємо за допомогою:

$$S \cdot B = R + k \cdot A.$$

Тоді підпис $(R, S)$.

За допомогою розділення секрету Шаміра ми змінюємо так, щоб кожна сторона мала значення секретного ключа $(s_i)$ і помножене на коефіцієнт Лагранжа $l_i$, який відповідає певній частці.

Потім відновлюємо за допомогою:

$$A_i = s_i \cdot l_i \cdot B,$$

$$R_i = r_i \cdot B,$$

$$S_i = r_i + k \cdot l_i \cdot s_i (mod\ p).$$

Повідомлення: Vadym Shukhman

Відкритий ключ:

e0c205f73ed6826a948f97330595c70708e4eecd316e4ea593a0c7cd0b7e99b9

Порогове значення Sig1:

e6d7797b689f38a3a7cc24e9624cd2876caf92fbed5dbc04e785c1cd83daca0233cc8fe1cb960b8cb3ff317a889c6888060e233f29cdff80f13cffb711d02a0b.

Порогове значення Sig2:

e6d7797b689f38a3a7cc24e9624cd2876caf92fbed5dbc04e785c1cd83daca02a1c95e2d24a83abea0b4c9fb1030ad63d21d8f05be493fc0ec5aedf2bbac330d.

Порогове значення Sig3:

e6d7797b689f38a3a7cc24e9624cd2876caf92fbed5dbc04e785c1cd83daca020fc72d797cb969f08d69617d99c3f13e9e2dfbcb52c67effe778db2d66893c0f.

Зміна підпису із спільний доступ 1 і 3:

e6d7797b689f38a3a7cc24e9624cd2876caf92fbed5dbc04e785c1cd83daca02c5cec0957385dc59c64a9af8ff0824ad3afeb6789450c041f61e117d67f32109.

Зміна підпису із спільний доступ 2 і 3:

e6d7797b689f38a3a7cc24e9624cd2876caf92fbed5dbc04e785c1cd83daca02c5cec0957385dc59c64a9af8ff0824ad3afeb6789450c041f61e117d67f32109.

Підпис перевірено.

Довжина підпису становить 64 байти, які складаються з 32 байтів для значення R і 32 байтів для значення s. Довжина відкритого ключа також становить 32 байти, а закритого –32 байти. Загалом Ed25519 створює один із найменших розмірів підпису та має невеликий розмір відкритого та закритого ключів.

## 2.4 Агреговані підписи

Агреговані підписи використовуються для зменшення розміру цифрових підписів. Ця схема особливо корисна в сценаріях, де використовується кілька цифрових підписів. Основна ідея полягає в тому, щоб

об’єднати кілька підписів в один підпис без збільшення розміру підпису одного повідомлення. Це просто тип цифрового підпису, який підтримує агрегацію. Невеликого сукупного підпису достатньо, щоб підтвердити верифікатору, що всі користувачі підписали свої вихідні повідомлення. Агреговані підписи зазвичай використовуються для зменшення розміру повідомлень у мережі та протоколах безпеки. Наприклад, розмір ланцюжків цифрових сертифікатів в інфраструктурі відкритих ключів (PKI) можна значно зменшити, стиснувши всі підписи в ланцюжку в єдиний підпис. Підписи Боне–Лінна–Шахама (BLS) є популярним прикладом агрегованого підпису [29, 30].

Агрегація дозволяє стискати багато підписів від багатьох різних користувачів в один 48-байтовий підпис. Замість того, щоб писати $x$ кількість підписів для $x$ кількості транзакцій, ви можете стиснути та записати лише один підпис у блокчейні. Ця властивість робить підписи BLS дуже корисними для блокчейнів завдяки перевагам економії місця. Крім того, його можна швидко перевірити, що робить його ще більш придатним для блокчейнів. Підписи BLS дозволяють досить легко створювати схем порогових значень типу $m$ із $n$, де для підписання транзакції з $n$ користувачів потрібен поріг із $m$ користувачів.

Підписи BLS використовуються в Ethereum та деяких інших блокчейнах. Підписи BLS мають довжину 48 байт; їх можна агрегувати та дозволяти будувати порогові схеми. Більшість нових блокчейнів використовують підписи BLS через їх менший розмір і функцію агрегації.

## 2.5 Кільцеві підписи

Кільцеві підписи були представлені в 2001 році Роном Ріввестом, Аді Шаміром і Яель Тауман [31].

Схеми кільцевого підпису дозволяють механізм, за допомогою якого будь-який член групи підписувачів може підписати повідомлення від імені всієї групи. Ключова вимога тут полягає в тому, що особистість фактичного

підписанта, який підписав повідомлення, повинна залишатися невідомою (неможливо визначити обчислювальним способом) сторонньому спостерігачеві. Здається однаково ймовірним, що будь-хто з довіреної групи підписантів міг підписати повідомлення, але неможливо з'ясувати, хто насправді підписав повідомлення.

Кожен член кільцевої групи зберігає відкритий і закритий ключ. Кільцеві підписи можна використовувати для надання послуг із збереженням конфіденційності (анонімності). У CryptoNote використовується різновид кільцевих підписів, які називаються відстежуваними кільцевими підписами, щоб можна було запобігти подвійним витратам. Криптовалюта Monero також використовує кільцеві підписи.

Кільцевий підпис – це цифровий підпис, створений членом групи, у кожного з яких є власні ключі. Тоді неможливо визначити особу в групі, яка створила підпис. Спочатку метод був створений Роном Рівестом, Аді Шаміром і Яель Тауман у 2001 році [32, 33].

У кільцевому підписі ми визначаємо групу об'єктів, кожна з яких має пари відкритих/приватних ключів $(P1, S1)$, $(P2, S2)$, …, $(Pn, Sn)$. Якщо нам потрібна сутність і щоб підписати повідомлення (повідомлення), вони використовують свій власний секретний ключ $(s_i)$, але відкриті ключі інших у групі $(m, si, P1 \dots Pn)$. Після цього повинна бути можливість перевірити дійсність групи, знаючи відкритий ключ групи, але неможливо визначити дійсний підпис, якщо немає інформації про приватні ключі всередині групи.

Припустимо, що Джон, Боб, Єва та Аліса знаходяться в групі, і кожен з них має свій відкритий та секретний ключ. Тепер Боб хоче підписати повідомлення від групи. Спочатку він генерує випадкове значення v, а потім генерує випадкові значення $(s_i)$ для кожного з інших учасників, але бере свій власний секретний ключ $(s_i)$ і використовує його для визначення іншого секретного ключа, а також зворотного до функції шифрування. Далі він приймає повідомлення та бере його хеш, і таким чином створюється ключ $(k)$. Ключ буде використовуватися з симетричним шифруванням для шифрування

кожного з елементів кільця ($E_k$), і кожен елемент кільця використовує функцію EX-OR з попереднього елемента (рисунок 2.4). Потім кожне з випадкових значень для інших учасників шифрується відкритим ключем даного учасника, а потім обчислюється значення ys щоб створити кільце (результат кільця має дорівнювати v). Потім він інвертує це значення, щоб отримати еквівалентний закритий ключ (xs). Тепер Боб відкриває загальний підпис і випадкові значення x разом із обчисленим секретним ключем. Щоб перевірити підпис, отримувач просто обчислює кільце та перевіряє, чи результат відповідає надісланому підпису.

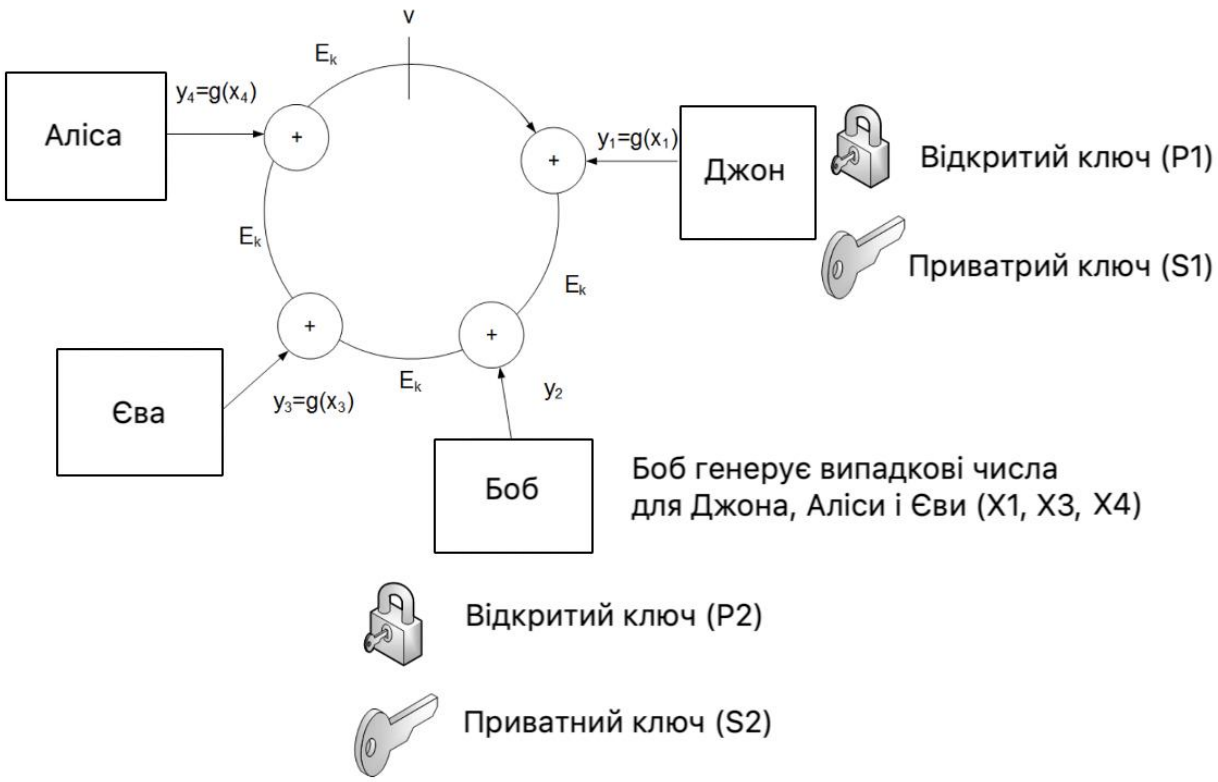


Рисунок 2.4 – Кільцевий підпис

Основні кроки кільцевого підпису:

1) шифрування за допомогою $k = Hash(message)$;

2) генерування випадкового значення $(u)$;

3) шифрування $u$, щоб отримати $v = E_k(u)$;

4) для кожної особи (крім відправника):

4.1) обчислення $e = {S_i}^{P_i}(mod\ N_i)$, де $s_i$ – випадкове число, згенероване для секретного ключа і-ї сторони, а $P_i$ – відкритий ключ сторони;

4.2) обчислення $v = v \oplus e$;

5) для підписаної сторони $(z)$ обчислення $s_z = (v \oplus u)^d (mod N_z)$ і де d є секретним ключем підписувача.

В результаті отримаємо підпис $(v = E_k(u))$, який завершує кільце.

Розглянемо приклад. У цьому прикладі ми маємо чотирьох підписувачів, тому підпис містить чотири секретні ключі (s0, s1, s2 і s3) і значення підпису.

Повідомлення: «Digital signature algorithms for blockchain technologies!»

Підпис – 25345300614735639055143943485649996302636183073 9

s0 дорівнює:

5636348685310013448396737364106768148090289566898186315457303747156532755797385294332493024569107116734718500697339878340711882086180146935341795339805696217237899847108433296304978661985725744129946187957930629571730818033164855829194801616788024119565520388973374551065619066454769951655964023395059447396

s1 дорівнює:

8885131513282245459920012852371698246406884435856612278601965699243286308368155485897572372569886824261744874778223371060760109214390923899581459938669920162154927980421215256386713621680623155834590170713228146993648298129243773223035213448334131223636377146286877680240991738603770257762897044818721595826 06

s2 дорівнює:

2187126811269718298335472546218475588526832549790182092344853860906330219924660943914365831770002631534396668449415900882445100027402028332563476227869625578854931298258913803298151267583622405065344260945533949009711055474181753321197532473127081701110360729636234375420993321714885740641116327148144065572 7

s3 дорівнює:

863151156562221730240147835325953504756975157247486695174017530403704700180223562903299063176405406664186652175713678040067840555944632975549577407917672391112524769690528307825568997532667204015293336969607936882663891292809939354629867900494364460810916576895244146483598716494683799310980569531711956309 76

Підпис перевірено: так.

# 3 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ АЛГОРИТМІВ ЦИФРОВИХ ПІДПИСІВ В БЛОКЧЕЙН

## 3.1 Порівняння схем цифрових підписів в технології блокчейн

У технології блокчейн цифрові підписи відіграють ключову роль у забезпеченні безпеки, автентифікації та цілісності даних. Серед популярних схем виділяють мультипідпис, кільцевий підпис та агрегований підпис, кожен із яких має унікальні особливості та сфери застосування. Мультипідпис забезпечує колективну відповідальність і використовується у системах з багатьма підписантами, тоді як кільцевий підпис підходить для конфіденційних транзакцій завдяки своїй анонімності. Агрегований підпис дозволяє значно зменшити розмір підписів, що є особливо важливим для блокчейн-систем із високою пропускною здатністю. У таблиці 3.1 наведено порівняння основних параметрів та характеристик схем цифрових підписів.

Таблиця 3.1 – Характеристик схем цифрових підписів

| Характеристика | Мультипідпис | Кільцевий підпис | Агрегований підпис |
|---|---|---|---|
| Призначення | Підпис кількома сторонами для спільного узгодження | Захист анонімності підписанта у групі | Об'єднання підписів декількох користувачів в один |
| Структура | Один підпис формується за участю кількох ключів | Один підпис формується з набору публічних ключів | Об'єднання кількох окремих підписів |

Продовження таблиці 3.1

| Характеристика | Мультипідпис | Кільцевий підпис | Агрегований підпис |
|---|---|---|---|
| Анонімність | Ні (ключі ідентифіковані) | Так (неможливо визначити підписанта) | Ні (ключі відомі, але зменшено розмір даних) |
| Розмір підпису | Менший за суму окремих підписів | Більший через включення інформації про групу | Один підпис для всіх транзакцій |
| Ефективність перевірки | Вимагає перевірки одного складного підпису | Простіший, оскільки перевіряється один підпис | Ефективний, оскільки перевіряється лише один підпис |
| Захист від підробки | Високий, базується на складності обчислення | Високий, але вразливий до витоку одного з ключів | Високий, залежить від стійкості окремих підписів |
| Використання | Смарт-контракти, багатокористувацькі транзакції | Системи голосування, конфіденційні платежі | Масштабовані блокчейни, групові транзакції |
| Складність реалізації | Середня (потрібно узгодження між сторонами) | Низька (потрібен лише набір ключів для | Середня (потрібна синхронізація підписів) |

| | | | |
|---|---|---|---|
| | | формування групи) | |

Продовження таблиці 3.1

| Характеристика | Мультипідпис | Кільцевий підпис | Агрегований підпис |
|---|---|---|---|
| Стійкість до атак | Стійкий до атак на кожного окремого підписанта | Може бути вразливий до витоку одного секретного ключа | Висока, але залежить від стійкості агрегованого підпису |
| Переваги | Спільний контроль, підвищення довіри | Анонімність і конфіденційність | Зменшення розміру блокчейну, швидкість перевірки |
| Недоліки | Складність узгодження | Розмір підпису збільшується зі зростанням групи | Вразливість до компрометації хоча б одного підпису |

Аналіз таблиці 3.1 показав, що мультипідпис підходить для сценаріїв, де потрібен спільний контроль, наприклад, управління криптогаманцями.

Кільцевий підпис забезпечує анонімність і конфіденційність та більш ефективний у системах голосуваннях чи конфіденційних транзакціях.

Агрегований підпис оптимізує розмір даних і швидкість, що робить його ефективним для масштабованих блокчейн-рішень.

Порівняння розмірів відкритого ключа, закритого ключа та цифрового підпису для схем мультипідпису, кільцевого та агрегованого підпису приведені в таблиці 3.2.

Аналізуючи таблицю 3.2, бачимо, що в мультипідписі відкритий ключ містить комбінацію ключів усіх учасників, тому його розмір пропорційний

кількості учасників. При цьому, закритий ключ не агрегується, його розмір залишається сталим. Підпис менший за суму окремих підписів, оскільки алгоритм узагальнює інформацію.

Таблиця 3.2 – Характеристик схем цифрових підписів

| Характеристика | Мультипідпис | Кільцевий підпис | Агрегований підпис |
|---|---|---|---|
| Розмір відкритого ключа | Залежить від кількості учасників $(n \times k)$, де k – розмір ключа одного учасника | Залежить від кількості учасників $(n \times k)$ | Залежить від кількості учасників $(n \times k)$ |
| Розмір закритого ключа | Розмір закритого ключа одного учасника $(k)$ | Розмір закритого ключа одного учасника $(k)$ | Розмір закритого ключа одного учасника $(k)$ |
| Розмір підпису | Менший за суму окремих підписів $(1 \times k)$ | Збільшується пропорційно кількості учасників $(n \times k)$ | Один агрегований підпис $(1 \times k)$ |

В схемі кільцевого підпису відкритий ключ містить інформацію про всіх учасників кільця, тому його розмір збільшується зі зростанням кількості учасників. Закритий ключ залишається сталим, оскільки для підписання потрібен лише ключ підписанта. Розмір підпису прямо пропорційний кількості учасників.

В схемі агрегованого підпису відкритий ключ також агрегується, але його розмір залежить від кількості учасників. Закритий ключ зберігає початковий розмір. Розмір підпису фіксований незалежно від кількості учасників, оскільки всі підписи зливаються в один (таблиця 3.3).

Як видно з таблиці 3.3 агрегований підпис є найбільш ефективним за розміром підпису. Мультипідпис оптимізує розмір підпису, але потребує більших відкритих ключів. Кільцевий підпис забезпечує анонімність, але має найбільший розмір підпису через включення ключів усіх учасників [35].

Таблиця 3.3– Кількісні характеристики цифрових підписів

| Метрика | Оцінка розміру в байтах (приклад: k = 32) | | |
|---|---|---|---|
| | Мультипідпис | Кільцевий | Агрегований |
| Відкритий ключ (n = 5) | 160 | 160 | 160 |
| Закритий ключ | 32 | 32 | 32 |
| Підпис | 32 | 160 | 32 |

Схеми мультипідпису, кільцевого та агрегованого підпису можуть бути реалізовані за допомогою різних криптографічних алгоритмів (таблиця 3.4).

Таблиця 3.4 – Алгоритми реалізації схем цифрових підписів

| Схема цифрового підпису | Можливі алгоритми реалізації |
|---|---|
| Мультипідпис | 1. ECDSA (Elliptic Curve Digital Signature Algorithm): використовується у блокчейнах, таких як Bitcoin. |
| | 2. Schnorr: забезпечує просту реалізацію мультипідпису з меншою складністю, ніж ECDSA. |
| | 3. EdDSA (Edwards-Curve Digital Signature Algorithm): зручний для мультипідпису завдяки ефективності та підтримці агрегованих операцій. |

Продовження таблиці 3.4

| Схема цифрового підпису | Можливі алгоритми реалізації |
|---|---|
| Кільцевий підпис | 1. RSA (Rivest-Shamir-Adleman): базовий підхід для реалізації кільцевих підписів. |
| | 2. ECC (Elliptic Curve Cryptography): дає менший розмір ключів і підписів. |
| | 3. One-Time Ring Signature (OTRS): специфічний підхід для конфіденційних транзакцій (використовується в Monero). |
| Агрегований підпис | 1. BLS (Boneh-Lynn-Shacham): дозволяє об'єднувати кілька підписів у один, забезпечуючи компактність та ефективність. |
| | 2. Schnorr: підтримує агреговані підписи завдяки математичній структурі, яка дозволяє додавати підписи. |

Мультипідпис найчастіше реалізується на основі алгоритмів ECDSA, Schnorr, або EdDSA. Застосовується для сценаріїв, де кілька учасників мають погодити транзакцію чи підписати документ.

Кільцевий підпис може використовувати алгоритм RSA або ECC, але популярним вибором є One-Time Ring Signature (OTRS), який використовується в анонімних криптовалютах, таких як Monero. Підписувач створює підпис, який показує приналежність до групи, але не розкриває особистість.

Агрегований підпис найчастіше реалізується за допомогою BLS або Schnorr. Ця схема особливо ефективна для зменшення розміру підписів у блокчейнах з великою кількістю транзакцій, таких як Ethereum 2.0.

Отже, алгоритм BLS є основним вибором для агрегованих підписів завдяки компактності.

Алгоритм Schnorr універсальний і застосовується для мультипідпису та агрегованих підписів.

Для кільцевих підписів найбільш ефективними є алгоритми, які забезпечують анонімність, наприклад, OTRS.

## 3.2 Алгоритм та програмна реалізація агрегованого цифрового підпису

Схема агрегованого цифрового підпису дозволяє об’єднувати кілька цифрових підписів від різних користувачів, підписаних на різних повідомленнях, в один компактний підпис. Цей агрегований підпис підтверджує, що всі учасники дійсно підписали свої повідомлення (рисунок 3.1).

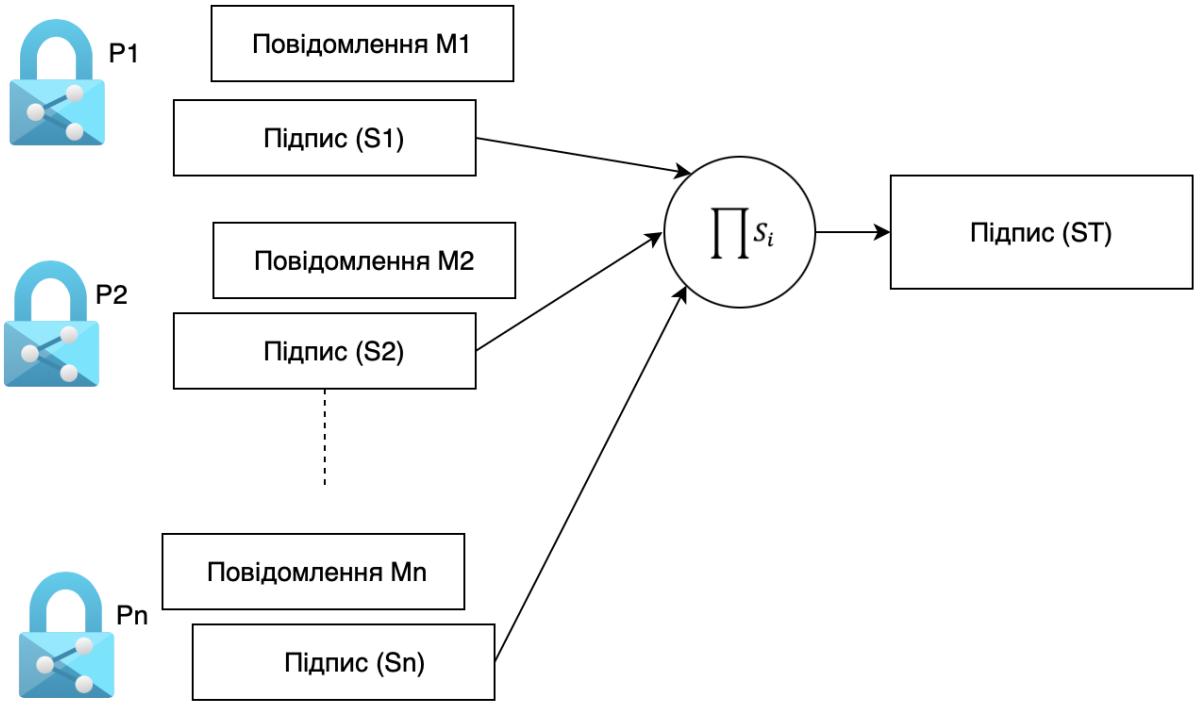


Рисунок 3.1 – Схема агрегації цифрового підпису

Якщо у нас є одне повідомлення, наприклад транзакція криптовалюти, ми можемо об’єднати відкриті ключі разом (агрегація ключів) і перевірити підпис (рисунок 3.2).

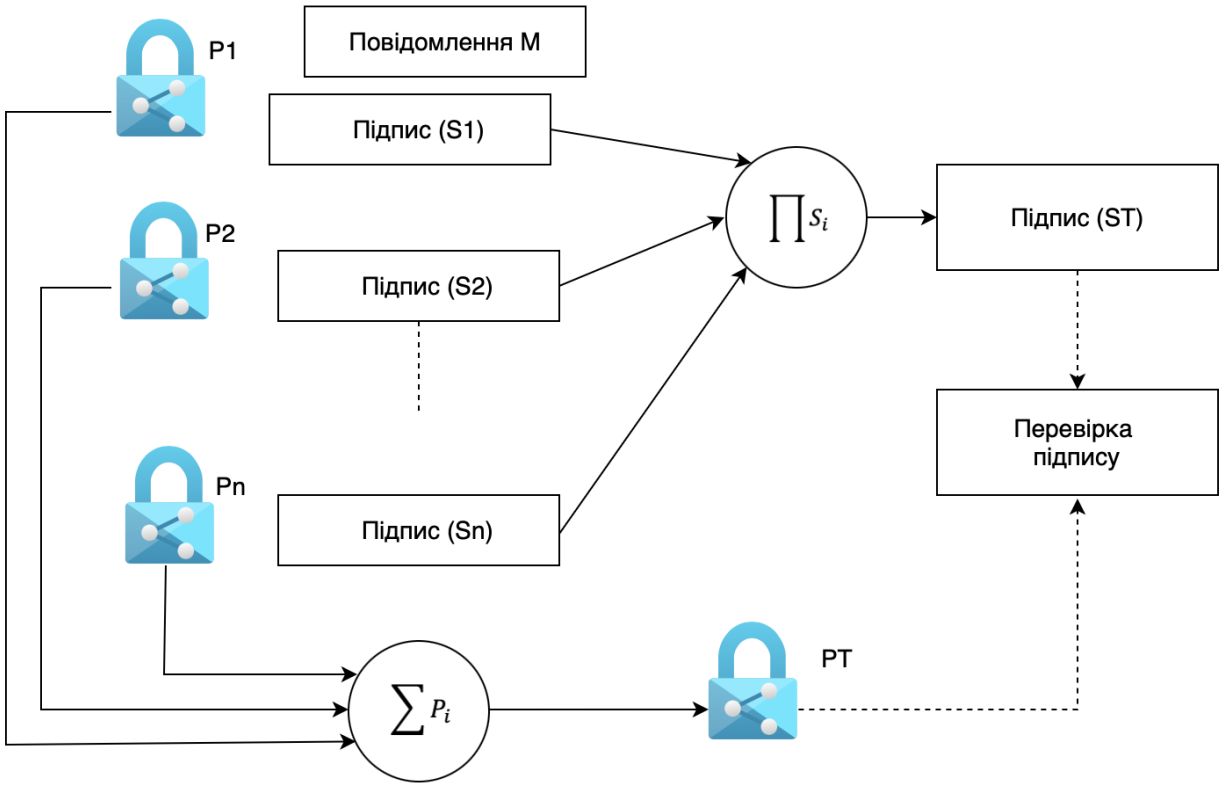


Рисунок 3.2 – Схема агрегації ключів

Схема агрегованого цифрового підпису складається з наступних елементів: створення ключів, формування вхідних даних, обчислення результату.

1. Створення ключів. Кожен підписант має приватний ключ для підпису повідомлень і відповідний публічний ключ для перевірки підпису.

2. Формування вхідних даних. Кожне повідомлення $m_i$, підписане приватним ключем відповідного учасника $PK_i$:

$$(m_1, PK_1), (m_2, PK_2), \dots, (m_n, PK_n).$$

3. Обчислення результату. Один компактний підпис $\sigma$, який можна перевірити за сукупністю всіх публічних ключів $\{PK_1, PK_2, \cdots PK_n\}$ і відповідних повідомлень.

Розглянемо кожний з етапів детальніше.

1. Генерація ключів. Кожен учасник $i$ генерує пару ключів $(SK_i, PK_i)$, де $SK_i$ – приватний ключ, $PK_i -$ публічний ключ.

2. Підписання. Кожен учасник підписує своє повідомлення $m_i$, отримуючи підпис $\sigma_i$, використовуючи свій приватний ключ $SK_i$.

3. Агрегування підписів.

Всі окремі підписи $\sigma_1$, $\sigma_2$, … $\sigma_n$, об'єднуються в один агрегований підпис $\sigma$, за допомогою математичної операції додавання або множення в еліптичних групах.

4. Перевірка. Сторона, яка перевіряє агрегований підпис має список усіх $(m_i, PK_i)$.

Використовує агрегований підпис $\sigma$ і список $\{PK_i\}$, щоб підтвердити, що всі повідомлення $m_i$, були підписані їх відповідними підписантами.

Багато схем агрегованих підписів базуються на еліптичних кривих та парах Біллінеарних карт (Bilinear Pairings), таких як схема BLS (Boneh–Lynn–Shacham).

Розглянемо приклад на основі BLS.

Створення ключів. Генерується група $G$ порядку $q$ з білінійним відображенням $e$: $G \times G \to G_T$.

Публічний ключ:

$$PK = g^{SK},$$

де $SK$ – приватний ключ, $g \in G$.

Підписання.

Підпис обчислюється за формулою:

$$\sigma_i = H(m_i)^{SK_i},$$

де $H(m_i)$ – хеш-функція, яка відображає $m_i$ в $G$.

Агрегування підписів. Агрегований підпис обчислюємо за формулою:

$$\sigma = \prod_{i=1}^{n} \sigma_i,$$

Перевірка підпису. Перевіряється умова:

$$e(\sigma, g) = \prod_{i=1}^{n} e(H(m_i), PK_i).$$

Алгоритм створення агрегованого цифрового підпису складається з наступних кроків:

1. Генерування ключів. Для кожного учасника генеруються пара ключів: приватний (SK) і публічний (PK).

Використовується модуль $p$, який є великим простим числом.

2. Підписування. Кожен учасник підписує своє повідомлення за допомогою хеш-функції SHA-256.

3. Агрегування. Підписи перемножуються за модулем $p$, утворюючи один агрегований підпис.

4. Перевірка. Перевіряється, чи агрегований підпис відповідає всім публічним ключам та повідомленням.

Код програми, яка реалізує концепцію агрегованого підпису приведений в додатку А. Вона дозволяє чотирьом учасникам підписати свої повідомлення, після чого всі підписи об'єднуються в один компактний агрегований підпис.

Програма містить такі основні блоки: генерація ключів, підпис повідомлення, агрегування підписів та перевірка агрегованого підпису.

Розглянемо роботу основних блоків детальніше.

В основній частині програми:

1) генеруються ключі для 4 учасників;

2) кожен учасник підписує своє повідомлення;

3) усі підписи агрегуються в один підпис;

4) перевіряється, чи агрегований підпис підтверджує всі повідомлення.

1. Генерація ключів.

Функція generate_keypair() генерує пару ключів для кожного учасника.

Приватний ключ (sk) – випадкове число в межах від 1 до $p - 1$, де $p$ – велике просте число.

Публічний ключ (pk) обчислюється як $sk\ (mod\ p)$.

Код функції:

```
sk = randint(1, p - 1)  # Приватний ключ
pk = sk % p  # Публічний ключ
```

Ці ключі використовуються для підписання та перевірки повідомлень.

2. Підпис повідомлення.

Опис функції sign_message().

1. Функція обчислює хеш суму повідомлення $m$ за допомогою SHA-256, щоб отримати його числове представлення.

2. Генерує підпис $\sigma$ для повідомлення $m$, використовуючи приватний ключ $sk$:

$$\sigma = H(m)^{SK}(mod\ p).$$

Код функції:

```
hashed_message = int.from_bytes(sha256(message.encode()).digest(),
byteorder='big') % p
signature = pow(hashed_message, private_key, p)
```

3. Агрегування підписів.

Функція aggregate_signatures() приймає список підписів і об'єднує їх в один агрегований підпис шляхом перемноження всіх підписів за модулем $p$:

$$\sigma_{agr} = \prod_{i=1}^{n} \sigma_i \ (mod\ p).$$

Код функції:

```
aggregate_signature = 1
for sig in signatures:
    aggregate_signature = (aggregate_signature * sig) % p
```

Даний алгоритм дозволяє створити компактний підпис, який представляє всі окремі підписи.

4. Перевірка агрегованого підпису.

Функція verify_aggregate_signature() перевіряє, чи дійсно агрегований підпис підтверджує всі повідомлення.

Перевірка здійснюється наступним чином.

1. Обчислюється ліва частина перевірки:

$$left_{side} = \sigma_{agr} (mod\ p).$$

Для кожного повідомлення та відповідного публічного ключа:

– обчислюється хеш повідомлення;

– обчислюється перевірочне значення:

$$H(m_i)^{PK_i} (mod\ p).$$

Перевіряється рівність:

$$\sigma_{agr} = \prod_{i=1}^{n} H(m_i)^{PK_i} \ (mod\ p).$$

Код функції:

```
for message, public_key in zip(messages, public_keys):
    hashed_message = int.from_bytes(sha256(message.encode()).digest(),
byteorder='big') % p
    right_side = (right_side * pow(hashed_message, public_key, p)) % p
return left_side == right_side
```

Результати виконання програми мають вигляд:

Публічні ключі:

[914589619402666818892962975609903638116461748207385494648532401263679783675 10,
419587111243859556214311080414585188870327856266266979074120098105878146371 09,
115296701853598706605140552127622907222335911014077606343543184930907854571 693,
682130218215367855320201292787342891082037376270008365572493924172960918160 93].

Підписи:

[135594046239952667259424745455031027571038290791550475634642689947613571599 42,
103557090806407259202227321827943072406742950146746904511581844362338161255 900,
140897962774170296048612837033661079626443287354330657229887770613321751825 93,
444447982735241858077462007085827481848645442679687096174863637391955874993 90].

Агрегований підпис:

373920170440289176146928466596520791469128059758020456113365555909569772529 32

Перевірка: успішна.

Агреговані підписи можна ефективно використовувати для підписання транзакцій в блокчейні. Деякі транзакції мають підписувати кілька користувачів, наприклад, коли витрачаються гроші з кількох адрес у криптовалюті. Розмір даних транзакції можна зменшити за допомогою агрегованих підписів: агрегація підписів може бути виконана майнерами перед включенням підпису в транзакцію. У цьому випадку навіть мультипідписів буде достатньо. Крім того, загальні агреговані підписи можна використовувати для агрегування підписів у різних транзакціях перед включенням у блок. У той же час потрібна додаткова обережність, оскільки деякі загальні схеми агрегованого підпису, наприклад, BLS, дозволяють зловмиснику створити відкритий ключ таким чином, щоб його підпис довільних даних був дійсним сукупним підписом для будь-яких відкритих ключів вибір супротивника та створений ним відкритий ключ. Одним із рішень цього є вимагати від користувача довести, що він правильно обчислив відкритий ключ, підтвердивши знання відповідного закритого ключа.

## 3.3 Дослідження постквантових алгоритмів цифрового підпису

На даний момент CRYSTALS підтримує два надійні механізми: Kyber для механізму інкапсуляції ключів (KEM) і обміну ключами; і Dilithium для алгоритму цифрового підпису.

CRYSTALS Dilithium використовує схеми Фіата-Шаміра на основі решітки та створює одну з найменших сигнатур серед усіх постквантових методів із відносно малими розмірами відкритого та закритого ключів. Три основні реалізації параметрів, які використовуються: Dilithium 2, Dilithium 3 і Dilithium 5. Загалом, Dilithium 3 еквівалентний 128-бітному підпису та, можливо, є відправною точкою для реалізації [36].

Проведемо порівняння основних параметрів Dilithium 2, Dilithium 3 і Dilithium 5 з використанням бібліотеки Cloudflare CIRCL. NIST

стандартизував Dilithium за допомогою FIPS 204 (стандарт цифрового підпису на основі модульної решітки - ML-DSA). Методи ML-DSA: ML-DSA-44 (рівень безпеки 1), ML-DSA-65 (рівень безпеки 3) і ML-DSA-87 (рівень безпеки 5).

1. Метод підпису: Ed25519. Повідомлення: Cyber security.

Результати обчислення.

Приватний ключ:

103bea9db21c00187f9d2786cb3bad0e350dbb6ff56facabe51646e9dc3e9921

(показано перші 32 байти).

Довжина приватного ключа: 64.

Відкритий ключ:

c25aec3fc77cc375c0436e2ec85c81453ecd558c6e22b4fc8a0a1d56e4565726

(показано перші 32 байти).

Довжина відкритого ключа: 32.

Підпис:

365b0073738fc8007ec412058a7af284bba5b8e2c5dacffe98e57c90cc0eb67d
4fd9294d1e46da61db6c884b41a95056ce63f2722f149bcfbd72b3fde9f2ab04

Довжина підпису: 64

Підпис перевірено.

2. Метод підпису: Ed448

Приватний ключ:

bcc12b61ad5315dfcff2a296332cee7f8e927be78795542b8bde84527d9c3af5

(показано перші 32 байти).

Довжина приватного ключа: 114

Відкритий ключ:

6133d61add9869c947dcc4d5e19a2ccca3d652ba43fc89a335fe6c72fc31b430

(показано перші 32 байти).

Довжина відкритого ключа: 57

Підпис:

ff47f47a914b586ee9483c8231b6da08b0c214ad79fca9ac5d2b6ef5d102b188
230e5da65f6af26559a51c72e446cc1583c74d9908e8b705807cb74368706945

Довжина підпису: 114

Підпис перевірено.

3. Метод підпису: Ed25519-Dilithium2

Приватний ключ:

9a03217d7da5ed76f021084a8cd6646c6093d5ce0b68b0deb4baf602dfae1ffa

(показано перші 32 байти).

Довжина приватного ключа: 2560

Відкритий ключ:

9a03217d7da5ed76f021084a8cd6646c6093d5ce0b68b0deb4baf602dfae1ffa

(показано перші 32 байти).

Довжина відкритого ключа: 1344

Підпис:

3c50f99e41d974e04c6212c67fa17d8bc956529c81deeb37491a890ea14723e
56aa3fd40d9be4ab23f9094ce521efab4b1cb4a83d49e5aedf14bbb40771a7921

Довжина підпису: 2484

Підпис перевірено.

4. Метод підпису: Ed448-Dilithium3

Приватний ключ::

2cdf0c0fcd863d3d1aa57119a37811a8699cbc50053b404a88d3df21447f2abc
(показано перші 32 байти).

Довжина приватного ключа: 4057

Відкритий ключ:

2cdf0c0fcd863d3d1aa57119a37811a8699cbc50053b404a88d3df21447f2abc

(показано перші 32 байти).

Довжина відкритого ключа: 2009

Підпис:

c7b60ba9b514bd3300777b8f5b58e6767d67219bfa770fc40e334ccaa6863f36
380c87a689a4cb934c4ce974efb5ea7f0dd841bb3cb9e86da4bb50c2f3c3b953

Довжина підпису: 3407

Підпис перевірено.

5. Метод підпису: Dilithium2

Приватний ключ:

9d793d13a104385a40be94891575b6405c114fe9139fe053f1f3eb7ad72d99a6

(показано перші 32 байти).

Довжина приватного ключа: 2528

Відкритий ключ:

9d793d13a104385a40be94891575b6405c114fe9139fe053f1f3eb7ad72d99a6

(показано перші 32 байти).

Довжина відкритого ключа: 1312

Підпис:

ca28dcd43f7fa25ced704ee2701107155e9e397f2656d0fcfdfedd6265a735498
bc8f1ac2f5e82310d17657835b8bbebf8b3648ea5e21324092043209bf7ecdc

Довжина підпису: 2420

Підпис перевірено.

6. Метод підпису: Dilithium3

Приватний ключ:

602e4bfbc4b7415be2722bed27ff0225238a59fe94168286b21b262734a6f829

(показано перші 32 байти).

Довжина приватного ключа: 4000

Відкритий ключ:

602e4bfbc4b7415be2722bed27ff0225238a59fe94168286b21b262734a6f829

(показано перші 32 байти).

Довжина відкритого ключа: 1952

Підпис:

9f189a6e13502ff57883764b37753b3b88b3a074fb45a005740bd04cfed73a82
72f331bd83a66270cb202c4507317bfeb7e8aea00dd0111800c15384053764be

Довжина підпису: 3293

Підпис перевірено.

7. Метод підпису: Dilithium5

Приватний ключ:
874dbb8456e099c8846951356fe949f94e30223498c0108d5dec6db0ba74be80
(показано перші 32 байти).

Довжина приватного ключа: 4864

Відкритий ключ:
874dbb8456e099c8846951356fe949f94e30223498c0108d5dec6db0ba74be80
(показано перші 32 байти).

Довжина відкритого ключа: 2592

Підпис:
af5fb61542f75d392586b3a7591617b243b1549d303a7ba04b9b38c314bdeac
5ac85423bf67ff9ed38c544d008d8f659a1632e7daa175f3d8a10c903245f3454

Довжина підпису: 4595

Підпис перевірено.

8. Метод підпису: ML-DSA-44

Приватний ключ::
4afc016eabb7ca665b1a61d4b1f663aa144fb47243ed521500097f5c3faccddd
(показано перші 32 байти).

Довжина приватного ключа: 2560

Відкритий ключ:
4afc016eabb7ca665b1a61d4b1f663aa144fb47243ed521500097f5c3faccddd
(показано перші 32 байти).

Довжина відкритого ключа: 1312

Підпис:

5a1a7432a64716c11b8455caa571e300f513c16d1ea10e71f4580e6543a22ff5
16931bd66ff5b3490fc8954b3b2302bfff9383b77681a2469b83b8e9f1c5d7de

Довжина підпису: 2420

Підпис перевірено.

9. Метод підпису: ML-DSA-65

Приватний ключ:
ce5298a2ed2e60988f920da591543d037012c54ba0091cc95821c4aae16d2b0f
(показано перші 32 байти).

Довжина приватного ключа: 4032

Відкритий ключ:
ce5298a2ed2e60988f920da591543d037012c54ba0091cc95821c4aae16d2b0f
(показано перші 32 байти).

Довжина відкритого ключа: 1952

Підпис:
5bfbd1e24b0e3cd38baf2c764bee831ddee0237686568dde2e88e3562897fc4b
2403473bbcb2021b27a6d621c18b21f04241be7451b23927751f85ba75d1dcc4

Довжина підпису: 3309

Підпис перевірено.

10. Метод підпису: ML-DSA-87

Приватний ключ:
165c3922e9455c57be8fde477dc2865bf593302f46e6afc77797aaf84a5e5bba
(показано перші 32 байти).

Довжина приватного ключа: 4896

Відкритий ключ:
165c3922e9455c57be8fde477dc2865bf593302f46e6afc77797aaf84a5e5bba
(показано перші 32 байти).

Довжина відкритого ключа: 2592

Підпис:

2d3cd87bc222b493b631acb2193302c91bb92c4df83d6de547dda646aec232f20022d06cea074031c37ad76b32479d30c1f62f26fb0eaea850315b0e43013860

Довжина підпису: 4627

Підпис перевірено.

Результати порівняння цифрових підписів приведені в таблиці 3.5.

Порівняння постквантових алгоритмів цифрового підпису дозволило оцінити їх ефективність, безпеку та придатність для різних сфер застосування. Основними критеріями аналізу були стійкість до квантових атак, продуктивність алгоритмів, розмір ключів і підписів, а також практичні аспекти їхньої реалізації.

Таблиця 3.5 – Порівняння цифрових підписів

| Метод підпису | Довжина приватного ключа | Довжина відкритого ключа | Довжина цифрового підпису |
|---|---|---|---|
| Ed25519 | 64 | 32 | 64 |
| Ed448 | 114 | 57 | 114 |
| Ed25519-Dilithium2 | 2560 | 1344 | 4595 |
| Ed448-Dilithium3 | 4057 | 2009 | 3407 |
| Dilithium2 | 2528 | 1312 | 2420 |
| Dilithium3 | 4000 | 1952 | 3293 |
| Dilithium5 | 4864 | 2592 | 4595 |
| ML-DSA-44 | 2560 | 1312 | 2420 |
| ML-DSA-65 | 4032 | 1952 | 3309 |
| ML-DSA-87 | 4896 | 2592 | 4627 |

Результати показали, що різні підходи мають свої переваги і недоліки. Наприклад, алгоритми на основі теорії решіток, такі як Crystals-Dilithium, забезпечують високу стійкість і баланс між швидкістю операцій та розміром

даних, тоді як коди Хеммінга та інші кодові криптосистеми мають значно більші розміри ключів, але пропонують добре вивчену безпеку. Алгоритми на основі хеш-функцій, такі як SPHINCS+, гарантують стійкість, однак можуть вимагати значних обчислювальних ресурсів, особливо для перевірки підписів.

Таким чином, вибір конкретного алгоритму залежить від конкретних вимог до системи: апаратних обмежень, рівня безпеки, швидкості виконання та стійкості до квантових атак. Загалом, розвиток постквантових алгоритмів цифрового підпису є важливим етапом у підготовці до ери квантових обчислень, що забезпечує захист інформації в умовах нових викликів кібербезпеки.

# ВИСНОВКИ

В кваліфікаційній роботі розв’язано актуальну задачу підвищення ефективності цифрових підписів в технології блокчейн. При цьому отримано наступні результати.

1. Проведено аналіз та порівняння алгоритмів цифрового підпису ECDSA та EdDSA. Аналіз показав, що EdDSA перевершує ECDSA у швидкості виконання операцій і стійкості до певних атак завдяки використанню криптографічної кривої Curve25519. Це робить EdDSA більш перспективним для використання в блокчейні, зокрема в умовах високого навантаження.

2. Досліджено алгоритми цифрового підпису в технології блокчейн. Блокчейн підтримує розвиток складніших криптографічних схем, таких як мультипідписи, порогові підписи, кільцеві підписи та агреговані підписи. Ці схеми дозволяють підвищити функціональність і масштабованість блокчейну, а також покращити конфіденційність і зменшити обсяг збережуваних даних.

3. Досліджено постквантові алгоритми, такі як Нідеррайтера, CRYSTALS-Dilithium, FALCO та SPHINCS+. Результати порівняння вказують на їх важливість у контексті майбутніх викликів квантових обчислень. Хоча вони мають певні недоліки, наприклад великий розмір підпису, їх криптографічна стійкість забезпечує безпеку блокчейну в довгостроковій перспективі.

4. Розроблено алгоритмічну та програмну реалізацію агрегованих підписів, які є ефективним рішенням для зменшення розміру блоків у блокчейні. Це сприяє оптимізації використання ресурсів і підвищенню масштабованості мережі. Для сучасних блокчейн-систем рекомендовано впроваджувати EdDSA у поєднанні з розширеними схемами підписів, такими як мультипідписи та порогові підписи, для досягнення балансу між безпекою та продуктивністю.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Bhutta, M. N. M., Khwaja, A. A., Nadeem, A., Ahmad, H. F., Khan, M. K., Hanif, M. A., Cao, Y. A survey on blockchain technology: Evolution, architecture and security. *Ieee Access*, 2021, *9*, 61048-61073.

2. Guo, H., & Yu, X. A survey on blockchain technology and its security. *Blockchain: research and applications*, 2022, *3*(2), 100067.

3. Sun, X., Yu, F. R., Zhang, P., Sun, Z., Xie, W., & Peng, X. A survey on zero-knowledge proof in blockchain. *IEEE network*, 2021, *35*(4), 198-205.

4. Блокчейн і децентралізовані системи: навч. посібник для студ. закладів вищ. освіти : в 3 частинах. Ч. 1 / П. Кравченко, Б. Скрябін, О. Дубініна. – Харків : ПРОМАРТ, 2019. – 452 с.

5. Блокчейн і децентралізовані системи: навч. посібник для студ. закладів вищ. освіти: в 3 частинах. Ч. 2 / П. Кравченко, Б. Скрябін, О. Курбатов, О. Дубініна. - Харків, 2019. – 412 с.

6. Digital Signature. [Електронний ресурс]. – Режим доступу: https://academy.binance.com/en/glossary/digital-signature.

7. Roy, A., & Karforma, S. A survey on digital signatures and its applications. *Journal of Computer and Information Technology*, 2012, *3*(1), 45-69.

8. Singh, S., Iqbal, M. S., & Jaiswal, A. Survey on techniques developed using digital signature: public key cryptography. *International Journal of Computer Applications*, 2015, *117* (16).

9. Gamage, H. T. M., Weerasinghe, H. D., & Dias, N. G. J. A survey on blockchain technology concepts, applications, and issues. *SN Computer Science*, 2020, *1*, 1-15.

10. Aggarwal, S., & Kumar, N. Digital signatures. In *Advances in Computers*. 2021, Vol. 121, pp. 95-107.

11. Zhang, P., Wang, L., Wang, W., Fu, K., & Wang, J. A blockchain system based on quantum-resistant digital signature. *Security and Communication Networks*, *2021*(1), 6671648.

12. Leng, J., Zhou, M., Zhao, J. L., Huang, Y., & Bian, Y. Blockchain security: A survey of techniques and research directions. *IEEE Transactions on Services Computing*, 2020, *15*(4), pp. 2490-2510.

13. Mehibel, N., & Hamadouche, M. H. A new enhancement of elliptic curve digital signature algorithm. *Journal of Discrete Mathematical Sciences and Cryptography*, 2020, *23*(3), pp. 743-757.

14. Basha, S. J., Veesam, V. S., Ammannamma, T., Navudu, S., & Subrahmanyam, M. V. V. S. Security enhancement of digital signatures for blockchain using EdDSA algorithm. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)* 2021, February, pp. 274-278.

15. Zou, X., & Zeng, P. A new digital signature primitive and its application in blockchain. *IEEE Access*, 2023, *11*, 54607-54615.

16. BIP 340 (Schnorr Signatures). [Електронний ресурс]. – Режим доступу: https://river.com/learn/terms/b/bip-340-schnorr-signatures/

17. Buchanan, William J. *Ed25519 - Edwards-curve Digital Signature Algorithm (EdDSA) using RFC 8032*, 2024. [Електронний ресурс]. - Режим доступу: https://asecuritysite.com/encryption/eddsa2

18. Maxwell, G., Poelstra, A., Seurin, Y., & Wuille, P. Simple schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 2019, *87*(9), pp. 2139-2164.

19. Neven, G., Smart, N. P., & Warinschi, B.. Hash function requirements for Schnorr signatures. *Journal of Mathematical Cryptology*, 2009, *3*(1), pp. 69-87.

20. Uganya, G., & Baskar, R. Modified Elliptic Curve Cryptography Multi-Signature Scheme to Enhance Security in Cryptocurrency. *Computer Systems Science & Engineering*, 2023, *45*(1).

21. Cai, Z., Liu, S., Han, Z., Wang, R., & Huang, Y. (2021). A quantum blind multi-signature method for the industrial blockchain. *Entropy*, *23*(11), 1520.

22. Li, X., Mei, Y., Gong, J., Xiang, F., & Sun, Z. A blockchain privacy protection scheme based on ring signature. *IEEE Access*, 2020, *8*, 76765-76772.

23. Perera, M. N. S., Nakamura, T., Hashimoto, M., Yokoyama, H., Cheng, C. M., & Sakurai, K. (2022). A survey on group signatures and ring signatures: Traceability vs. anonymity. *Cryptography*, 2020, *6*(1), 3.

24. Chatterjee, R., Garg, S., Hajiabadi, M., Khurana, D., Liang, X., Malavolta, G., ... & Shiehian, S. (2021). Compact ring signatures from learning with errors. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41*, pp. 282-312.

25. Devidas, S., Rao YV, S., & Rekha, N. R. A decentralized group signature scheme for privacy protection in a blockchain. *International Journal of Applied Mathematics and Computer Science*, 2021, *31*(2).

26. Zhao, Y. Practical aggregate signature from general elliptic curves, and applications to blockchain. In *Proceedings of the 2019 ACM asia conference on computer and communications security*. 2019, July, pp. 529-538.

27. Li, X., Mei, Y., Gong, J., Xiang, F., & Sun, Z. A blockchain privacy protection scheme based on ring signature. *IEEE Access*, 2020, *8*, 76765-76772.

28. Boneh, D., Drijvers, M., & Neven, G. Compact multi-signatures for smaller blockchains. In *International Conference on the Theory and Application of Cryptology and Information Security*. 2018, October, pp. 435-464.

29. Maxwell, G., Poelstra, A., Seurin, Y., & Wuille, P. Simple schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 2019, *87*(9), 2139-2164.

30. Lu, X., & Feng, D. Quantum digital signature based on quantum one-way functions. In *The 7th International Conference on Advanced Communication Technology, 2005, ICACT 2005*, 2005, February, Vol. 1, pp. 514-517.

31. Rivest, R.L., Shamir, A. and Tauman, Y. How to leak a secret. In International Conference on the Theory and Application of Cryptology and Information Security. 2001, December, pp. 552–565. Springer, Berlin, Heidelberg. https://link.springer.com/content/ pdf/10.1007/3-540-45682-1_32.pdf

32. Kokoris Kogias, E., Malkhi, D., & Spiegelman, A. Asynchronous Distributed Key Generation for Computationally-Secure Randomness, Consensus, and Threshold Signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, October, pp. 1751-1767.

33. Tomescu, A., Chen, R., Zheng, Y., Abraham, I., Pinkas, B., Gueta, G. G., & Devadas, S. Towards scalable threshold cryptosystems. In *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, May, pp. 877-893.

34. Genç, Y., & Afacan, E. Design and implementation of an efficient elliptic curve digital signature algorithm (ECDSA). In *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*, 2021, April, pp. 1-6.

35. Шухманн В.А., Яцків Н.Г. Алгоритми цифрового підпису для захисту транзакцій в мережі біткоїн. Матеріали XVI Всеукраїнська науково-практична конференція “Актуальні проблеми комп’ютерних наук (АПКН – 2024)”, м.Хмельницький, ХНУ, 15-16 листопада 2024. – С. 559-561.

36. Шухманн В., Смирнов Д., Кондратюк В. Порогова схема цифрового підпису для захисту анонімності в блокчейні. Матеріали проблемно-наукової міжгалузевої конференції «Кібербезпека та комп’ютерно-інтегровані технології» (КБКІТ - 2024), Тернопіль, 2024. – С.78-81.

## ДОДАТОК А

Код програми реалізації агрегованого підпису

```
from hashlib import sha256
from random import randint
# Генерація ключів учасників
def generate_keypair():
    # Просте моделювання ключів для схеми
    p = 2**256 - 189  # Просте число для модульних операцій
    sk = randint(1, p - 1)  # Приватний ключ
    pk = sk % p  # Публічний ключ
    return sk, pk, p
# Генерація підпису
def sign_message(private_key, message, p):
    # Хешуємо повідомлення
    hashed_message = int.from_bytes(sha256(message.encode()).digest(), byteorder='big') % p
    # Підпис = хеш^приватний_ключ mod p
    signature = pow(hashed_message, private_key, p)
    return signature
# Агрегування підписів
def aggregate_signatures(signatures, p):
    # Перемножуємо всі підписи за модулем p
    aggregate_signature = 1
    for sig in signatures:
        aggregate_signature = (aggregate_signature * sig) % p
```

```
    return aggregate_signature
# Перевірка агрегованого підпису
def verify_aggregate_signature(aggregate_signature, messages, public_keys, p):
    # Обчислення загальної перевірочної умови
    left_side = pow(aggregate_signature, 1, p)
    right_side = 1
    for message, public_key in zip(messages, public_keys):
        hashed_message = int.from_bytes(sha256(message.encode()).digest(), byteorder='big') % p
        right_side = (right_side * pow(hashed_message, public_key, p)) % p
    return left_side == right_side
# Приклад роботи
if __name__ == "__main__":
    # Генерація ключів для 4 учасників
    keys = [generate_keypair() for _ in range(4)]
    private_keys = [key[0] for key in keys]
    public_keys = [key[1] for key in keys]
    p = keys[0][2]  # Просте число одне для всіх
    # Повідомлення для кожного учасника
    messages = ["Message 1", "Message 2", "Message 3", "Message 4"]
    # Генерація підписів
    signatures = [sign_message(sk, msg, p) for sk, msg in zip(private_keys, messages)]
    # Агрегування підписів
    aggregated_signature = aggregate_signatures(signatures, p)
    # Перевірка
```

```
    is_valid = verify_aggregate_signature(aggregated_signature, messages, public_keys, p)

    print("Публічні ключі:", public_keys)

    print("Підписи:", signatures)

    print("Агрегований підпис:", aggregated_signature)

    print("Перевірка:", "Успішно" if is_valid else "Не успішно")
```

**ДОДАТОК А**

Копії публікацій

Міністерство освіти і науки України
Хмельницький національний університет


ЗБІРНИК НАУКОВИХ ПРАЦЬ
за матеріалами XVI Всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2024»

*15-16 листопада 2024*

Хмельницький 2024

УДК 004:37:001:62

Збірник наукових праць за матеріалами XVI Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2024». Хмельницький. 2024. 582с.

У збірнику наукових праць подані перспективні практичні розробки аспірантів, студентів та здобувачів в області сучасних інформаційних технологій. Розглянуто актуальні проблеми комп'ютерних наук, комп'ютерної інженерії, прикладної математики й інженерії програмного забезпечення, приведено ряд робіт по впровадженню інформаційних технологій у виробництво та управління. Висвітлено перспективні розробки сучасних систем пошуку, обробки й захисту інформації, медійних та комунікаційних системи.

УДК 004:37:001:62

Матеріали конференції відтворені з авторських оригіналів, друкуються в авторській редакції та наведені в алфавітному порядку прізвищ авторів. При макетуванні можливі незначні зміни компоновки контенту авторських оригіналів. Відповідальність за якість та зміст публікацій несе автор.

Участь у конференції та складові всіх її етапів (розгляд праць, перевірка на плагіат, макетування, публікація збірника наукових праць та видача сертифікатів) є безкоштовними для всіх учасників. Оргкомітет конференції висловлює подяку учасникам конференції та сподівається на подальшу співпрацю.

З питань проведення конференції та подальшого обміну інформацією звертатись на e-mail конференції: *apkt.khnu@gmail.com*

УДК 004.56

Шухман В.А., Яцків Н.Г.

*Західноукраїнський національний університет*

# АЛГОРИТМИ ЦИФРОВОГО ПІДПИСУ ДЛЯ ЗАХИСТУ ТРАНЗАКЦІЙ В МЕРЕЖІ БІТКОЇН

*Розглянуто алгоритми цифрового підпису, які лежать в основі забезпечення безпеки та цілісності транзакцій в мережі Біткойн. Проведено аналіз застосування мультипідпису в технології блокчейні та виділено їх переваги.*

*A comparative analysis of digital signature algorithms employed to safeguard the security and integrity of Bitcoin transactions has been performed. The study has focused on the application of multisignature schemes in blockchain technology and has identified their key benefits.*

Цифрові підписи є важливим елементом безпеки в блокчейн-технологіях, оскільки вони забезпечують надійний захист транзакцій, аутентифікацію користувачів і захист від шахрайства. Їх актуальність обумовлена такими ключовими причинами [1]:

1) забезпечення аутентифікації та авторизації**.** Цифровий підпис дозволяє підтвердити, що транзакція була ініційована дійсним власником криптовалютного гаманця. Усі транзакції в блокчейні підписуються особистим (приватним) ключем користувача, що гарантує, що жодна стороння особа не може відправити транзакцію від імені власника без доступу до цього ключа;

2) цілісність даних. Цифрові підписи дозволяють верифікувати цілісність транзакцій, забезпечуючи захист від змін після підписання. Будь-які модифікації даних призведуть до недійсності підпису, і транзакція буде відхилена, що гарантує точність інформації в блокчейні;

3) захист від несанкціонованих змін та шахрайства**.** Цифрові підписи знижують ризик несанкціонованих змін транзакцій, забезпечуючи криптографічний захист від підробки. Це особливо важливо у випадку великих фінансових транзакцій, коли необхідно захистити активи від шахрайства чи кібератак;

4) масштабованість і зниження витрат**.** Сучасні алгоритми цифрового підпису, такі як Шнорра, дозволяють об'єднувати кілька підписів у один. Це зменшує розмір транзакцій у блокчейні, оптимізує використання пам'яті, знижує комісії та підвищує ефективність, що особливо актуально для блокчейнів із великою кількістю користувачів;

5) Підвищення конфіденційності та гнучкості. Деякі алгоритми цифрового підпису, як-от мультипідпис (multisig) та порогові підписи, дозволяють гнучко розподіляти контроль над активами, що є важливим для спільного управління

активами. Завдяки мультипідпису або підписам Шнорра транзакції можна налаштувати так, щоб забезпечити конфіденційність і підтвердження від кількох учасників;

6) довіра до децентралізованої мережі. Цифрові підписи створюють криптографічно захищене середовище, де користувачі можуть довіряти один одному без потреби в централізованому управлінні. Завдяки цифровим підписам кожен вузол мережі може верифікувати транзакції, що підтримує чесність і прозорість системи.

Метою роботи є аналіз алгоритмів цифрового підпису, що використовуються в мережі біткойн, що дозволить краще зрозуміти їх значення для забезпечення безпеки блокчейну.

В криптовалютах використовують різні види цифрових підписів, зокрема: мультипідписи, порогові, агреговані та кільцеві підписи [2].

На практиці застосовують також різні алгоритми цифрового підпису, такі як RSA, DSA (Digital Signature Algorithm) і схеми цифрового підпису на основі ECDSA (Elliptic Curve Digital Signature Algorithm) та EdDSA (Edwards-curve Digital Signature Algorithm). RSA є найпоширенішим, однак із розвитком ECC схеми на основі ECDSA стали досить популярними. Це має переваги використання в блокчейнах, оскільки ECC забезпечує той самий рівень безпеки, що й RSA, але займає менше місця. Крім того, генерація ключів відбувається набагато швидше в ECC порівняно з RSA, тому це сприяє загальній продуктивності блокчейну.

У мережі Bitcoin основним алгоритмом цифрового підпису є ECDSA. Зокрема, він використовує еліптичну криву secp256k1. ECDSA є стандартним методом цифрового підпису в мережі Bitcoin, забезпечуючи: аутентифікацію відправника; цілісність транзакцій; захист від несанкціонованих змін транзакцій.

З 2021 року в мережу Bitcoin було введено оновлення Taproot, яке додало підтримку підписів Шнорра. Це стало можливим завдяки активації BIP-340 (Bitcoin Improvement Proposal), який офіційно впровадив підписи Шнорра для транзакцій у Bitcoin [3].

До особливостей підписів Шнорра у Bitcoin необхідно віднести:

– агрегація підписів. Підпис Шнорра дозволяє об'єднувати кілька підписів у один, що зменшує розмір транзакцій;

– конфіденційність. Завдяки можливості агрегування, підписи Шнорра підвищують конфіденційність, оскільки мультипідписні транзакції виглядають як звичайні, без явного вказування кількості учасників;

– оптимізація продуктивності. Підпис Шнорра підвищує ефективність обробки та перевірки підписів.

Таким чином, в даний час в Bitcoin використовуються: ECDSA для традиційних транзакцій та підписи Шнорра для транзакцій з Taproot, які забезпечують кращу масштабованість та конфіденційність.

При цьому мультипідписи мають багато ширше застосування в технології блокчейн, зокрема:

1) спільні гаманці. Для компаній, які мають спільні фонди, мультипідписний гаманець забезпечує безпечний доступ до коштів. Наприклад, благодійний фонд може налаштувати гаманець "2 з 3", щоб витрачати кошти лише за наявності підписів двох членів ради;

2) смарт-контракти. В блокчейнах із підтримкою смарт-контрактів (наприклад, Ethereum) мультипідпис використовується для управління контрактами, які потребують схвалення кількох сторін. Це дозволяє реалізувати складніші сценарії, де витрачання коштів вимагає консенсусу;

3) забезпечення конфіденційності та контролю в DeFi. Мультипідпис часто використовується в децентралізованих фінансах (DeFi) для створення безпечних та спільних скарбниць (treasuries). Це дозволяє спільно керувати фінансами, одночасно забезпечуючи прозорість і контроль на рівні платформи;

4) децентралізовані автономні організації (DAO). DAO використовують мультипідпис для управління спільними активами. Наприклад, для затвердження пропозицій і витрачання коштів потрібен підпис кількох уповноважених учасників DAO, що забезпечує консенсус і захист від односторонніх рішень;

5) управління особистими фінансами для підвищення безпеки. Користувачі можуть застосовувати мультипідпис для власних гаманців для підвищення особистої безпеки. Наприклад, зберігати свої кошти в гаманці з мультипідписом з налаштуванням "2 з 3", де ключі розміщені на різних пристроях (наприклад, телефон, ноутбук, апаратний гаманець);

6) запобігання одностороннім рішенням у командній роботі. Наприклад, у стартапі, де кілька співзасновників мають спільний контроль над капіталом, можна використовувати мультипідпис, щоб ухвалювати фінансові рішення лише за згодою кількох співвласників.

Отже, мультипідпис це важливий інструмент в екосистемі блокчейну для забезпечення спільного управління активами та безпечної взаємодії кількох користувачів з коштами. Завдяки можливості налаштування різних схем підпису, він забезпечує гнучкість і адаптивність до вимог різних користувачів.

Загалом, цифрові підписи є фундаментальним компонентом у захисті цифрових транзакцій, забезпечуючи надійний рівень безпеки, довіри та ефективності в технології блокчейн.

**Перелік посилань**

1. Digital Signature. Режим доступу: https://academy.binance.com/en/glossary/digital-signature.
2. Блокчейн і децентралізовані системи: навч. посібник для студ. закладів вищ. освіти: в 3 частинах. Ч. 2 / П. Кравченко, Б. Скрябін, О. Курбатов, О. Дубініна. - Харків, 2019. – 412с.
3. BIP 340 (Schnorr Signatures). Режим доступу: https://river.com/learn/terms/b/bip-340-schnorr-signatures/



*ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ*
*НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»*
*ГАЛИЦЬКИЙ ФАХОВИЙ КОЛЕДЖ ІМ. В'ЯЧЕСЛАВА ЧОРНОВОЛА*

# *КІБЕРБЕЗПЕКА ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ (КБКІТ – 2024)*

науково-практична конференція
молодих вчених, аспірантів та студентів

26–28 серпня 2024
Тернопіль

*БЕЗПЕКА ІНТЕРНЕТ РЕЧЕЙ*



*КРИПТОГРАФІЧНІ МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ*

УДК 004.056

***Вадим ШУХМАНН, Дмитро СМИРНОВ, Владислав КОНДРАТЮК***

*Західноукраїнський національний університет*

## ПОРОГОВА СХЕМА ЦИФРОВОГО ПІДПИСУ ДЛЯ ЗАХИСТУ АНОНІМНОСТІ В БЛОКЧЕЙНІ

**Вступ**. Технологія блокчейн революціонізує багато галузей, від фінансів до логістики та Інтернет речей. Одним із ключових аспектів безпеки блокчейну є цифровий підпис, який гарантує автентичність та цілісність даних. У контексті децентралізованих систем важливим є поняття порогових схем, які дозволяють розподіляти контроль над приватним ключем між кількома учасниками [1].

Актуальність дослідження обумовлена постійним зростанням інтересу до технології блокчейн та необхідністю забезпечення високого рівня безпеки децентралізованих систем. Вибір оптимального алгоритму цифрового підпису для порогової схеми є критичним рішенням, яке впливає на ефективність і надійність всієї системи [2].

Порогові схеми дозволяють розподіляти повноваження підписанта між кількома учасниками, що забезпечує додатковий рівень захисту та надійності системи. Такі схеми широко застосовуються у фінансових транзакціях, управлінні розподіленими базами даних, блокчейнах та інших системах з високими вимогами до безпеки. Вибір оптимального алгоритму цифрового підпису є важливим завданням при розробці та впровадженні порогових схем, оскільки він впливає на ефективність та швидкодію системи, а також на стійкість до атак.

**Мета:** аналіз ефективності алгоритмів цифрового підпису для побудови порогових схем.

### 1. Порогова схема цифрового підпису

Порогові схеми (threshold schemes) - це криптографічні протоколи, які дозволяють розподілити секретний ключ (наприклад, приватний ключ для цифрового підпису) між кількома учасниками. Для відновлення секретного ключа або створення підпису необхідна участь лише певної кількості учасників (поріг).

Алгоритми цифрового підпису в порогових схемах відіграють ключову роль у забезпеченні безпеки та ефективності таких систем.

В порогових схемах використовуються наступні алгоритми цифрового підпису:

1. ECDSA (Elliptic Curve Digital Signature Algorithm):

- широко використовується в блокчейні завдяки своїй ефективності та високому рівню безпеки;
- в порогових схемах ECDSA може бути використаний з різними протоколами розподілу секретного ключа.

До переваг необхідно віднести: висока швидкість, невеликий розмір підпису.

До недоліків необхідно віднести: може бути вразливий до певних типів атак при неправильній реалізації.

2. Schnorr signatures:

- модернізований варіант алгоритму DSA, який пропонує більш ефективну перевірку підпису;
- забезпечує високий рівень безпеки та добре масштабується для великих кількостей учасників.

До переваг необхідно віднести: швидка перевірка підпису, компактні підписи.

До недоліків необхідно віднести: може бути менш поширеним у порівнянні з ECDSA.

3. BLS signatures (Boneh–Lynn–Shacham):

- базуються на парних спарюваннях на еліптичних кривих.
- дозволяють агрегувати кілька підписів в один, що значно зменшує розмір транзакцій.

До переваг необхідно віднести: компактні агреговані підписи, висока пропускна здатність.

До недоліків необхідно віднести: більш складні в реалізації порівняно з ECDSA та Schnorr signatures.

В таблиці 1 наведено якісне порівняння основних алгоритмів цифрового підпису, які використовуються для побудови порогових схем.

Таблиця 1 - Якісне порівняння алгоритмів цифрового підпису

| **Характеристика** | **ECDSA** | **Schnorr signatures** | **BLS signatures** |
|---|---|---|---|
| Швидкість | Середня | Висока | Висока |
| Розмір підпису | Середній | Середній | Дуже малий (при агрегації) |
| Безпека | Висока | Висока | Висока |
| Складність реалізації | Середня | Середня | Висока |
| Агрегація підписів | Можлива, але менш ефективна | Можлива | Легко |

Порогова схеми є специфічним типом мультипідпису. Вона не покладається на те, що користувачі підписуватимуть повідомлення своїми унікальними ключами; натомість для цього потрібен лише один відкритий ключ і один закритий ключ, що призводить до лише одного цифрового підпису. У режимі мультипідпису кінцеве повідомлення містить цифрові підписи всіх підписувачів і потребує індивідуальної перевірки стороною, що перевіряє, але в порогових підписах перевіряльник перевіряє лише один цифровий підпис. Основна ідея схеми полягає в тому, щоб розділити закритий ключ на кілька частин, і кожен підписувач зберігає свою власну частину закритого ключа. Процес підписання вимагає від кожного користувача використання відповідної частини закритого ключа для підпису повідомлення. У даній схемі лише підмножина підписувачів може створити підпис зі своєю часткою, і немає необхідності, щоб усі учасники співпрацювали для створення підпису.

Порогові підписи також можна використовувати для надання послуг анонімності в мережі блокчейн. Це відбувається тому, що схеми порогового підпису не розкривають членів порогової групи, які підписалися для створення

підпису. Це відрізняється від схем мультипідпису, які розкривають особи всіх підписантів. Крім того, у дозволених мережах порогові підписи можна використовувати в сценаріях, де для узгодження операції потрібен поріг користувачів.

**2. Алгоритм цифрового підпису на основі кривої Едвардса**

Алгоритм цифрового підпису на основі кривої Едвардса (EdDSA) використовується для створення цифрового підпису з використанням покращення підпису Шнорра за допомогою скручених кривих Едвардса. Загалом він швидший, ніж багато інших методів цифрового підпису, і надійний з точки зору безпеки.

Одним із прикладів EdDSA є Ed25519, який базується на кривій 25519. Він генерує 64-байтове значення підпису (R, s) і має 32-байтові значення відкритого та закритого ключів. Розділимо закритий ключ на секретну політику спільного доступу 2 із 3 і розподілимо його між кількома сторонами. Далі кожна зі сторін може створити частину підпису, а потім вони об'єднуються разом для створення загального підпису. Таким чином закритий ключ ніколи не потрібно перебудовувати, щоб створити підпис для об'єкта даних.

Перевагою використання Ed25519 над ECDSA є те, що можна об'єднувати підписи та відкриті ключі [3].

Спочатку кожна сторона створює секрет $(s_i)$ і передає відкритий ключ

$$A_i = s_i \cdot B,$$

де $B$ - базова точка на кривій.

Сторона також генерує випадкове значення $r_i$ та зобов'язується:

$$R_i = r_i \cdot B.$$

Нарешті кожна сторона генерує:

$$S_i = r_i + h \cdot s_i (mod\ l),$$

де $q$ - порядок кривої. Потім кожна сторона зобов'язується $(A_i, R_i, S_i)$ і надсилає іншим сторонам та чекає на відповідь про зобов'язання. З усіма отриманими зобов'язаннями можемо відновити:

$$R = R_1 + R_2 + \cdots R_2,$$

$$S = S_1 + S_2 + \ldots S_n.$$

Тоді відкритий ключ $(A)$:

$$A = A_1 + A_2 + \ldots A_n,$$

який дорівнює:

$$A = s_1 \cdot B + s_2 \cdot B + \ldots s_n \cdot B.$$

Потім перевіряємо за допомогою:

$$S \cdot B = R + k \cdot A.$$

Тоді підпис $(R, S)$.

За допомогою розділення секрету Шаміра змінюємо секретний ключ так, щоб кожна сторона мала значення секретного ключа $(s_i)$ помножене на коефіцієнт Лагранжа $l_i$, який відповідає певній частці.

Потім відновлюємо за допомогою:

$$A_i = s_i \cdot l_i \cdot B,$$

$$R_i = r_i \cdot B,$$

$$S_i = r_i + k \cdot l_i \cdot s_i (mod\ p).$$

Розглянемо приклад. Нехай повідомлення: Vadym Shukhman

Відкритий ключ:
e0c205f73ed6826a948f97330595c70708e4eecd316e4ea593a0c7cd0b7e99b9

Порогове значення Sig1:
e6d7797b689f38a3a7cc24e9624cd2876caf92fbed5dbc04e785c1cd83daca0233cc8fe1cb960b8cb3ff317a889c6888060e233f29cdff80f13cffb711d02a0b.

Порогове значення Sig2:
e6d7797b689f38a3a7cc24e9624cd2876caf92fbed5dbc04e785c1cd83daca02a1c95e2d24a83abea0b4c9fb1030ad63d21d8f05be493fc0ec5aedf2bbac330d.

Порогове значення Sig3:
e6d7797b689f38a3a7cc24e9624cd2876caf92fbed5dbc04e785c1cd83daca020fc72d797cb969f08d69617d99c3f13e9e2dfbcb52c67effe778db2d66893c0f.

Зміна підпису із спільний доступ 1 і 3:
e6d7797b689f38a3a7cc24e9624cd2876caf92fbed5dbc04e785c1cd83daca02c5cec0957385dc59c64a9af8ff0824ad3afeb6789450c041f61e117d67f32109.

Зміна підпису із спільний доступ 2 і 3:
e6d7797b689f38a3a7cc24e9624cd2876caf92fbed5dbc04e785c1cd83daca02c5cec0957385dc59c64a9af8ff0824ad3afeb6789450c041f61e117d67f32109.

Підпис перевірено.

Довжина підпису становить 64 байти, які складаються з 32 байтів для значення R і 32 байтів для значення s. Довжина відкритого ключа також становить 32 байти, а закритого –32 байти. Загалом Ed25519 створює один із найменших розмірів підпису та має невеликий розмір відкритого та закритого ключів.

**Висновок**. В роботі проведемо аналіз ефективності різних алгоритмів цифрового підпису, таких як ECDSA, Schnorr signatures та BLS signatures, у контексті їх використання для побудови порогових схем у технології блокчейні. Розглянуто переваги та недоліки кожного з алгоритмів, їх продуктивність, безпеку та відповідність вимогам сучасних блокчейн-систем. Крім того, наведено приклад обчислення цифрового підпису на основі еліптичної кривої 25519. Загалом Ed25519 створює один із найменших розмірів підпису та має невеликий розмір відкритого та закритого ключів.

**Перелік використаних джерел.**

1. Блокчейн і децентралізовані системи: навч. посібник для студ. закладів вищ. освіти: в 3 частинах. Ч. 2 / П. Кравченко, Б. Скрябін, О. Курбатов, О. Дубініна. - Харків, 2019. - 412 с.

2. Digital Signature. [Електронний ресурс].- Режим доступу: https://academy.binance.com/en/glossary/digital-signature.

3. Buchanan, William J (2024). Ed25519 - Edwards-curve Digital Signature Algorithm (EdDSA) using RFC 8032. [Електронний ресурс]. - Режим доступу: Asecuritysite.com. https://asecuritysite.com/encryption/eddsa2