Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Безпеки інформаційних технологій
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Сучасні методи захисту конфіденційності інформації в децентралізованих системах
(тема)

Виконав:

студент 2 курсу, групи БІКСм-20-1

Кустов А. К.
(прізвище, ініціали)

Спеціальність 125 Кібербезпека
(код і повна назва спеціальності)

Освітня програма «Безпека інформаційних і комунікаційних систем»
(повна назва освітньої програми)

Керівник доц. Власов А. В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Халімов Г.З.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Безпеки інформаційних технологій
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 125 Кібербезпека
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна, або освітньо-наукова)

Освітня програма «Безпека інформаційних і комунікаційних систем»
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри: _____ Халімов Г.З.
(підпис)
«____»_____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Кустов Андрій Костянтинович
(прізвище, ім'я, по батькові)

1. Тема роботи Сучасні методи захисту конфіденційності інформації в децентралізованих системах
затверджена наказом по університету від 08.11.2021 р. № 1685 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 грудня 2021 р.

3. Вихідні дані до роботи
Проаналізовані та порівняні дані про методи захисту конфіденційної інформації в децентралізованих системах

4. Перелік питань, що потрібно опрацювати в роботі
Поняття конфіденційності; конфіденційність у різних децентралізованих системах, методи захисту конфіденційності в децентралізованих системах; конфіденційність користувачів в мережі; переваги та недоліки методів захисту конфіденційності в децентралізованих системах

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) презентаційний матеріал у вигляді слайдів

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Вибір здобувачем теми кваліфікаційної роботи	02.09.2021	Виконано
2	Затвердження плану і завдання кваліфікаційної роботи	09.11.2021	Виконано
3	Аналіз завдання, пошук та аналіз літературних джерел за темою роботи	10.11.2021-18.11.2021	Виконано
4	Виконання кваліфікаційної роботи	19.11.2021-30.11.2021	Виконано
5	Оформлення пояснювальної записки	01.12.2021-12.12.2021	Виконано
6	Здача на перевірку та підпис кваліфікаційної роботи керівнику	10.12.2021	Виконано
7	Проходження перевірки на плагіат та нормоконтроль кваліфікаційної роботи	10.12.2021	Виконано
8	Допуск завідувачем кафедри до захисту кваліфікаційної роботи	13.12.2021	Виконано
9	Захист кваліфікаційної роботи	14.12.2021	Виконано

Дата видачі завдання _____ _____20__ р.

Студент _____
(підпис)

Керівник роботи _____ __доцент Власов А. В.__
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка включає в себе 88 с., 5 табл., 39 рис., 38 джерел.

БЛОКЧЕЙН, ДЕЦЕНТРАЛІЗОВАНІ СИСТЕМИ, ПРИВАТНІСТЬ, BITCOIN, КОНФІДЕНЦІЙНІСТЬ, ТРАНЗАКЦІЯ, МЕРЕЖА, ГРАФ, BLIND SIGNATURE, COINJOIN, COINSHUFFLE, MIMBLEWIMBLE

Об’єкт дослідження – захист інформації в децентралізованих системах.

Предмет дослідження – методи захисту конфіденційності інформації в децентралізованих системах.

Мета роботи – розгляд та аналіз методів, що забезпечують конфіденційність даних, які формуються та обробляються в децентралізованих системах (адреси, підписи, транзакції тощо) та конфіденційність самих користувачів у децентралізованих системах.

Методи дослідження – аналіз інформації щодо побудови та впровадження методів захисту конфіденційності інформації та користувачів з моделюванням їх застосування в тестових децентралізованих системах.

ABSTRACT

The explanatory note includes 88 pages, 5 tables, 39 figures, 38 sources.

BLOCKCHAIN, DECENTRALIZED SYSTEMS, PRIVACY, BITCOIN, CONFIDENTIALITY, TRANSACTION, NETWORK, GRAPH, BLIND SIGNATURE, COINJOIN, COINSHUFFLE, MIMBLEWIMBLE

The object of research is the protection of information in decentralized systems.

The subject of research – methods of protecting the confidentiality of information in decentralized systems.

The purpose of the work is to consider and analyze the methods that ensure the confidentiality of data generated and processed in decentralized systems (addresses, signatures, transactions, etc.) and the confidentiality of users in decentralized systems.

Research methods – analysis of information on the construction and implementation of methods for protecting the confidentiality of information and users with modeling of their application in test decentralized systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

IT – інформаційна технологія

API – Application Programming Interface, інтерфейс створення додатків

CA – Certification Authority, центр сертифікації

DID – Decentralized ID, система децентралізованої ідентифікації

DNS – Domain Name System, система доменних імен

GDPR – General Data Protection Regulation, захист персональних даних на території Європейського Союзу

PII – Personal Identifiable Information, особиста ідентифікаційна інформація

URL – Uniform Resource Locator, система уніфікованих адрес

ZKP – Zero-Knowledge Proof, доведення з нульовим розголошенням

R1CS – Rank-1 Constraint System, система обмежень рангу

QAP – Quadratic Arithmetic Programs, програми квадратичної арифметики

CTs – Confidential Transactions, конфіденційні транзакції

UTXO – Unspent Transaction Output, вивід невикористаних транзакцій

VPN – Virtual Private Network, віртуальна приватна мережа

TTP – Trusted Third Party, довірена третя сторона

PoW – Proof-of-Work, доказ виконаної роботи

DDoS – Denial of Service, відмова в обслуговуванні

SSH (Secure Shell) – мережевий протокол прикладного рівня, що дозволяє забезпечувати віддалене керування операційною системою та тунелювання TCP-з'єднань.

Off-chain транзакції – це транзакції, які виконуються користувачами у криптовалютній мережі, але поза основним блокчейном.

Блокчейн – англ. Blockchain – це розподілена база даних, що зберігає впорядкований ланцюжок записів (так званих блоків), що постійно довшає.

Блок – англ. Block – це базова складова блокчейну. Усі блоки, створені раніше і створювані прямо зараз, формують ланцюжок блоків.

Децентралізація – процес перерозподілу функцій, повноважень, людей або речей від центрального управління.

Криптовалюта – англ. Cryptocurrency – різновид цифрової валюти, емісія та облік якої виконується децентралізованою платіжною системою повністю в автоматичному режимі (без можливості внутрішнього або зовнішнього адміністрування).

Біткоін – англ. Bitcoin – перша і найвідоміша з безлічі інших криптовалют (набір даних, ретельно захищений від злому і копіювання за допомогою різних криптографічних методів захисту).

Транзакція – операція, яка полягає в переведенні коштів з одного рахунку на інший.

Хеш-деревом, деревом Меркла – англ. Merkle Tree – називають повне двійкове дерево, у листові вершини якого поміщені хеші від блоків даних, а внутрішні вершини містять хеші від складання значень у дочірніх вершинах.

ВСТУП

Сьогодні використання аналітичних можливостей дозволяє компаніям не лише враховувати звички користувачів, а й впливати на їхню поведінку. Все це стало можливим завдяки обробці даних про покупки та інші дії користувачів у мережі. Ці дані приносять компаніям великі доходи. Більше того, якщо проаналізувати отриману компаніями інформацію, то більшість з неї можливо віднести до конфіденційної. Тому питання приватності у цифровому світі займає одне з перших місць серед тих, на які слід звертати увагу. А враховуючи розвиток технологій та аналітичних інструментів, це питання стає дедалі актуальнішим.

Припустимо, на одній вулиці є декілька ресторанів. Вони конкурують за потік відвідувачів і кожен прагне отримати більший дохід. Відомості про внутрішні процеси ціноутворення та доходи залишаються таємницею для конкурентів. Однак, що буде, якщо хтось опублікує відомості про всі внутрішні транзакції та іншу документацію цих ресторанів у відкритий доступ. Менш успішні конкуренти зможуть керуючись нечесними намірами нашкодити найбільш успішному конкуренту матеріально. За умови, що найбільш успішний ресторан був найпривабливішим для відвідувачів, оскільки там готували смачні страви з якісних продуктів, від руйнування цього закладу ніхто не виграє, окрім менш успішних конкурентів.

Питання конфіденційності актуальне і в галузі наукових досліджень. Може статися так, що група вчених, які зробили значні відкриття, забажають залишитися анонімними (як у випадку з Bitcoin), наприклад, пояснюючи це тим, що вони працювали у великій команді, кожен учасник якої зробив свій внесок у відкриття. У зв'язку з цим може стати актуальним питання щодо можливостей легального анонімного фінансування наукових досліджень для розвитку незалежних та вільних від цензури дослідницьких інститутів.

Саме з обставин актуальності забезпечення конфіденційності та захисту даних користувачів в кваліфікаційній роботі розглянуто методи та інструменти захисту конфіденційності користувачів та даних, що обробляються (циркулюють) в децентралізованих системах.

У першому розділі розглянуті поняття конфіденційності та її складові.

У другому розділі детально розглянуті стандарти, протоколи та методи захисту конфіденційності інформації в децентралізованих системах.

У третьому розділі проаналізовані підходи до забезпечення конфіденційності користувачів в децентралізованих системах. Також буде обрано один із методів для подальшої реалізації.

В четвертому розділі наведено опис програмної реалізації одного із механізмів захисту конфіденційності в децентралізованих системах.

1 ПОНЯТТЯ КОНФІДЕНЦІЙНОСТІ ТА ЇЇ СКЛАДОВІ

1.1 Складові конфіденційності

Поняття конфіденційності включає дві основні складові: untraceability і anonymity. Untraceability, або непростежуваність, має на увазі неможливість віднести групу дій до деякого користувача в мережі. Anonymity, або анонімність, пов'язана з неможливістю достовірно встановити особу користувача в цій мережі. На рис. 1.1 наведено приклад типової людини, яка бажає захистити себе а, отже, і свою приватність у мережі.

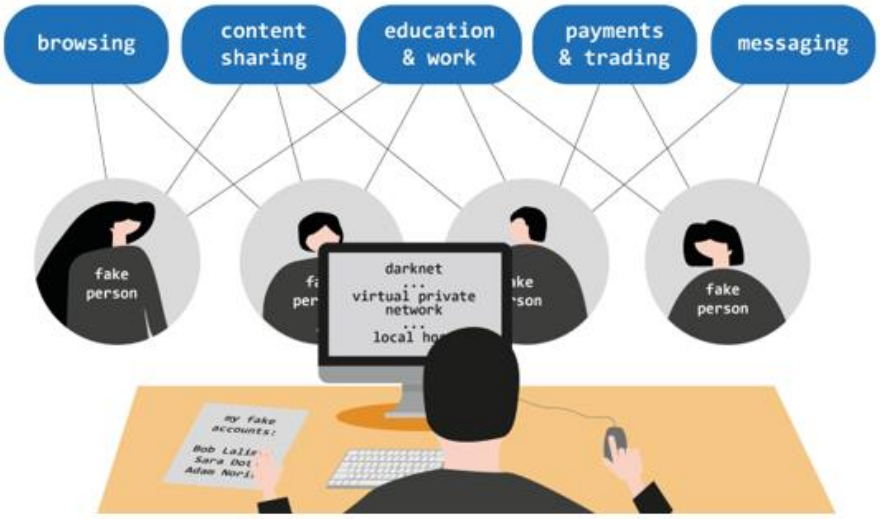

Рисунок 1.1 – Приклад типової людини, яка бажає захистити свою приватність у мережі

Поняття анонімності визначає наявність соціального та технічного компоненту. Соціальна анонімність включає відомості, які може вказувати користувач про свою особистість. Технічна анонімність значно складніша і основні зусилля щодо забезпечення анонімності докладаються саме в цій галузі.

Що ж до підвищення рівня приватності, це може забезпечуватися спеціальними механізмами приховування наступних атрибутів активності користувачів:

– прямий зв'язок між усіма діями, повідомленнями, транзакціями;

– версії додатків та операційної системи;

– локалізації додатків;

– тимчасові мітки повідомлень та запитів;

– ідентифікатори (username, email, phone number);

– у деяких випадках сам факт дії;

– розташування пристроїв;

– мережеві адреси пристроїв.

Деякі реалізації Tor-браузеру та BitTorrent-клієнтів вже підтримують розширені опції конфіденційності, які допомагають приховувати деякі з перерахованих атрибутів. Методики можуть бути дійсно специфічними, наприклад, використання випадкових портів на мережевому рівні, висока надмірність запитів, зміна роздільної здатності екрану за умовчанням.

Аналіз функціонування децентралізованої мережі Bitcoin [1] свідчить про те, що фінансова система може бути прозорою і легкою у перевірці для всіх учасників за умови збереження певного рівня конфіденційності користувачів. Проте стало зрозуміло, що для деяких варіантів використання рівень прозорості, що забезпечується технологією blockchain, є неприйнятним. Таким чином, нові методи для забезпечення вищого рівня конфіденційності користувачів були запропоновані та навіть реалізовані в деяких проектах криптовалют. Ці методи ефективно вирішують питання конфіденційності транзакцій, залишаючи їх публічно перевіреними та незмінними.

Розвиток децентралізованих систем та цифрових валют ускладнюється проблемами масштабованості та приватності користувачів. Більше того, будь-яка система обліку активів, яка використовує відкриту базу даних, стикається з проблемою конфіденційності [2]. А якщо мова йде про криптовалюти, то там принцип повної прозорості історії транзакцій взагалі є базовим.

1.2 Конфіденційність у децентралізованих системах

Поширення децентралізованих систем та їх розвиток визначені в першу чергу наявністю властивості анонімності (користувачів, їх дій та відповідно даних). Але цю властивість потрібно гарантувати та забезпечити, що не й так дуже легко зробити на практиці [3]. Не досягається повною мірою і властивість невідстежуваності (untraceability) користувачів. Зловмисник має можливість проаналізувати граф транзакцій і зробити висновок щодо відношення цього графу до певних анонімних гаманців [4]. Якщо хоча б одна адреса була скомпрометована у контексті анонімності, зловмисник має можливість встановити зв'язок з певними користувачами. Найпростіші реалізації прикладних додатків (наприклад гаманців) здатні забезпечити лише мінімальний рівень конфіденційності.

Припустимо, що для кожного вхідного платежу (формування нових даних) користувач створює нову адресу. Аудитор, який аналізує граф транзакцій, у цьому випадку вже може достовірно пов'язати конкретні факти, які стосуються дій користувачів і розподілу даних (криптовалюти) поміж них. Але навіть у такій ситуації рівень конфіденційності користувача не занадто високий, як цього потребують користувачі.

В мережі Bitcoin приватність користувача залежить від його контрагентів. Одержувач платежу знатиме історію походження монет, а відправник монет знатиме, кому він їх передає та відповідну адресу. Більше того, зможе простежувати історію подальшої передачі монет.

Генерація нових адрес ускладнює можливість відстеження даних та їх взаємозв’язків (ланцюжків блоків та транзакцій), але робить їх абсолютно непростежуваними. Існують метадані, які можуть бути доступними стороннім користувачам: характер угоди, дані про гаманець, дані про знаходження користувача тощо. Крім того, є методи, алгоритми, аналізатори, які дозволяють побудувати дерево з ланцюжків, знайти залежності за часом, по IP-адресам, за способом побудови транзакцій та з певною ймовірністю можуть

дати досить наближену до реальності картину розподілу монет [5]. Крім того, якщо користувач захоче витратити монети, які зберігаються на кількох адресах, то за допомогою транзакції, яка містить безліч входів, що належать одному користувачу, виникне можливість повністю деанонімізувати його.

Одним із найпростіших способів заплутування історії транзакцій є використання алгоритмів «міксерів» [6]. Ці алгоритми дозволяють змішати разом монети окремих користувачів. Для цього складається одна транзакція, а на її вхід подаються монети від різних власників. Монети з усіх входів підсумовуються, при цьому змішуючись, а потім розподіляються між усіма виходами, для яких знову вказуються отримувачі. У такому разі неможливо однозначно визначити історію передачі монети, тому що історії змішалися разом, а потім довільним чином розійшлися. Таке змішування у загальних транзакціях можна виконувати навіть кілька разів поспіль. У результаті ми отримаємо монети із неоднозначною історією. Однак цей спосіб не єдиний і має певні недоліки. Наприклад, деякі сервіси не приймають монети, які проходять через «міксер».

Щоб забезпечити максимальний рівень конфіденційності, потрібно розуміти, які дані про транзакцію слід приховувати у першу чергу.

Дані про транзакції, які слід в першу чергу захищати:

– історія походження монет;

– суми платежів;

– адреси (ідентифікатори) користувачів;

– мережеві адреси пристроїв користувачів;

– метадані (час розповсюдження по мережі, версія гаманця, тощо).

Також до цього списку можна віднести дані про походження монет, які безпосередньо впливають на властивість взаємозамінності (fungibility), дуже важливої для грошей та цінностей. На рівні протоколу Bitcoin ця властивість забезпечується (всі монети однакові та правила їх обробки спільні для всіх), але на практиці функціональність легко порушити. Наприклад, деякі сервіси

можуть аналізувати історію походження монет і відхиляти платежі, якщо вона викликає у них сумніви.

Наступне, що має сенс приховувати, це суми платежів, адреси відправника та отримувача у тілі транзакції. Важливо також приховати мережеві адреси користувачів, що зазвичай досягається за допомогою даркнетів, де використовуються такі протоколи, як Freenet, TOR і I2P. Виникає питання: як приховати відомості про дані та їх історію (суми, історію та адреси)?

В першому розділі дане визначення поняття конфіденційності, а також розглянуто її складові. Також розглянуто питання конфіденційності в різних децентралізованих системах та визначено які дані вважаються конфіденційними.

В наступному розділі детально розглянуті різні методи захисту конфіденційності користувачів, які використовуються в сучасних децентралізованих системах. А також наведено опис механізмів захисту конфіденційності користувачів у мережі.

2 МЕТОДИ ЗАБЕЗПЕЧЕННЯ КОНФІДЕНЦІЙНОСТІ В СУЧАСНИХ ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМАХ

В даному розділі розглянуто технологічні аспекти щодо забезпечення конфіденційності користувачів, які впроваджені в децентралізованих системах [7]:

– підходи, засновані на комбінуванні даних. Дані підходи використовують вхідні та вихідні дані про різні операції з подальшим їх об'єднанням в єдину велику операцію для приховування зв'язків між адресами відправників та одержувачів. Такі підходи до підвищення конфіденційності реалізовано в протоколах: CoinJoin, Mimblewimble та Monero;

– конфіденційність, заснована на нульовому розголошенні. Це технологія, в якій користувачі надають докази з нульовим розголошенням - дані, які свідчать про знання якоїсь інформації без розкриття самої інформації. При правильному використанні дана криптографічна технологія здатна забезпечувати конфіденційність транзакцій та станів мережі, і вірність блокчейну. Прикладом реалізації є Blind Signature, zk-SNARK.

2.1 Метод CoinJoin та його модифікації

Найпростіший метод для анонімізації транзакцій називається CoinJoin. Його суть полягає у створенні спільної транзакції, внаслідок чого походження монет, що відправляються, стає неоднозначним [8]. Формується група з користувачів, які створюють спільну транзакцію, в рамках якої одночасно відбувається кілька платежів. Тобто, користувачам не потрібно створювати окремі транзакції.

Уявімо групу із трьох користувачів, у якій кожен користувач має намір придбати товар в інтернет-магазині. При цьому для кожного з них свій магазин. Користувачі створюють одну транзакцію на три входи, по одному від

кожного користувача, три виходи, по одному на кожен інтернет-магазин. Крім цього, створюються ще три виходи для решти (рис. 2.1).

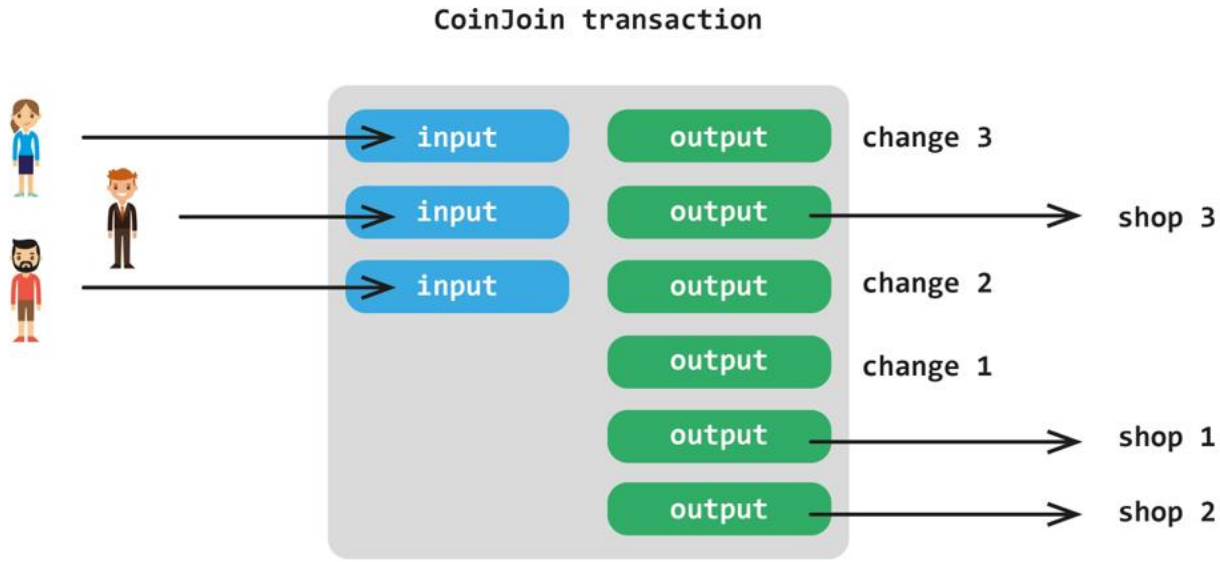

Рисунок 2.1 – Схема роботи методу CoinJoin

Далі всі виходи перемішуються між собою випадковим чином. Кожен користувач перевіряє ще раз отриману транзакцію і підписує відповідний йому вхід. У разі успіху транзакція вважається правильною, розповсюджується по мережі та отримує підтвердження.

На рисунку 2.2 зображено два графи: верхній та нижній. Для верхнього графа характерно, що кожна транзакція має один або два виходи, а для нижнього – транзакції можуть мати три і більше виходів. Нижній граф більш заплутаний та складніше піддається аналізу.

Коли в біткоін-гаманці метод CoinJoin застосовується практично, формується численна група користувачів. Тоді транзакції можуть мати десятки входів та виходів (іноді більше). Зображений на площині, граф таких транзакцій вийде дуже заплутаним (рис. 2.2). Монета, яка пройшла ланцюжок таких транзакцій, має тисячі можливих варіантів походження. Важко знайти серед усіх варіантів реальний.

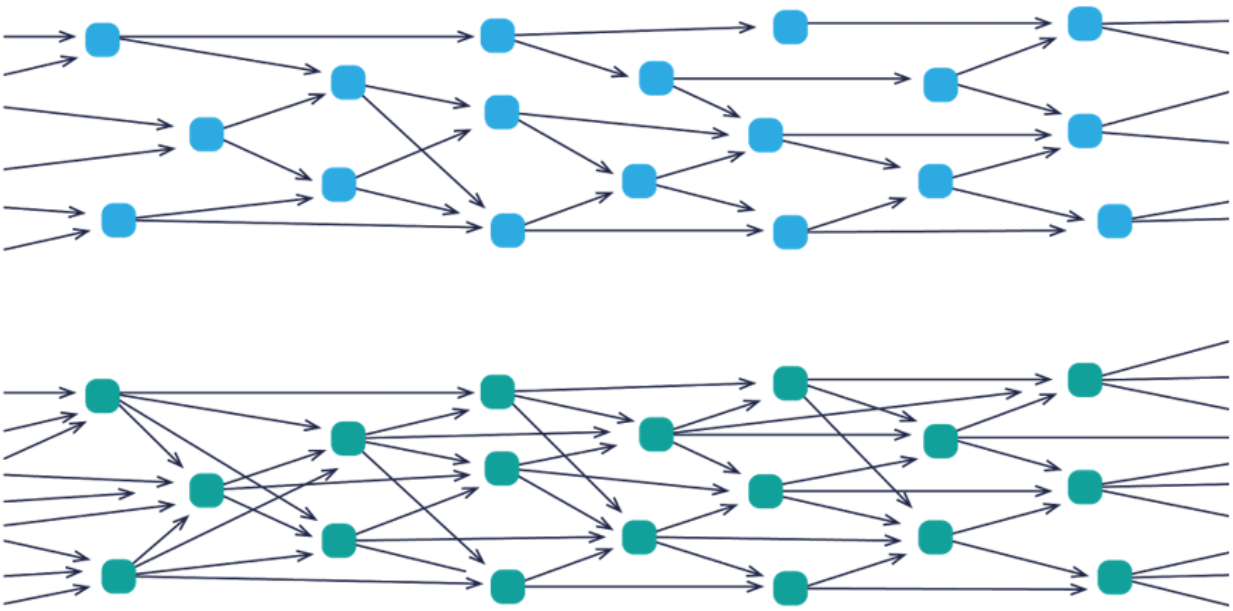

Рисунок 2.2 – Два графи: верхній (стандартний) та нижній (заплутаний методом CoinJoin)

Далі будуть розглянуті різні модифікації методу CoinJoin.

2.1.1 Недоліки методу CoinJoin

Існує велика складність off-chain взаємодій для формування транзакції, тобто необхідно організувати формування груп та взаємодію користувачів між собою.

Але вагоміший недолік полягає в тому, що CoinJoin у чистому вигляді не приховує суми платежів. Як результат, він уразливий до CoinJoin Sudoku analysis, який заснований на зіставленні сум на виходах транзакцій і дозволяє розплутати історію походження монет після її багаторазового заплутування. З цією проблемою можна боротися, наприклад, використовувати для вихідних значень транзакцій лише певні суми (0,1 BTC, 1 BTC, 10 BTC тощо), але це створює додаткові складності та обмеження.

Іншим способом, що вирішує проблему відкритості сум переказів, є концепція Confidential Transactions (див. підроз. 2.4) та Zero-knowledge Proofs (див. підроз. 2.3).

2.1.2 Метод Chaumian CoinJoin

Одна з модифікацій CoinJoin називається Chaumian CoinJoin [8]. В даній модифікації застосовується централізований оператор і використовується сліпий підпис. Оператор потрібен для того, щоб виконати перемішування входів та виходів, після чого скласти кінцеву транзакцію. Але оператор не може вкрасти монети або порушити конфіденційність перемішування завдяки сліпому підпису.

Користувач попередньо «засліплює» дані до їх передачі оператору (використовуючи механізм сліпого підпису). Коли оператор підписує ці дані, він не має доступу до фактичного вмісту. Підписані дані повертаються користувачеві, після чого він прибирає «засліплення» і все виглядає, як звичайний цифровий підпис.

Як відбувається взаємодія між користувачем та оператором при формуванні спільної транзакції? Кожен користувач заздалегідь готує вхід, де витрачаються монети, що належать йому, адресу для отримання решти, а також «засліплену» адресу для відправки платежу, після чого об'єднує ці дані в одну послідовність і передає оператору (рис. 2.3).

Оператор перевіряє вхід та суму платежу, підписує вихідну адресу та повертає підпис користувачеві. При цьому оператор не має доступу до адреси, на яку користувач хоче відправити платіж, оскільки він «засліплений». Далі користувач прибирає «засліплення» з вихідної адреси, анонімно перепідключається до оператору та передає йому підписану вихідну адресу. Оператор, у свою чергу, перевіряє, що він дійсно підписував цю адресу своїм ключем і відповідний вхід у нього вже є, але при цьому він не може знати, який вхід відповідає якому виходу. Після того, як усі користувачі виконали такі дії, вони знову анонімно перепідключаються до оператору та надають підписи, які підтверджують володіння монетами на вході загальної транзакції. Готову транзакцію можна розповсюджувати по мережі для підтвердження.

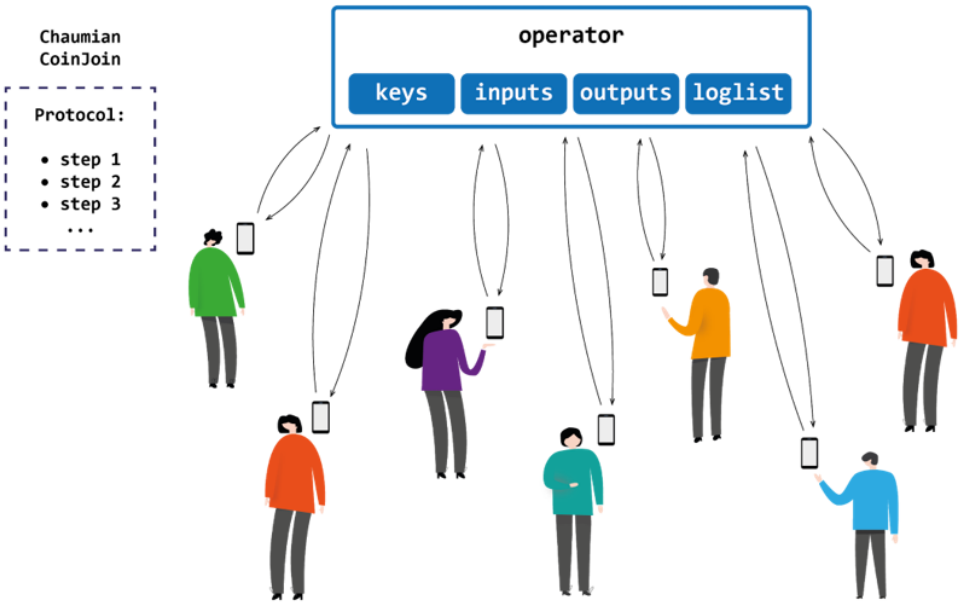

Рисунок 2.3 – Загальна схема роботи методу Chaumian CoinJoin

У такому разі ні користувачі, ні сам оператор не можуть деанонімізувати монети на вихідних адресах. А формування транзакції за нормальних умов займає не більше однієї хвилини. Взаємодія користувачів повинна проводитися через анонімні мережі передачі даних, якими можна використовувати Tor, I2P або Bitmessage (див. підроз. 2.7).

Серед користувачів можуть бути зловмисники, мета яких порушити процес створення спільної транзакції будь-якими способами. Існує цілий список можливих сценаріїв поведінки користувачів, у тому числі шахрайський.

2.1.3 Метод CoinShuffle

Модифікацію CoinShuffle запропонували у 2014 році [9]. В даній модифікації не використовується центральній оператор, і це є перевагою в порівнянні з методом Chaumian CoinJoin.

В методі CoinShuffle користувачі самостійно формують спільну транзакцію, спілкуючись між собою. При цьому вони однаково не можуть порушити конфіденційність перемішування вихідних адрес. Ще одна перевага цього методу полягає в тому, що користувачам не обов'язково

використовувати додаткові мережі для анонімізації трафіку, тому що всі необхідні властивості будуть досягнуті при використанні одного протоколу взаємодії P2P користувачів.

У даному випадку застосовується спрямоване шифрування, де використовується пара ключів (відкритий і особистий). Повідомлення шифрується за допомогою відкритого ключа, а розшифрувати його може лише власник особистого ключа. Для комунікації між учасниками використовується протокол DiceMix, у якому також передбачено протистояння порушникам.

Уявимо невелику групу користувачів. Кожен з них має одну невитрачену монету в обліковій системі Bitcoin за адресами A, B, C і D відповідно (рис. 2.4).

Кожен учасник хоче витратити монету і приховати історію її походження. З цією метою кожен учасник групи дізнається адресу, на яку має бути відправлено його монету, але не розголошує цю адресу решті учасників.

Далі кожен учасник генерує нову пару ключів для спрямованого шифрування, після чого учасники групи обмінюються відкритими для шифрування ключами між собою, причому новий відкритий ключ підписується особистим ключем, який відповідає адресі з невитраченою монетою. Таким же чином підписуватимуться всі повідомлення учасників при подальшій взаємодії. Це є першим етапом.

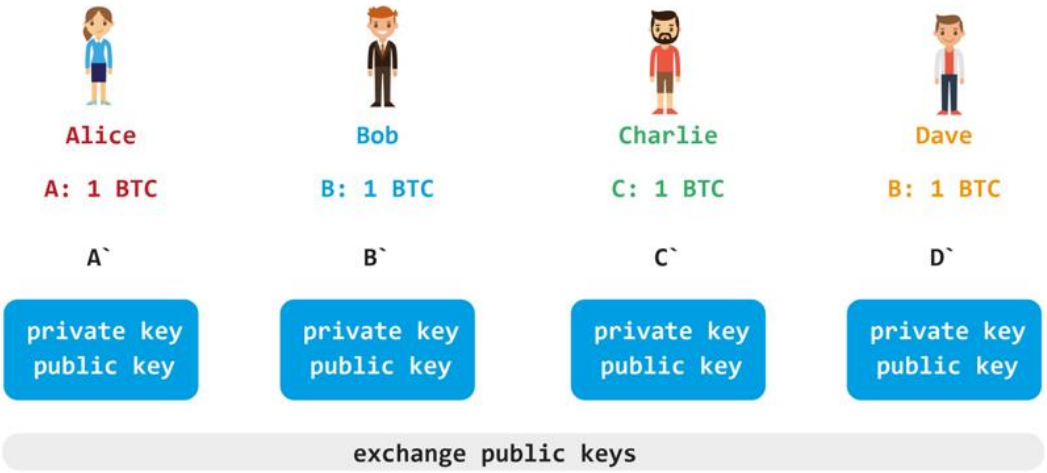

Рисунок 2.4 – Перший етап роботи методу CoinShuffle

Учасники перемішуються та утворюють чергу. Тепер Аліса бере адресу A' і шифрує, використовуючи відкритий ключ Дейва. Шифротекст, який отримала Аліса, знову шифрується з використанням відкритого ключа Чарлі. Далі отриманий шифротекст знову шифрується, але вже з використанням відкритого ключа Боба.

Результат шифрування Аліса передає Бобу. Боб розшифрує своїм особистим ключем отримане повідомлення. Потім Боб бере адресу B' і шифрує з використанням відкритого ключа Дейва, потім з використанням відкритого ключа Чарлі і додає у список. Цей список він перемішує випадково і передає Чарлі. Чарлі розшифровує елементи списку своїм особистим ключем, додає адресу C', зашифровану з використанням відкритого ключа Дейва, у список і перемішує всі елементи списку випадковим чином. Список передається Дейву, який його дешифрує, отримує відкриті дані адрес для відправлення монет, додає адресу D', перемішує випадковим чином і на підставі цих адрес, відомих входів транзакції та сум формує загальну транзакцію (рис. 2.5).

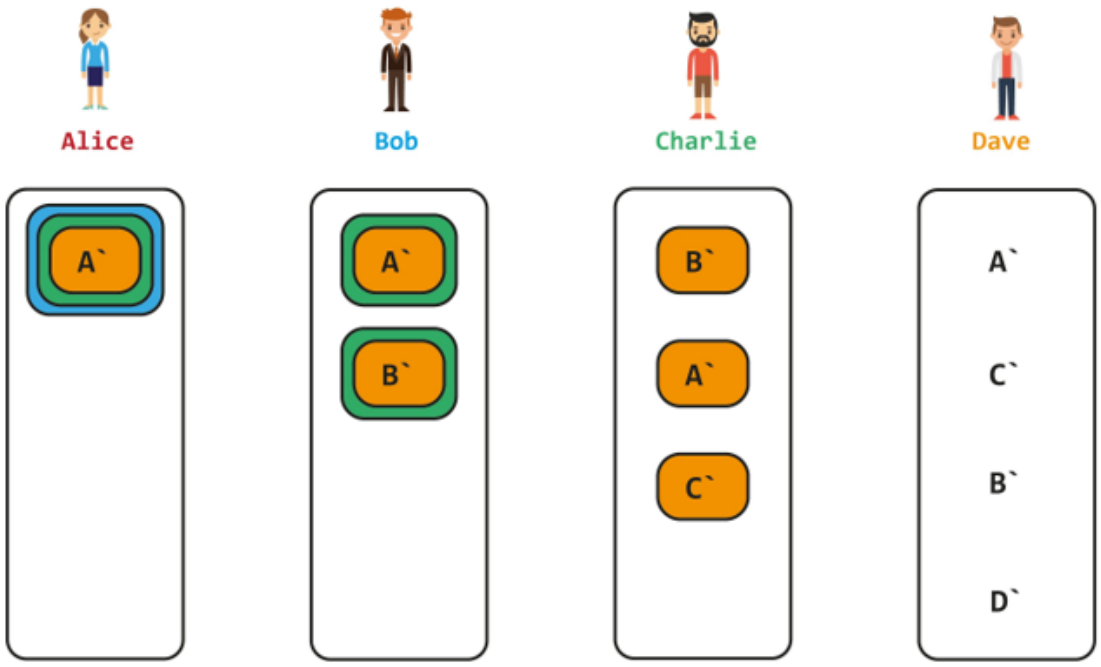

Рисунок 2.5 – Другий етап роботи методу CoinShuffle

Дейв поширює заготівку транзакції решті учасників групи. Далі кожен перевіряє, чи є у виходах транзакції потрібна йому адреса і чи збігається сума. Якщо умова виконана, учасник підписує транзакцію, підтверджуючи володіння монетами свого входу (рис. 2.6). Учасники обмінюються підписами і якщо транзакція набирає всі необхідні підписи, то може бути поширена через мережу для підтвердження.

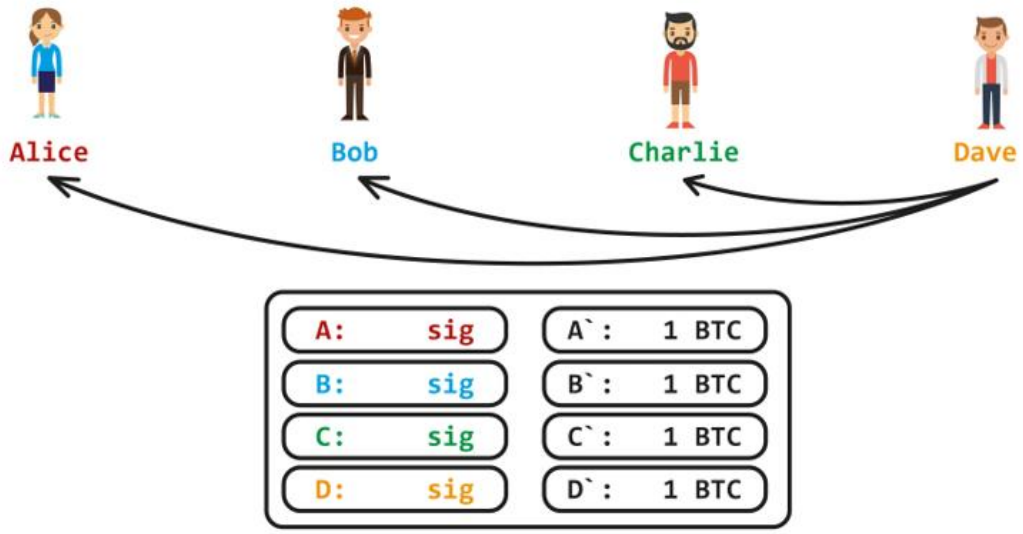

Рисунок 2.6 – Третій етап роботи методу CoinShuffle

Якщо хтось із учасників починає відхилятися від основного сценарію взаємодії, то інші можуть спільно проаналізувати історію взаємодії та вивести порушників із групи, щоб повторити усі етапи без них. Це важлива особливість даної модифікації.

2.2 Метод Blind Signature

Розглянемо один із методів, який дозволяє кільком сторонам взаємодіяти та забезпечує при цьому необхідний рівень конфіденційності.

Сліпий підпис (blind signature) – різновид електронно-цифрового підпису, особливість якого полягає в тому, що сторона, яка підписує, не може точно знати, що міститься в документі, який підписується [10].

Основна ідея сліпих підписів полягає в наступному:

1. Відправник А шифрує документ і надсилає його стороні В.

2. Сторона В, не має можливості ознайомитись зі вмістом документу, підписує його і повертає назад стороні А.

3. Сторона А знімає свій шифр, залишаючи на документі тільки підпис сторони В.

По завершенні виконання цього протоколу сторона В нічого не знає ні про повідомлення, ні про підписи під цим повідомленням.

Мета «сліпого» підпису полягає в тому, щоб перешкодити підписувачу В ознайомитися з повідомленням сторони А, яку він підписує, і з відповідним підписом під цим повідомленням. Тому надалі підписане повідомлення неможливо пов'язати зі стороною А.

Схема безпечного сліпого підпису повинна задовольняти 3 властивостям, а саме:

– нульове розголошування. Ця властивість допомагає користувачеві отримати підпис на даному повідомленні, не розкриваючи самого повідомлення підписуючій стороні;

– невідстежуваність. Підписуюча сторона не може відстежити пару підпис-повідомлення після того, як користувач оприлюднив підпис на повідомленні;

– непідкладність. Тільки підписуюча сторона може згенерувати дійсний підпис. Ця властивість найважливіша і повинна задовольнятися для всіх схем підписів;

Завдяки властивостям нульового розголошення і невідстежуваності, схема сліпого підпису може бути широко задіяна в додатках, де необхідна конфіденційність, наприклад, в системах електронного голосування або децентралізованих системах.

Наприклад, Аліса хоче, щоб Боб наосліп підписав повідомлення m. Для цього необхідно виконати наступні дії (рис. 2.7):

1. Аліса зашифровує повідомлення m функцією f, отримуючи зашифроване повідомлення $c = f(m)$.

2. Аліса відсилає зашифроване повідомлення Бобу.

3. Боб наосліп (так як не знає, що знаходиться всередині) підписує повідомлення c за допомогою функції g, отримуючи $c' = g(c) = g(f(m))$.

4. Боб посилає c' назад Алісі.

5. Аліса отримує c' та прибирає своє шифрування (розшифровує), отримуючи: $c'' = g(f(m))*f^{-1} = g(m)$.

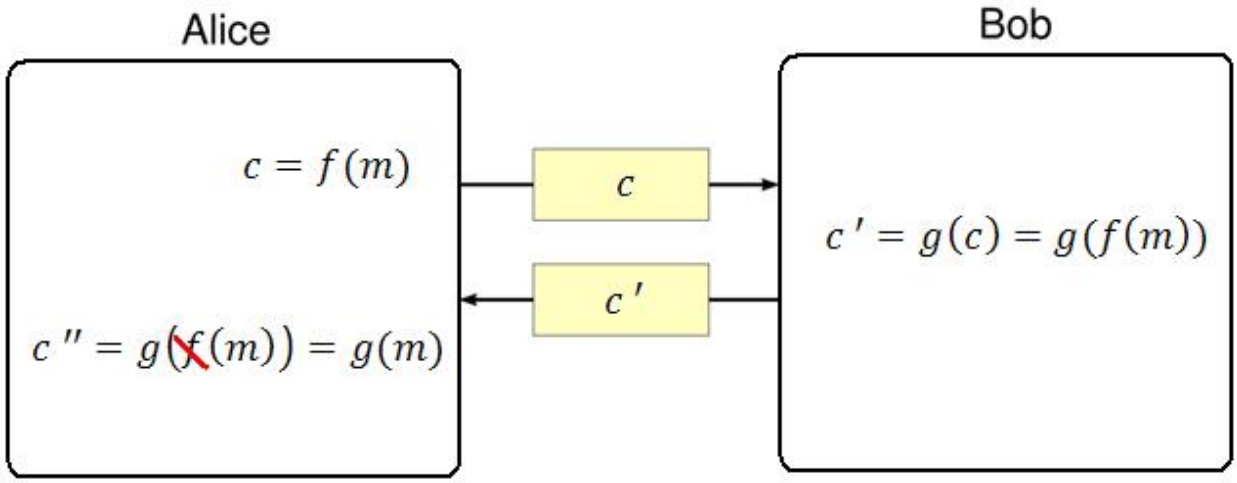

Рисунок 2.7 – Схема роботи методу Blind Signature

Найширше застосування протоколи сліпого підпису знайшли у сфері цифрових валют та таємного голосування.

Перша реалізація алгоритму «сліпого» підпису була розроблена з застосуванням криптосистеми RSA.

Припустимо, що спочатку у Боба є відкритий ключ (p,e), де p — це модуль, а e — публічна експонента ключа.

1. Аліса вибирає випадковий маскуючий множник r , який э взаємно простим з p, і обчислює $m' \equiv mr^e \ mod \ p$.

2. Аліса відсилає m' по відкритому каналу Бобу.

3. Боб обчислює, $s' \equiv (m')^d \ mod \ p$ використовуючи свій закритий ключ (p,d).

4. Боб відсилає s' назад Алісі.

5. Аліса прибирає своє початкове маскування і отримує підписане Бобом вихідне повідомлення m наступним чином: $s \equiv s' * r^{-1} \ mod\ p \equiv m^d \ mod\ p$.

Відповідно до джерела [10] уразливістю «сліпого» підпису є те, що алгоритм RSA може стати об'єктом атаки, завдяки якій стає можливим розшифрувати раніше підписане наосліп повідомлення, видавши його за повідомлення, яке тільки ще треба підписати. Виходячи з того, що процес підпису еквівалентний розшифровці стороною, яка підписує (з використанням секретного ключа), атакуючий може замінити для підпису вже підписану наосліп версію повідомлення m, зашифрованого за допомогою відкритого ключа сторони, яка підписує.

2.3 Метод Zero-Knowledge Proof

Zero-knowledge proof (ZKP) [11] є методом, який дозволяє одній стороні довести володіння знанням про деякий секрет без розголошення будь-яких даних про цей секрет.

Для подібних схем доказів можуть використовуватись різні математичні підходи, які обов'язково повинні забезпечити докази такими властивостями.

Властивості методу zero-knowledge proofs:

– completeness (повнота);

– soundness (коректність);

– zero knowledge (нульове знання).

Властивість completeness передбачає, що якщо перевіряючий і той користувач, хто доводить, чесно дотримуються протоколу, то той, хто доводить, гарантовано може переконати того, хто перевіряє, в коректності затвердження (крім дуже малої частки ймовірності).

Властивість soundness означає, що якщо твердження є невірним, то той, хто доводить, не зможе переконати того, хто перевіряє у своїй правоті, за винятком дуже малої ймовірності.

Властивість zero knowledge передбачає, що під час доказу перевіряючий не може дізнатися нічого про затвердження крім факту, що воно є вірним/невірним.

Аналіз функціонування методу ZKP свідчить про те, що існують два варіанти реалізації цього методу: інтерактивний та неінтерактивний.

Інтерактивний варіант передбачає, що той користувач, хто доводить і той користувач, хто перевіряє, спілкуються безпосередньо між собою. Під час спілкування перевіряючий дедалі більше переконується у цьому, що твердження доведено правильно. Особливістю даного підходу є факт того, що жодну третю сторону не можна переконати в тому, що твердження є вірним, використовуючи ті ж самі докази.

У разі неінтерактивного протоколу той користувач, хто доводить, за заздалегідь визначеною схемою, формує окремий набір даних. Цей набір є доказом твердження, яке не має конкретного адресату. Маючи такий доказ, будь-хто може будь-якої миті часу переконатися у тому, що твердження доведено правильно.

Прикладом неінтерактивного протоколу є zk-SNARKs – протокол, який дозволяє довести правильність обчислення інформації без будь-якої взаємодії між обома сторонами (користувача, який доводить і користувача, який перевіряє). Цей протокол складається з великої кількості складових частин:

– гомоморфне шифрування;

– polynomial blind evaluation;

– algebraic circuit;

– rank-1 constraint system (R1CS);

– quadratic Arithmetic Programs (QAP).

2.4 Метод Confidential Transactions

Особливість методу Confidential Transactions (CTs) [12] полягає у тому, що даний метод допомагає повністю приховати фактичні суми на входах та

виходах транзакції від третіх осіб. Кожен користувач може перевірити, що сума всіх виходів не перевищує суму всіх входів, чого вже достатньо для підтвердження цієї транзакції. Це стало можливим завдяки використанню вищеописаного підходу, zero-knowledge proof.

Властивості методу Confidential Transactions:

– суми платежів приховані для всіх, крім відправника та одержувача;

– транзакція містить дані, достатні для підтвердження;

– використовується zero-knowledge proof.

Для доказу того, що сума виходів не перевищує суми входів, використовується підхід Pedersen Commitment, який базується на перетвореннях групи точок на еліптичній кривій. З метою боротьби з неконтрольованою емісією монет у цій схемі обов'язково застосовується доказ використання допустимих сум на виході транзакції. Щоб перевірити, чи були використані невід'ємні суми, які перевищують порядок базової точки, застосовуються підхід Range Proofs.

Недоліком цього підходу є те, що процес доведення використання невід’ємних сум у транзакції є дуже витратним з точки зору обчислювальних ресурсів.

2.4.1 Метод Ring Confidential Transactions

Наступна методика називається Ring Confidential Transactions. Для заплутування історії походження монет тут використовують кільцеві підписи. Відправник у вході своєї транзакції посилається не на один конкретний вихід (UTXO), а відразу на кілька. Далі, за допомогою кільцевого підпису він доводить, що йому належать монети одного з кількох виходів, але не розголошується з якого конкретно. З цього випливає, що неможливо однозначно відстежити історію походження монет.

Застосування кільцевих підписів таким чином було вперше запропоновано в протоколі CryptoNote [13], на базі якого працюють декілька криптовалют. Ring Confidential Transactions побудовані на основі CTs. Вони

дозволяють створювати транзакції з безліччю входів і виходів, де неможливо однозначно відстежити походження кожного входу, суми платежів приховані, а взаємодія з іншими користувачами для виконання транзакції не потрібна (рис. 2.8).

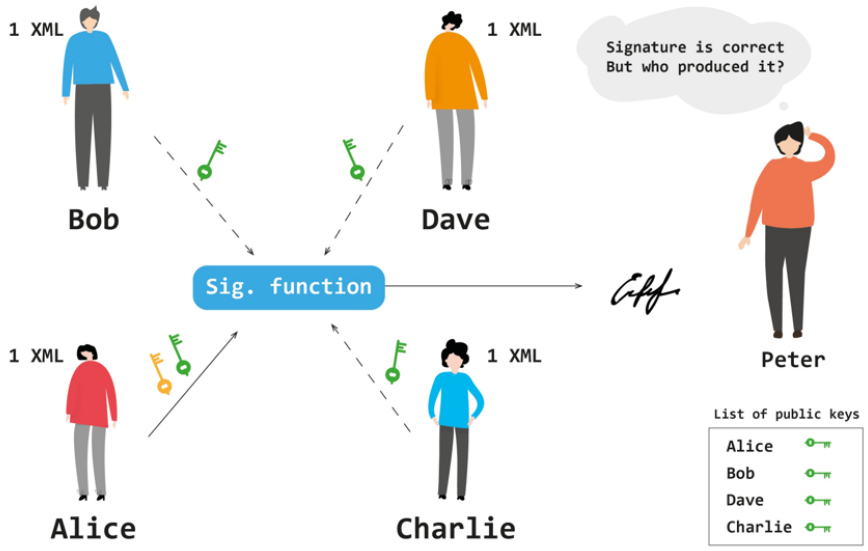

Рисунок 2.8 – Загальна схема роботи методу Ring Confidential Transactions

2.5 Стандарти CryptoNote

Основною відмінністю сучасних децентралізованих систем є використання криптографічних операцій, зокрема цифрових підписів, для доказу володіння монетами та підтвердження всіх змін бази даних децентралізованої системи. Проте дуже часто використання стандартних криптографічних алгоритмів обмежує конфіденційність учасників мережі (простота розплутування ланцюжків транзакцій). У цьому випадку доводиться вдаватися до методів, таких як confidential transactions, ring signatures, stealth addresses тощо [13].

У 2013 році з’явилася серія стандартів CryptoNote, на основі яких побудовано анонімні криптовалюти, такі як Monero, Bytecoin, Karbo та багато інших. У цьому підрозділі ми розглянемо ключові концепції цих стандартів, які вони використовують, а також розберемося, за рахунок чого досягається

анонімність і непростежуваність у протоколах перерахованих криптовалют.

Серія стандартів CryptoNote складається з десяти стандартів, які описують основні концепції побудови анонімної криптовалюти. У стандартах описані технології побудови загальної історії транзакцій, структура блоку та транзакції, використання ключів для формування адрес та підписання транзакцій (використання механізму кільцевого підпису) [14].

Розглянемо технологічні особливості стандартів CryptoNote, а також особливості їх застосування на прикладі криптовалюти Monero.

2.5.1 Підписи у стандарті CryptoNote

CryptoNote описує алгоритм одноразового кільцевого підпису. Такий підхід дозволяє користувачу підписати транзакцію і зберегти при цьому анонімність, оскільки верифікатор може переконатися, що підпис був вироблений одним із членів групи, не маючи можливості дізнатися, ким саме. Для запобігання атаці подвійного витрачання було вирішено використовувати механізм одноразового підпису.

Що таке одноразовий кільцевий підпис і яким чином він обчислюється? Так само як і для обчислення значення звичайного підпису, користувачеві необхідно мати ключову пару – особистий та відкритий ключі. Однак у випадку з кільцевим підписом також повинні використовуватися відкриті ключі решти учасників групи.

У разі використання звичайного підпису верифікатор однозначно може пов'язати значення підпису з відповідним відкритим ключем (ідентифікатором користувача), оскільки публічний ключ використовується для перевірки підпису (рис. 2.9).

Для обчислення кільцевого підпису, крім особистого ключа користувача, необхідно використовувати його відкритий ключ, а також відкриті ключі всіх учасників групи.

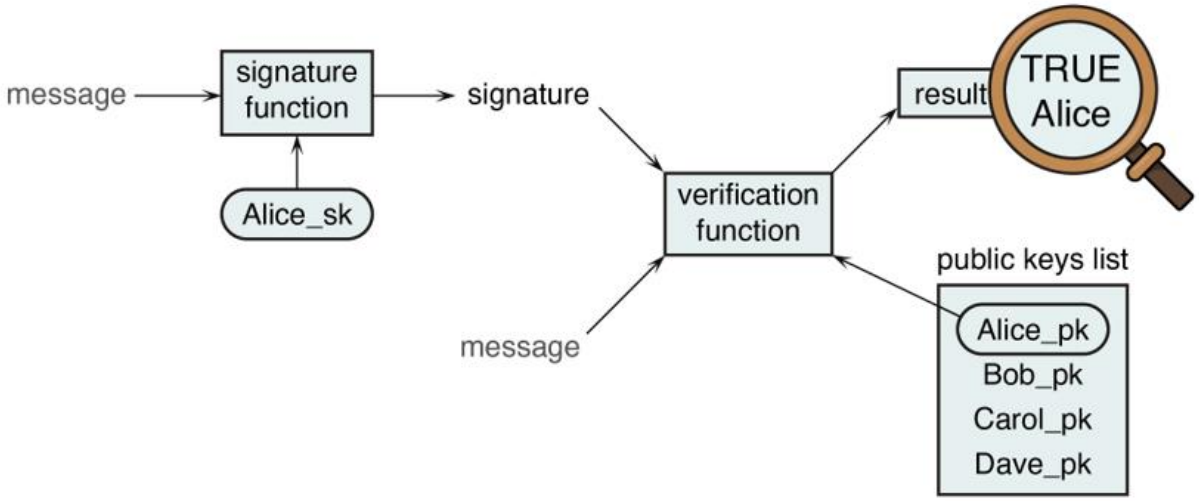

Рисунок 2.9 – Схема функціонування класичного цифрового підпису

Кільцевий підпис дозволяє верифікатору переконатися, що підпис сформував один із учасників групи, але, оскільки для верифікації необхідні всі відкриті ключі, сказати точно, хто був підписувачем, неможливо (рис. 2.10).

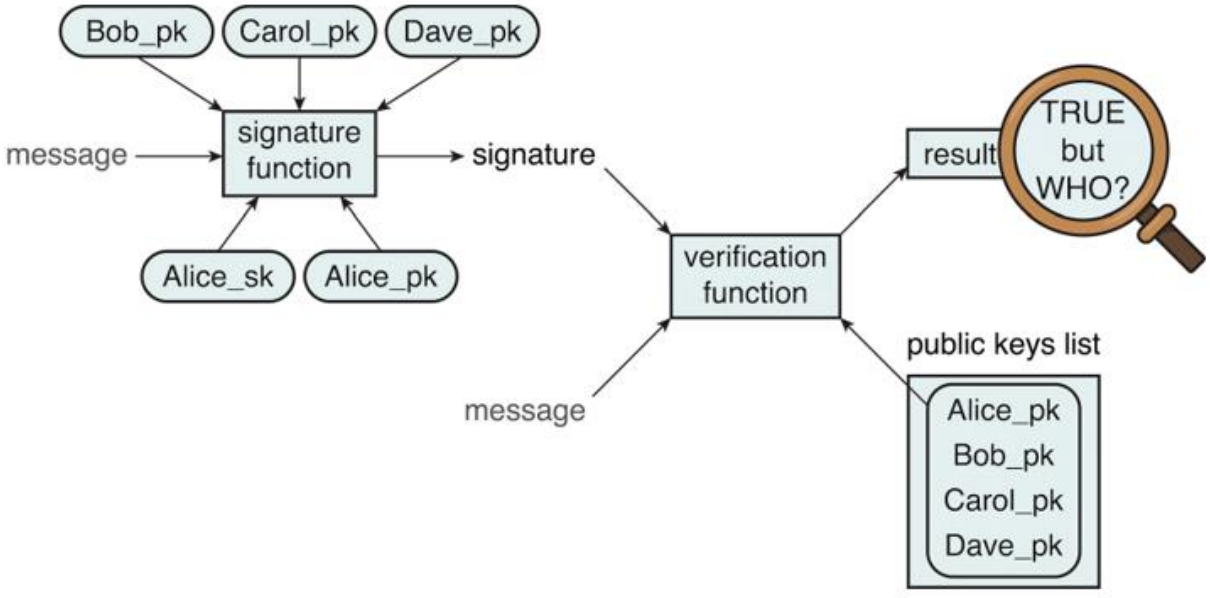

Рисунок 2.10 – Схема функціонування кільцевого підпису

Таким чином, використання одноразового кільцевого підпису дозволяє верифікувати дані але не дозволяє ідентифікувати, який користувач ці дані підписав.

2.5.2 Зв'язок ключів та адрес

На відміну від більшості криптовалют, де користувач має одну ключову пару для керування однією адресою/акаунтом, стандарти CryptoNote припускають, що кожен користувач має дві ключові пари. Фактично, користувач генерує два великі секретні значення, які є його особистими ключами. Відкриті ключі так само, як і в інших децентралізованих системах є множенням цих секретів на значення базової точки. Відкриті ключі мають такі назви: public spend key та public view key.

Адреса формується шляхом конкатенації префіксу (18 для основної мережі), двох відкритих ключів, а також checksum від попередніх полів (рис. 2.11). Checksum є хеш-значення KECCAK, усічене до 4 байт. Отриманий результат кодується Base58 [14].

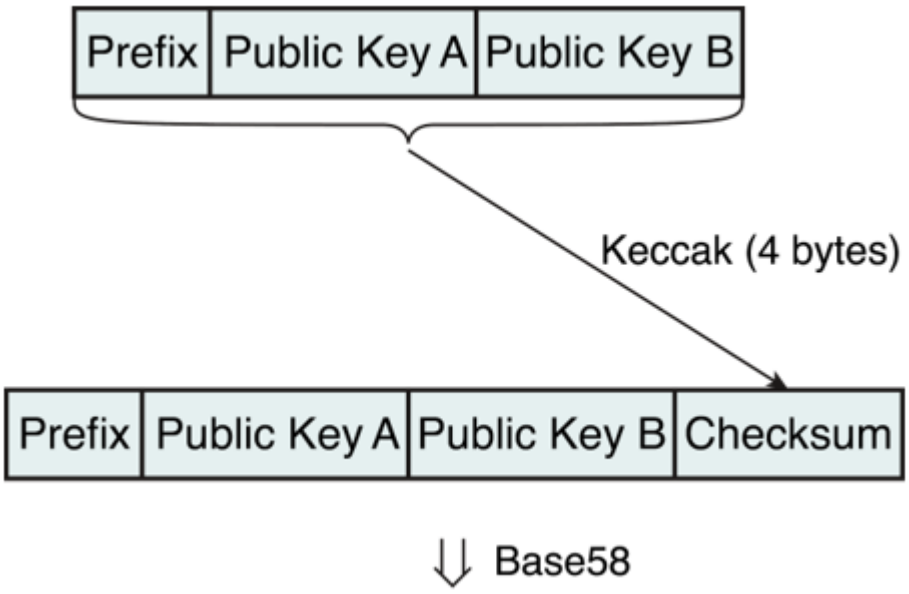

Рисунок 2.11 – Схема формування адреси

2.5.3 Stealth addresses

У більшості криптовалют користувачі мають відкриті адреси/акаунти. Будь-яка третя сторона може пов'язати всі транзакції, які мають відношення до цієї адреси, а відповідно, і до її власника, якщо зв'язок хоча б однієї транзакції та сторони, яка бере участь у ній, буде розкрито. У таких умовах

рекомендується створювати нову адресу для кожного нового вхідного платежу та здачі. Однак такий підхід є не зовсім зручним, тому що він передбачає часту генерацію ключів і позбавляє користувача зручності володіння однією адресою/акаунтом.

Стандарти CryptoNote передбачають використання так званих stealth addresses [15]. Доступ до них має лише власник особистих ключів, причому будь-якій третій стороні складно розкрити зв'язок між звичайною адресою та stealth address. Давайте розглянемо, яким чином відбувається формування таких адрес та як користувач отримує доступ до монет, які на такій адресі заблоковані (рис. 2.12).

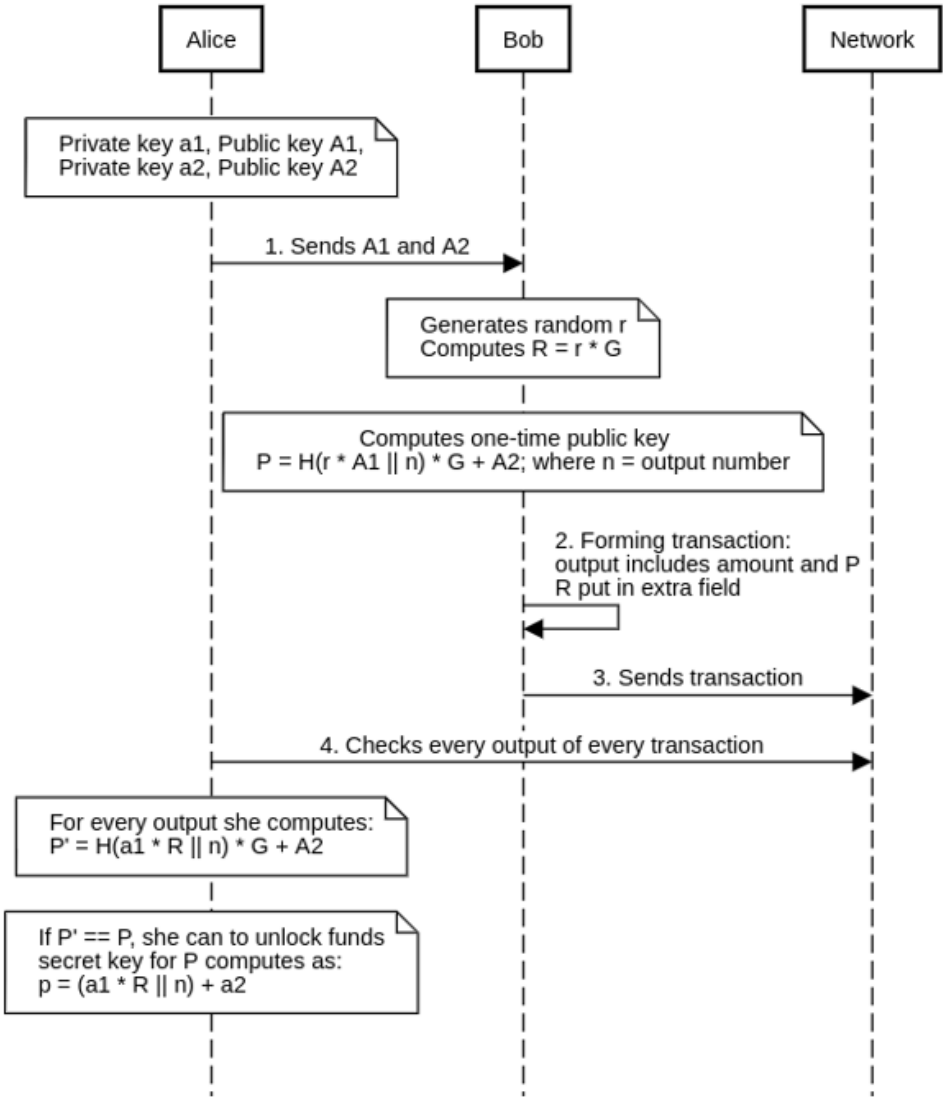

Рисунок 2.12 – Схема надсилання платежу на stealth address

Попередньо Аліса генерує для себе дві ключові пари та формує адресу. Для отримання платежу на першому етапі Аліса надсилає свою адресу Бобу. Боб отримує адресу і виділяє значення двох відкритих ключів. Після цього Боб генерує одноразове секретне значення *r* і множить його на базову точку для отримання *R*. Далі Боб використовує отримані значення формування одноразового відкритого ключа, до якого будуть прив'язані монети; тільки власник відповідного секретного ключа зможе отримати доступ до них.

У цьому випадку одноразовий відкритий ключ *P* обчислюється так:

$$P = H(r * A1 \,||\, n) * G + A2, \tag{2.1}$$

де, *H* – криптографічна хеш-функція, *r* – одноразове секретне значення, *A1, A2* – публічні ключ, *G* – базова точка у групі еліптичних кривих, *n* – індекс виходу.

Другим кроком є формування та відправлення Бобом транзакції до мережі. Для цього він формує транзакцію, подає на вхід невитрачені виходи, додає у вихід суму переказу та значення одноразового відкритого ключа, а в полі extra вставляє значення R. Після цього на третьому кроці Боб відправляє транзакцію для підтвердження її валідаторами.

Оскільки монети відправлені на одноразовий публічний ключ, а не безпосередньо на адресу Аліси, вона сама спочатку не знає, які виходи їй належать. Зважаючи на це (четвертий крок), вона перевіряє всі виходи всіх підтверджених транзакцій у пошуках того, який адресований їй. Для кожного з виходів вона обчислює таке значення:

$$P' = H(a1 * R \,||\, n) * G + A2, \tag{2.2}$$

де, *H* – криптографічна хеш-функція, *a1* – секретний ключ, *R* – секретне значення, *n* – індекс виходу, *G* – базова точка у групі еліптичних кривих, *A2* – публічний ключ.

Якщо це значення дорівнює вказаному у виході транзакції *P*, вихід дійсно належить Алісі. Для розблокування транзакції їй необхідно обчислити відповідний секретний ключ:

$$p = H(a1 * R \,||\, n) + A2, \tag{2.3}$$

де, *H* – криптографічна хеш-функція, *a1* – секретний ключ, *R* – секретне значення, *n* – індекс виходу, *A2* – публічний ключ.

Та підписати наступну транзакцію, яка витрачає цей вихід.

2.5.4 Механізм захисту від подвійної витрати

Використання кільцевих підписів дозволяє користувачеві виконати атаку подвійної витрати. Давайте розглянемо приклад, у якому у мережі є чотири користувача, кожен із яких має один невитрачений вихід (рис. 6.5).

Рисунок 2.13 – Невитрачені виходи учасників кільця

Кожен вихід закріплений за конкретним відкритим ключем. Для створення кільцевого підпису користувач збирає відкриті ключі з інших невитрачених виходів, після чого формує кільцевий підпис. Наприклад, Аліса хоче відправити собі монети на іншу адресу та формує транзакцію, яку підписує, поєднуючи відкриті ключі Аліси, Боба та Керол (рис. 2.14).

Рисунок 2.14 – Формування транзакції Аліси

Фактично, валідатор може перевірити, чи закріплений за кожним ключем невитрачений вихід та чи свідчить підпис про те, що цей вихід витрачається. Однак при цьому валідатор не знає, який саме із цих виходів витрачається. Тому нічого не заважає Алісі створити альтернативну транзакцію, яка витрачає ті ж монети на іншу адресу, після чого підписати її, використовуючи відкриті ключі Боба та Дейва (рис. 2.15).

Рисунок 2.15 – Спроба Аліси виконати атаку подвійної витрати

Якщо не введено додатковий механізм захисту, то для валідатора це значення підпису також буде коректним, а отже, кошти буде переведено двічі. Тому стандарт CryptoNote передбачає використання зображення особистого ключа як засіб захисту від атаки подвійної витрати. Фактично, цей механізм дозволяє виявити, чи були кілька підписів обчислені за допомогою того самого особистого ключа, без розкриття самого ключа. Зображенням секретного ключа є його хеш значення, і воно використовується для перевірки підпису, що не дозволяє підмінити його.

Якщо Аліса намагається виконати те саме, валідатор побачить, що підписи були обчислені за допомогою одного і того ж особистого ключа і конфліктуюча транзакція буде відхилена.

2.5.5 Структура блоків у CryptoNote

Стандарт CryptoNote включає в себе організацію історії транзакцій у ланцюжок блоків. Кожен блок пов'язаний із попереднім за допомогою

унікального ідентифікатору – хеш-значення блоку. На рисунку 2.16 можна побачити структуру блоку CryptoNote.

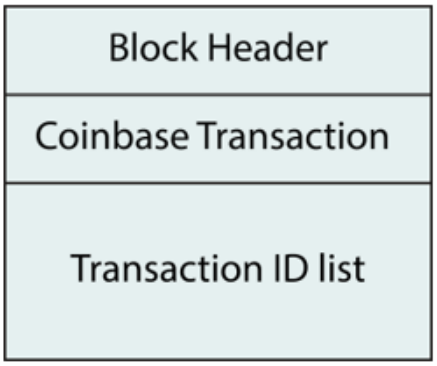

Рисунок 2.16 – Структура блоку згідно з CryptoNote

Блок складається з трьох компонентів: заголовка блоку (block header), тіла Coinbase транзакції та списку ідентифікаторів включених транзакцій. Сам блок не містить жодних даних транзакцій, крім їх ідентифікаторів. Це зроблено для досягнення невеликого розміру блоку, проте повним вузлам все ще потрібно зберігати всі транзакції.

Ключовою відмінністю блоків Monero і подібних криптовалют є їх динамічний розмір [16]. Середній розмір блоку Monero розраховується як середнє значення від розмірів попередніх 100 блоків. Максимальний розмір блоку не може перевищувати середній більш ніж 2 рази. Також зазначимо, що у протоколі передбачено механізм штрафування майнерів за видобуток блоків, розмір яких перевищує середнє значення.

Розмір штрафу розраховується так:

$$P = R * ((B / M) - 1) * 2, \quad (2.4)$$

де, P – розмір штрафу; R – базова винагорода за видобуток блоку; B – розмір видобутого блоку; M – середнє значення розмірів останніх 100 блоків.

Таким чином, якщо сформований блок має максимально допустимий розмір, то винагорода за його створення дорівнює штрафу. Також варто зазначити, що сума штрафу сплачується валідатором наступного сформованого блоку.

Давайте детальніше розглянемо заголовок блоку та його поля (табл. 2.1).

Таблиця 2.1 – Заголовок блоку та всі його поля

Поле	Опис
MajorVersion	Основна версія заголовка
MinorVersion	Другорядна версія заголовка
Timestamp	Мітка часу створення блоку
PrevID	Хеш попереднього блоку
Nonce	Необхідність створення блоку

У полях MinorVersion та MajorVersion вказуються версії протоколу: відповідно перша та версія яку підтримує вузол-валідатор. У полі Timestamp вказується час створення блоку (у форматі UNIX timestamp). Поле PrevID містить хеш-значення (ідентифікатор) попереднього блоку в ланцюжку, а поле Nonce – значення, необхідне для доказу вирішення ресурсномісткої задачі.

Крім заголовку та списку ідентифікаторів транзакцій, блок містить тіло Coinbase транзакції, яка сплачує винагороду валідатору. Coinbase транзакція також повинна бути перевірена рештою учасників.

Розглянемо, як формується ідентифікатор блоку відповідно до стандарту CryptoNote. Для обчислення хеш-значення блоку використовується хеш-функція CryptoNight. Особливість цієї хеш-функції – вимогливість до обсягів необхідної пам'яті (близько 2 МБ), що робить використання спеціалізованих інтегральних схем неефективним. Процедуру отримання ідентифікатору блоку показано рисунку 2.17.

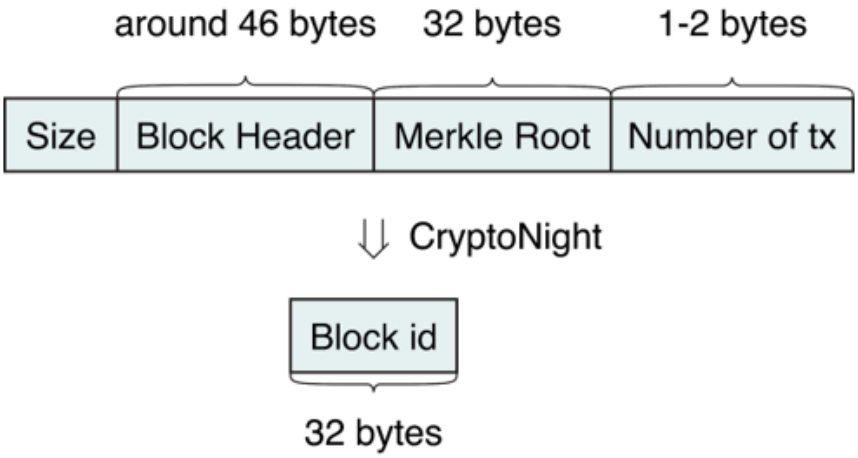

Рисунок 2.17 – Схема формування ідентифікатора блоку

Зазначимо, що для отримання Merkle Root [17] використовується функція KECCAK-256. Як приклад давайте розглянемо один із підтверджених блоків у мережі Monero (рис. 2.18).


```
{
"status":"OK",
"block_data":{
        "id":"0",
        "jsonrpc":"2.0",
        "result":{
                "block_header":{
                        "block_size":17515,
                        "block_weight":17515,
                        "cumulative_difficulty":28938438630754569,
                        "depth":0,
                        "difficulty":34957140781,
                        "hash":"9b54e41c73355e17f6ff54a4fdfc1586c025375e80573f749226bae6c30903e8",
                        "height":1818136,
                        "long_term_weight":17515,
                        "major_version":11,
                        "minor_version":11,
                        "nonce":92274898,
                        "num_txes":7,
                        "orphan_status":false,
                        "pow_hash":"",
                        "prev_hash":"ca1f6ca92686ed39ac450d13be6af8c787142bb37724822875a5bbb97d77dc06",
                        "reward":2887451463258,
                        "timestamp":1555881700},
                        "status":"OK",
                        "untrusted":false
                }
                "miner_tx_hash":"babe94e2cbed26b05c9e30de566f4431856a315c38217a096066be688b8d0698",
                "status":"OK",
                "untrusted":false,
                "tx_hashes": [
                        "ce27b70423f384c964f6d2a2f4364f2f7cca034358071bc6869a0ab7c873b4b1",
                        "aba3df615be680e6ff150e9470ac17c0d05532f44f9afb6f4518decf884af7f8",
                        "e2f29e24daef4ff116d09edef0eab6387710c6313ff8ec7f37029f1538630c06",
                        "d0280ec61e9156cfdb16027784e8945705c4476b79a2df49d63c0fcbcc1802fd",
                        "deaac4cfd259ae8c8d79ce0c1d15bceacf6cdca304e32ed4fd365c5547610730",
                        "96fe084b2823074495b42c50d8e13c4f1dd94e821b92b14545735a3aa52e0e79",
                        "eddacb37616d0aabc219f296e8c6d68edcaa518eaa8b906c39d460e66b26530d"
        }
}
```


Рисунок 2.18 – Приклад блоку, сформованого за протоколом Monero

2.5.6 Структура транзакцій у CryptoNote

Транзакції в анонімних валютах, побудованих за цим протоколом, схожі структурою з транзакціями в Bitcoin і будуються за моделлю UTXO (кожна транзакція містить входи – посилання на транзакції, з яких монети були отримані – і виходи, в яких містяться умови витрачання монет).

Кожна транзакція складається із двох компонентів: префіксу транзакції та набору підписів. Префікс транзакції містить основні дані транзакції (хто, кому і скільки перекладає) і має таку структуру (табл. 2.2).

Таблиця 2.2 – Поля транзакції та їх призначення

Поле	Призначення
Version	Версія структури транзакції
UnlockTime	UNIX timestamp; мітка часу, по досягненню якого транзакція може бути включена до блоку
InputNum	Кількість входів транзакції
Inputs	Масив входів транзакції
OutputNum	Кількість виходів транзакції
Outputs	Масив виходів транзакції
ExtraSize	Кількість додаткових даних у байтах
Extra	Додаткові довільні

У заголовку транзакції є 2 поля: version та unlock_time. Перше поле відображає номер версії транзакції. І тут поле дорівнює 2, отже, транзакція підтримує механізм ring confidential transaction. У першій версії протоколу була реалізація лише одноразового кільцевого підпису для приховання історії походження монет. Використання ring confidential transaction дозволило додатково приховувати суми переказів. У полі unlock_time вказується значення часу, після якого транзакція може бути підтверджена.

Далі йдуть входи транзакції. Вхід транзакції містить поля amount, key_offsets та k_image. У полі amount міститься значення кількості монет на

вході. У нашому випадку значення amount дорівнює 0, що означає, що сума входу прихована. Поле key_offsets містить індекси зміщення ключів (причому кожний наступний індекс означає зміщення щодо попереднього). Фактично це індекси на невитрачені виходи в UTXO database (рис. 2.19).


```
{
"status":"OK",
"transaction_data":{
    "version":2,
    "unlock_time":0,
    "vin":[{
        "key":{
            "amount":0,
            "key_offsets":[8546498,867206,16243,62207,3704,2688,1084,2763,13358,1400,90],
            "k_image":"b91767da2ad5fbbd3e209c6d4048f6d1e938f52c357f99065d4ae378c29b5640"}}],
    "vout":[{
        "amount":0, "target":{ "key":"925be433334fea5039851d13a0f8018c38010b119b4eed7333050dac969d28c6"}},
    {
        "amount":0, "target":{ "key":"5f6f7faa04a1a9519005b3c420896fbc3b059773b541b824bf661b7b56b45c33"}}],
    "extra":[2,33,0,249,83,246,177,236,240,68,147,190,88,37,15,230,28,84,64,245,
        163,231,35,114,116,190,130,203,182,23,168,44,119,107,20,1,30,193,94,
        132,92,85,9,27,15,190,43,243,22,199,64,165,168,143,1, 36,53,193,231, 242,230,125,185,18,214,73,222,115,248],
    "rct_signatures":{
        "type":4, "txnFee":34500000,
        "ecdhInfo":[{"amount":"8e4b7df755163f51"}, {"amount":"5556ecf232344ef8"}],
        "outPk":["153f9272da055eee5bd9b4527d2109c52ed8f5c5792aad07e67d570c20ed277b",
            "217745c1d23379f5ee972403f197d3d908302064706f72f4ef64acb8b28511c9"]},
    "rctsig_prunable":{
        "nbp":1, "bp":[{
            "A":"3087896cdf7d2fd7a5a02f2c87728df4220a51eef2faf081758f598e4d326248",
            "S":"a47b78da44b6dca11cd5af768c9074fdd289a165e06d28cd67a78ee32de10c86",
            "T1":"666215ee9caf09d3b47d529aa591a6c14481891b78992cb24e5bd3591304b946",
            "T2":"b14a6194c3ed503c676f148934ef6f90f4caf0d5954d37e0a4d697f45518d655",
            "taux":"e65f14343a257487db5a56151ed37ecf47086d7738f49234fa4cc98be870180e",
            "mu":"8fee646a0d958fd2d223c33b874eca5951a93dba68bd4b2e33e82d1883b5d40f",
            "L":[
                "4021aa49e3c4cd201f4e665f7417c32b39cda29B9465f0741cc99b0f55e01232",
                "66f1b9de0ad1f425d80d401deec8b0f4776c591bab0d1ee1ba80ad920c625237",
                "0ea388b7fb427b763eff04d158f43c95e3e45e336a8505ae89d8c33b08bc28eb",
                "b357d0bb8deca286005d38cc5e00c36dbfed5c8376d5c1bb6b999595c897dfd4",
                "69c764ac248f8201e171e411f66edc6cb377b750d7c17e68724807d13908e458",
                "e48048f56d818ac2ad17f414713b69c5d5e93c5ed05825d6fcedad5a7509f711",
                "2e1e6f09cf56a9655ea3ce5b1a896b717b4c1fcfdf21e8ffb523b69238cb471a"],
            "R":[
                "5376c42030d8299b3eb9a33c3d4a662e4a5fe56497faaa046f9929c9d649470b",
                "b7a6078365b6baf7a02b233cb00eb9ea55f6c3685f84e57dfe02127178e450d0",
                "618879aa6a4c6baf21e163e30920c2717899a8a5d0f4a129cb1bef03bff3500e",
                "5e7ced843e85f51fb70bfd141e7f17421b07f455b94ea013183eb9e273272d5d",
                "da94b43e652263f941ca6c6440a7d18c4784f1c12a4e12d65470d1010eff65fd",
                "1e4d274b304bb5b970dfd29ae07001c7a886334f0a81d7c48d5409ac82acf9af",
                "705eed13e0371e3ccdf83fe75186e5168af55f7443784bece8f24874b87005b"],
            "a":"4e39f5ca50d3c50fa209fadda2f09bf40a0840e40929721a84106017e4f3b807",
            "b":"9163b8714ea5f2912ccfc8cfce9adbe1f5a142e5821c5f5cbc460a5434e7930d",
            "t":"38e1d8b1f1c64fb217fc9b89e57e81afabdcfd7313729c9ff1e4a140f7cff406"}],
        "MGs":[{
            "ss":[
                ["a7152489a591fdba18d47a7dad98829356dd60876aaee61a4ea4d6c221270d603",
                "7e3e763cc1ccfa25a6542dd31842eac80a8c0b9f6ddde15f63ea1724e52e470a"],
                                    ["9b223704cd8b7d43379405cb4dfc5e7fabbf45206ed769d3ecdbeec0753f8501",
                "322d867881e22b6506a3fbec0b42296b155a3c4c8e530a70a1655263ad81a80c"],
                ["ee5eab54f22a25b9e89a6e31c37b0d4154ae9b15a71f5186520590c1252ce80d",
                "49fb20ec84050f439d130c08153e4f72d26522733b6778e06455c835eb1c8c05"],

                ["12c3b4ccddd81b3f54c9fd328054ee686e04a72ff08549b730bed27770be6202",
                "991b72c3afc642cf56dc5c9cbef914dd1ad56c76058ce906ba3a6d4e2a68d50a"],
                ["a0da4ff357a7c6c251774aae029736f0a3ae30119cc2e61574dff348419afe0c",
                "6b670fd437e2196058e3bc586387db43935986c0ef00b55e52f3be5109f23302"],
                ["984ae2debb28ac31f89c086853d58569788ccc21bcd6a229354c053e45518e07",
                "0b72112372f26ceb42feeea139811d27561b3d2b54412f7d10b4e7c879ff340a"],
                ["2e79d42e85b5977bae09c315447369331a3729e4d6fb2f62b03a0ae31a01d001",
                "33ba5e9af98da96c54a16065058b2f16b763eae22a71e795ccbacc6e25c14b0c"],
                ["372d185ab1786c1b25cbf58a575b0622053b1263f0e7b33cd43ad23fd07bad08",
                "5fb6d201d249f710cd5fe727fff2eb9d8fe2b2ab6008fe46b5b51f20a26b4202"],
                ["1ef57373019f83b487d45fbaf85edfe7027bdc9a36cc1489eab6b9c8fec85304",
                "e2b16eeab63c4324ad680d0d8ad02e7d250da6b70035700f097fdfd7f1e6c60c"],
                ["313b7340cd4a8af314a0f2bc568d857bb8381bdb1cb5b73b0ac4cef4fc52790f",
                "0b52e0ab9490cf47b59316326ae273e348789f762ca3129b3064f2a602bfcf0d"],
                ["2ba9d1f0585a854cddca0329f43399a4010d236ace3f9b5fdf3354b53ebf8b05",
                "32508f7b570ae1b2ae92d5f650b6092faf989dd32a43665b9a4ebec91f996b01"]],
            "cc":"5a8b6bcfbac12e13cfc7f7a54eabf7267238f5049efedb192282d826b4188309"}],
        "pseudoOuts":["2d80d3bd06ca8bad8d6bd8776f816cb9b6595580e9cea6eae64db962e5c358cc"]
}}}
```


Рисунок 2.19 – Приклад транзакції у Monero

Наявність групи виходів необхідне для формування кільцевого підпису. У нашому випадку key_offsets містить 11 значень. Це означає, що, крім ключової пари відправника, використовується ще 10 публічних ключів інших користувачів для обчислення кільцевого цифрового підпису. Поле k_image містить значення зображення секретного ключа та дозволяє уникнути атаки подвійної витрати.

Після входів розташовані виходи транзакції. У нашому випадку кількість виходів дорівнює 2, кожен з яких містить два поля: amount та key. Поле amount містить суму виходу транзакції (в «засліпленому» вигляді), а key – відкритий ключ одержувача. Ключ одноразовий і жодна третя сторона не може пов'язати його з головним (master) відкритим ключем одержувача (метод stealth addresses).

Всі інші дані транзакції містять докази того, що транзакція правильна і може бути підтверджена. Фактично, ці докази – цифрові підписи (включаючи дані range proofs та ring signatures). Розмір цих доказів становить понад 70% розміру транзакції. Середній розмір транзакції Monero більш ніж у 8 разів перевищує розмір стандартної транзакції в Bitcoin.

2.6 Протокол MimbleWimble

Протокол MimbleWimble э протоколом побудови мережі з високим рівнем конфіденційності користувачів і високою масштабованістю системи [18].

Згідно з дослідженням [19] можна виділити наступні проблеми, які вирішує MimbleWimble:

– Перша проблема полягає в тому, що більшість цифрових активів вимагають збереження повної історії транзакцій, яка з часом може лише збільшуватись. Відповідно, валідаторам необхідно виділяти все більше фізичної пам'яті для зберігання історії транзакцій (інакше повноцінна верифікація транзакцій буде неможлива).

– Другою проблемою є складність забезпечення конфіденційності платежів у permissionless system. Якщо ми візьмемо облікову систему Bitcoin, то в класичному варіанті деталі транзакції вказуються відкритий ключ, підпис відправника монет, сума переказу та адреса одержувача. Таким чином, у валідаторів є можливість перевірити, що конкретний користувач має монети, які збирається витратити, а також перевірити, що сума виходів транзакції не перевищує суму її входів. Відкритість є найважливішою особливістю облікової системи Bitcoin, проте через це не забезпечується конфіденційність переказів, а це означає, що деанонімізувати учасників транзакції дуже просто. Крім того, можна втратити властивість взаємозамінності монет (монети, з якими взаємодіяли власники конкретних адрес, можуть виділятися серед інших монет).

Аналіз функціонування протоколу MimbleWimble [19] свідчить про те, що у ньому об'єднуються кілька концепцій: confidential transactions для забезпечення конфіденційності переказів, range proofs для доказу, що сума конкретного входу і виходу не виходить за певні межі, CoinJoin для підвищення заплутування історії монет, а також cut-through для оптимізації розмірів зберігання розмірів транзакцій та блоків.

Властивості протоколу MimbleWimble:

– конфіденційність перекладів;

– простота масштабованості;

– відсутність необхідності зберігати історію транзакцій у повному вигляді;

– відсутність у деталях транзакції адрес користувачів.

2.6.1 Модель транзакцій MimbleWimble

Протокол MimbleWimble підтримує модель UTXO для проведення транзакцій. Це означає, що транзакція містить у собі набір входів (посилання на попередні виходи та докази володіння ними) та набір виходів, що містять суму переказів та інформацію про нового власника монет. Зазначимо, що

виходи витрачаються повністю (не можна витратити лише частину конкретного виходу), а решта є окремим виходом, довести володіння яким може сам відправник.

Давайте розглянемо приклад, у якому поступово перейдемо від повністю відкритих транзакцій (подібних до транзакцій у Bitcoin) до транзакцій з механізмами забезпечення конфіденційності. Припустимо, Аліса має намір відправити Бобу 8 монет. У Аліси є два невитрачені виходи на 3 і 5 монет. Тоді транзакція виглядатиме так, як зображено на рисунку 2.20.

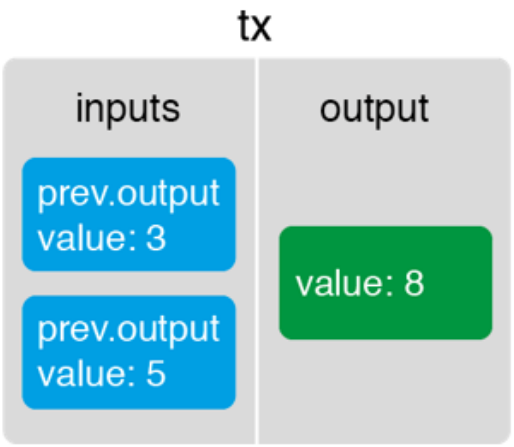

Рисунок 2.20 – «Незасліплені» значення входів та виходів транзакції

Для підтвердження транзакції валідатори перевіряють, що Аліса дійсно володіє виходами попередніх транзакцій, а також, що сума виходів транзакції, що підтверджується, не перевищує суми її входів. У цьому випаду це легко перевірити, оскільки суми відкриті (все, що потрібно зробити валідаторам, – підсумувати всі входи і виходи та порівняти результат). Однак Аліса не бажає, щоб хтось із валідаторів знав, скільки монет він передає. Для цього їй потрібно «засліпити» (приховати) значення входів та виходів транзакцій.

Для цього у протоколі MimbleWimble передбачено використання зобов'язань Педерсена для кожного значення входу та виходу транзакції. Зобов'язання Педерсена можна описати наступною формулою:

$$v * H + x * G, \qquad (2.5)$$

де, v – сума конкретного входу чи виходу; H – загальновідома точка на

еліптичній кривій; *x* – «засліплююче» значення (секрет згенерований користувачем); *G* – ще одна загальновідома точка на еліптичній кривій.

На наступному етапі необхідно використовувати додатковий «засліплюючий» секрет. Якщо ми приховуватимемо суму, що передається, тільки за рахунок множення її на базову точку, властивість незворотності могла б спрацювати, але так як сума переказу обмежена невеликим значенням, друга сторона може підібрати його шляхом перебору і знайти відповідну суму.

Давайте розглянемо, як після такої модифікації зміняться значення входів та виходів транзакції. Наприклад, Аліса, маючи такі самі входи, як у попередньому прикладі, використовує для засліплення секретні значення 19 та 37. Таким чином, застосовуючи зобов'язання Педерсена, формується транзакція (рис. 2.21).

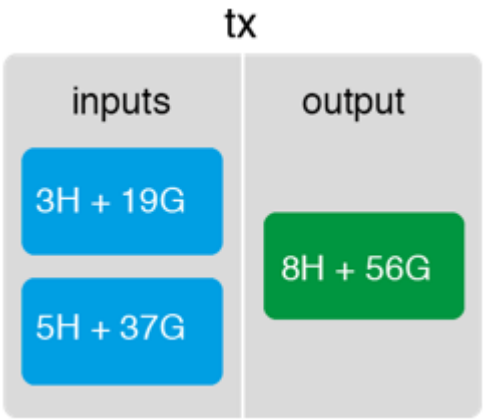

Рисунок 2.21– «Засліплені» значення входів та виходів транзакції

Тепер суми переказів приховані від валідаторів, проте все ще можна перевірити, що сума виходів не перевищує суму входів:

$$(3H + 19G) + (5H + 37G) - (8H + 56G) = 0, \qquad (2.6)$$

де, *H* та *G* – загальновідомі точки на еліптичній кривій.

Після аналізу даної транзакції можна зробити висновок, що в структурі даної транзакції є проблема. Необхідно, щоб сума «сліпучих» факторів на входах транзакції дорівнювала сумі «сліпучих» факторів на її виходах. Відповідно, після відправлення такої транзакції Аліса знає «сліпучий» фактор

(секрет) Боба, а значить, вона може вкрасти у нього монети.

За допомогою протоколу MimbleWimble можна вирішити цю проблему. Припустимо, Боб згенерував своє секретне значення, яке дорівнює 45. Тоді транзакція буде мати наступний вигляд (рис. 2.22).

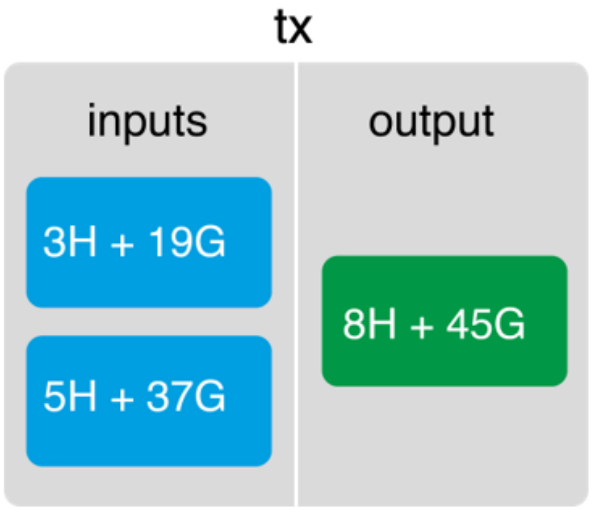

Рисунок 2.22 – Осліплені значення входів та виходів транзакції

При цьому сума виходів і входів не дорівнює *0* як раніше.

$$(3H + 19G) + (5H + 37G) – (8H + 45G) = 11G, \qquad (2.7)$$

де, *H* та *G* – загальновідомі точки на еліптичній кривій.

Як у цьому випадку валідаторам перевірити, що транзакція не створює нових монет? Якщо сума входів дорівнює сумі виходів, отриманий залишок буде відкритим ключем на *G*. Значення, отримане після обчислення (*11G*) називається ядром транзакції. Щоб підтвердити транзакцію, валідатори повинні перевірити, що це значення дійсно є відкритим ключем на кривій і що сторони спільно володіють спільним секретним ключем від нього.

Найпростішим способом довести це є цифровий підпис. Механізм цифрового підпису у протоколі MimbleWimble дозволяє обчислити таке загальне значення сторонам, які володіють частинами загального секретного ключа.

2.6.2 Range Proofs

Якщо проаналізувавши попередню транзакцію можна відзначити наступну проблему – вищеописана схема працює тільки якщо значення входів та виходів позитивні. Однак у користувача залишається можливість провести кілька виходів, один або кілька негативних (рис. 2.23).

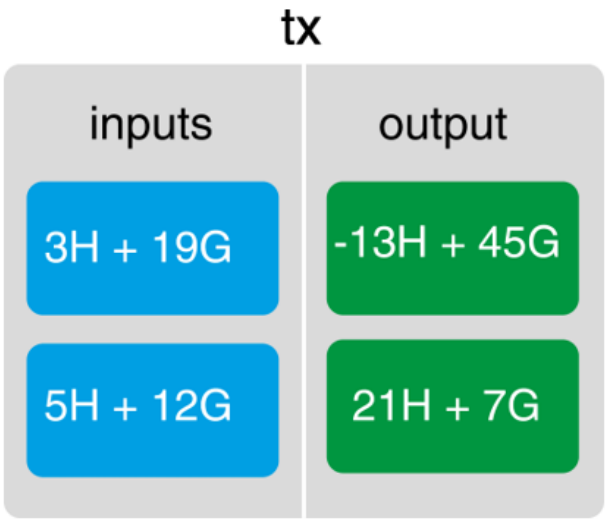

Рисунок 2.23 – «Засліплені» значення входів та виходів транзакції

Як можна побачити, в одному з виходів транзакції є негативне значення суми переказу, однак при цьому загальна сума сходиться, і для валідаторів транзакція вважається коректною:

$$(3H + 19G) + (5H + 37G) - (-13H + 25G) - (21H + 7G) = 24G, \quad (2.8)$$

де, H та G – загальновідомі точки на еліптичній кривій.

Для доказу того, що суми виходів та входом не є негативними, а також не перевищують максимально допустиме значення, використовуються докази діапазону (range proofs) [20].

2.6.3 Етапи проходження транзакції

Після того, як ми розглянули основні особливості транзакцій у протоколі MimbleWimble, необхідно розглянути, яким чином така транзакція формується, передається та перевіряється. Наприклад, Боб має невитрачений

вихід на 100 монет і хоче передати Алісі лише 50. Боб має намір заплатити 5 монет комісії і 45 повернути як здачу. Тому він формує транзакцію (рис. 2.24) наступного типу (*V1* – секрет для розблокування монет, *V2* – секрет для отримання здачі, *H* та *G* – загальновідомі точки на еліптичній кривій, *m* – додаткові дані).

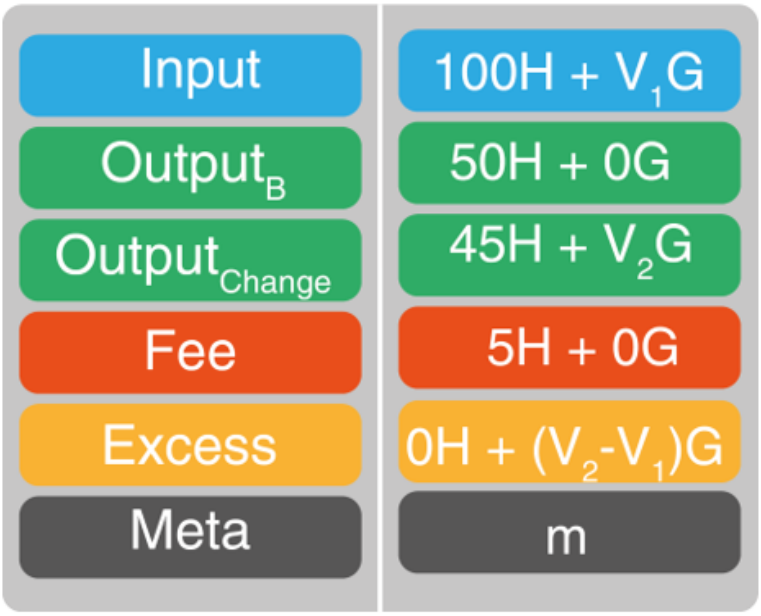

Рисунок 2.24 – Заготівля транзакції, що передається off-chain від відправника одержувачу

Також Боб, за допомоги протоколу MimbleWimble, підготовлює докази діапазону кожного з виходів транзакції. Після цього він відправляє отриману транзакцію Алісі.

Аліса при отриманні транзакції виконує перевірку полів, після чого генерує власне секретне значення («засліплюючий» фактор), формує відкритий ключ та обчислює значення виходу, адресованого їй. Після цього Аліса підписує ядро транзакції, яке покриває надмірне комісію та метадані транзакції. Аліса повертає Бобу набір наступних значень (рис. 2.25).

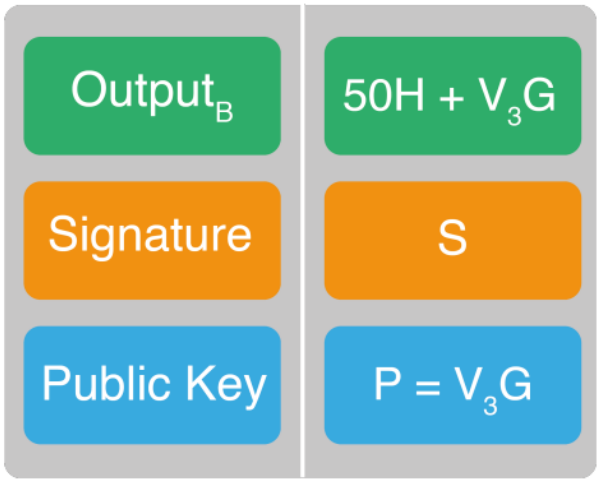

Рисунок 2.25 – Набір даних, що передається від одержувача відправнику

При отриманні цього набору даних Боб має можливість сформувати кінцеву транзакцію та відправити її до мережі для підтвердження. Транзакція складається фактично з двох частин: тіла та ядра. Тіло транзакції зберігає значення всіх входів та виходів транзакції. Ядро транзакції зберігає загальний відкритий ключ, значення підпису, значення комісії та додаткові метадані (рис. 2.26).

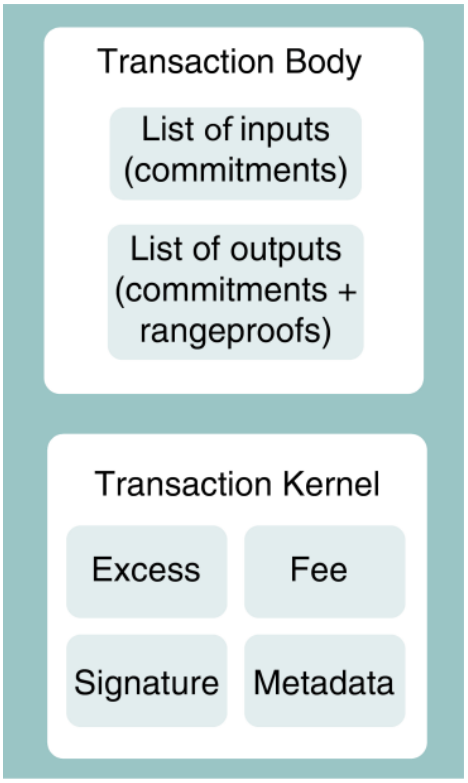

Рисунок 2.26 – Структура транзакції

2.6.4 Перевірка та розповсюдження транзакції

Коли Боб закінчив формування транзакції, він відправляє її одному з валідаторів. Вузлу, що отримав транзакцію, необхідно перевірити, що транзакція валідна, перш ніж передати її іншим вузлам мережі. Перевірка транзакції виконується наступним чином:

1. Валідатор перевіряє, що всі входи транзакції невитрачені. Кожен вузол відповідно до історії транзакцій зберігає кінцевий стан у вигляді списку невитрачених виходів. Якщо вихід транзакції Боба є в цьому списку, то цей вихід може бути успішно витрачений.

2. Валідатор перевіряє, чи знаходяться суми у виходах транзакції у встановленому діапазоні (перевіряє range proofs).

3. Валідатор перевіряє, чи дорівнює сума входів сумі виходів та комісії. Для цього йому необхідно скласти всі суми на входах та виходах транзакції та переконатися, що отримане значення є точкою на еліптичній кривій (валідним відкритим ключем).

4. Після цього перевіряється, чи відповідає підпис ядра транзакції отриманому відкритому ключу.

5. Далі виконуються системні перевірки, які вже не належать до правил протоколу (наприклад, перевіряють, чи достатній розмір комісії для того, щоб транзакцію підтвердили валідатори).

2.6.5 Відсутність адрес

У протоколі MimbleWimble не використовуються адреси. Кожна транзакція має бути ініційована як відправником монет так і їх одержувачем. Для формування транзакції, сторони спілкуються off-chain (безпосередньо один з одним). Необхідно зазначити, що монети не можуть бути передані без участі одержувача (на відміну від децентралізованої системи Bitcoin, де можна відправляти монети на будь-яку адресу без очікування будь-яких дій від одержувача).

У результаті взаємодії відправник вказує невитрачені виходи, і навіть їх

суми. Одержувач формує вихід, вказуючи обчислене «засліплене» значення. Після цього обидва користувачі формують доказ того, що суми виходів не перевищують суми входів, а також докази діапазону.

2.6.6 Метод cut-through

Досвід впровадження інформаційних технологій в децентралізовані мережі [21] свідчить про те, що протокол MimbleWimble використовує метод cut-through для видалення надлишкових виходів, які вказані в тому самому блоці і як входи. Цей метод дозволяє скоротити місце в блоці та зменшити кількість даних, які повинні зберігатися у ланцюжку блоків.

Таким чином, валідатори все ще можуть переконатися, що в блоці не було створено нових монет (за винятком нагороди за видобуток блоку), оскільки суми на входах та виходах врівноважують одна одну.

Більше того, таке видалення виходів (скорочення історії передачі монет) може використовуватися не тільки в межах одного блоку, а також між кількома блоками (оскільки входи одних транзакцій у будь-якому випадку посилаються на виходи попередніх, за винятком Coinbase транзакцій). В результаті розмір ланцюжка блоків може зменшуватися під час функціонування системи, якщо валідатор формує блок з великою кількістю входів (що знищують попередні виходи) і малою кількістю виходів. Тобто фактично валідатори можуть скорочувати входи та виходи, але при цьому вони повинні зберігати історію всіх ядер транзакції.

2.6.7 Структура транзакції та блоку

Блок MimbleWimble складається із заголовку, набору входів, набору виходів та набору ядер транзакцій.

Давайте розглянемо структуру заголовка блоку на прикладі підтвердженого блоку в мережі (рис. 2.27).

```
"version":1,
"height":5403,
"previous":"00000e40c1042fa655cf236d9f45241e712851114dce03dbd1fbf19625b2805c",
"prev_root":"d4e2d39e441f0598795234d6adf18793cdaf9caa6023d0a24512e70461d3f695",
"timestamp":"2019-01-19T14:15:09+00:00",
"output_root":"b24b1486c19ad65894373ba3afe7efe0ee08f0f8b621d2c47dfba08af661f303",
"range_proof_root":"073116e54caec3e85a988df44a486bd3d12ccec7448f728ef12316f665d44462",
"kernel_root":"18e229cb9b0535f2809aac04178637fda8035502018cef7700d2ecf1034b3b9e",
"nonce":11481980022248833820,
"edge_bits":29,
"cuckoo_solution":[ ⊞ ],
"total_difficulty":2998596263144,
"secondary_scaling":484,
"total_kernel_offset":"e4c68b605211368228e0dab18e2c25bc46197ce75aad477f8b761b9d59e2732f"
```


Рисунок 2.27 – Приклад заголовка блоку у децентралізованій системі Grin

Отже, кожен заголовок блоку містить версію протоколу, згідно з правилами якого він був сформований. Також заголовок блоку містить хеш значення попереднього блоку і значення Merkle root від полів заголовка попереднього блоку (корінь Merkle mountain ranges [22]). Наступним значенням є мітка часу формування блоку. Далі три значення Merkle Root: для виходів блоку, для доказів діапазону, для ядер транзакцій. За ними можна побачити набір значень, пов'язаних з розв'язанням ресурсномісткої задачі: складність, алгоритм, що використовується тощо. Останнім елементом блоку є агреговане значення ядер усіх валідних транзакцій.

2.7 Механізми захисту конфіденційності користувачів в мережі

У цьому підрозділі розглянуто найпопулярніші підходи підвищення рівня анонімності користувача мережі за рахунок приховування тих чи інших даних від інших користувачів чи сторонніх спостерігачів.

2.7.1 Підходи до анонімізації користувачів в мережі

Передача трафіку через проксі-сервер, використання SSH-тунелю, підключення через VPN або до інших посередників дозволяє приховати

мережеву адресу користувача мережі від ресурсу, до якого він звертається. А сторонній спостерігач, який бачить трафік між користувачем і сервером проксі, не може визначити до якого саме ресурсу звертається користувач.

Недолік такого підходу полягає в тому, що сам посередник, який грає роль проксі сервера, знає, до яких ресурсів підключаються всі його користувачі. Крім цього, користувачі зазвичай налаштовують з'єднання з одним проксі сервером і використовують його тривалий час, що дозволяє сторонньому спостерігачеві та самому ресурсу не знаючи мережевої адреси користувача стверджувати, що всі дії виконані якимось одним користувачем. Такий підхід також не є ефективним проти глобального спостерігача, який аналізуючи трафік у всій мережі одночасно побачить кореляцію за часом і розміром пакетів між відправником та одержувачем.

Підхід з використанням Onion Routing («цибульної» маршрутизації) [23] або її модифікацій дозволяє приховати мережеві адреси одержувача та відправника не лише від стороннього спостерігача, а й від посередників. А наявність великої кількості вузлів, з яких користувач може сам вибирати посередників, дозволяє йому досить часто перебудовувати маршрут і навіть використовувати кілька різних ланцюжків посередників одночасно для різних підключень. Проте такий підхід все одно неефективний проти глобального спостерігача і не захищає користувача від навмисних спроб ресурсу привласнити йому ідентифікатор на підставі таких даних як назва та версія браузера, особливості виконання ним скриптів, роздільна здатність екрану, збереження та використання даних про попередні сесії тощо.

Використання попереднього підходу разом зі спеціалізованим на анонімності браузером захищає користувача від низки способів його ідентифікації ресурсом до якого він підключається. Такий браузер не видає справжні дані про операційну систему та браузер користувача, про підключені плагіни та справжню роздільну здатність екрана, а також дозволяє повністю відключити виконання скриптів і збереження даних про попередні сесії. Завдання спеціалізованого браузера – зробити так, щоб усі його користувачі

«виглядали однаково». Очевидно, що використання такого браузера буде неефективним, якщо користувач відвідує ресурси, які вимагають реєстрації і при цьому використовує один і той же обліковий запис.

Підхід з використанням систем поширення даних з великими затримками дозволяє приховати зв'язок між відправником, одержувачем та самими даними, що передаються між ними. При цьому навіть глобальні спостерігачі не можуть побачити кореляцію в передачі даних між учасниками такої системи. Основний принцип такого підходу полягає в тому, що дані передаються через посередників, але весь маршрут передачі заздалегідь не відомий. Одне повідомлення може бути розбите на кілька складових частин, кожна з яких передаватиметься незалежно від інших.

Для передачі від посередника до посередника частини різних повідомлень об'єднуються випадковим чином і очікують на випадкову кількість часу, а до одержувача вони доходять в асинхронному порядку. Час надсилання одного повідомлення від відправника одержувачу може досягати 30 хвилин. Очевидно, що організований подібним чином зв'язок не є системою передачі загального призначення. Вона не може використовуватись для доступу до традиційних HTTP-серверів, аудіо/відео дзвінків або листування в режимі реального часу. Протоколи, які використовують цей підхід для підвищення анонімності, зазвичай реалізують ізольоване середовище спілкування на кшталт електронної пошти, публічних форумів або середовище для передачі транзакцій.

2.7.2 Використання Onion Routing для анонімізації з'єднань

Це система проксі-серверів, що дозволяє встановлювати анонімне з'єднання, захищене від прослуховування. Розглядається як анонімна мережа віртуальних тунелів, що дозволяє передавати дані у зашифрованому вигляді.

Пропускна здатність та рівень анонімності мережі Tor залежать від кількості вузлів – чим більше, тим краще. Це пов'язано з тим, що пропускна здатність кожного вузла обмежена, а велика кількість вузлів складніше

аналізувати. Згідно з дослідженням [24] у клієнтів мережі Tor налічується понад два мільйони вузлів, а передачею трафіку займаються понад 7000 активних вузлів.

За замовчуванням користувачі Tor передають трафік через 3 випадково вибрані вузли, таким чином трафік проходить наступний шлях: користувач (client), вхідний вузол (guard relay), проміжний вузол (middle relay), вихідний вузол (exit relay), пункт призначення (destination).

Спочатку користувач зашифровує дані так, щоб їх міг розшифрувати лише вихідний вузол. Далі отриманий шифротекст зашифровується так, щоб його міг розшифрувати тільки проміжний вузол. А потім цей шифротекст знову зашифровується так, щоб його міг розшифрувати лише вхідний вузол (рис. 2.28).

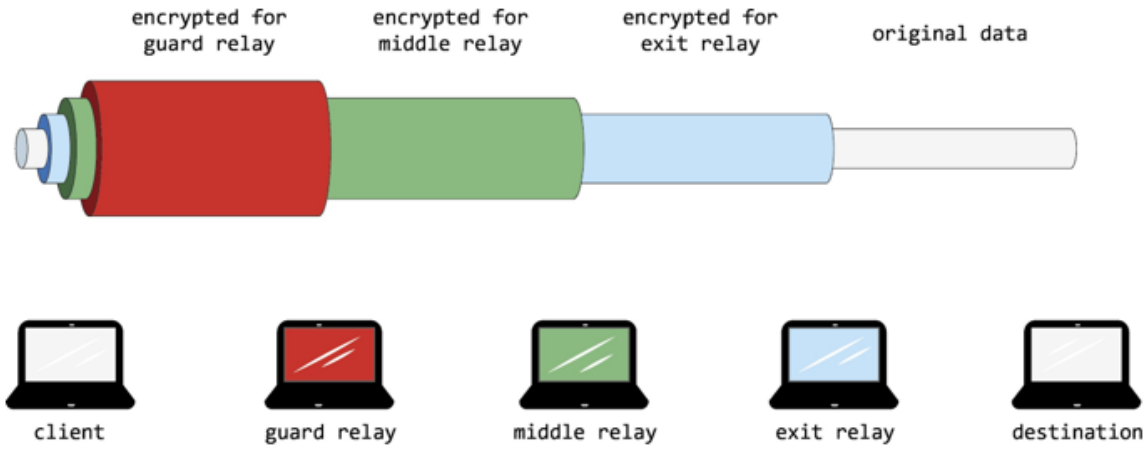

Рисунок 2.28 – Схема обробки даних, що передаються в Tor

2.7.3 Мережі з підвищеним рівнем анонімності користувачів

Для підвищення рівня анонімності користувачів в мережі був започатковано окремий проект для анаонімізації дій користувачів.

Internet invisible project (I2P) – це проект, що реалізує P2P мережу передачі з підвищеним рівнем анонімності користувачів, що функціонує без центральних серверів. Відповідно до дослідження [25] проект I2P було розпочато у 2003 році з метою створення анонімного засобу розповсюдження

даних з низькими затримками та, порівняно з альтернативними засобами, більш розподіленого та стійкого до атак. Даний проект також заснований на передачі трафіку через ланцюжок посередників (проміжних вузлів), проте способи пошуку вузлів та організації потоку пакетів відрізняються.

Мережа I2P насамперед орієнтована на передачу даних між її учасниками. Тому обидва «співрозмовники» (наприклад, користувач та ресурс) повинні використовувати спеціальне ПЗ (I2P-маршрутизатор). Можливість користувача I2P підключитися до ресурсу у звичайному Інтернеті також підтримується, однак таке підключення відбувається через один із спеціально налаштованих вузлів (outproxy), яких у мережі набагато менше.

В даному розділі детально розглянуті стандарти, протоколи та методи захисту конфіденційності інформації в децентралізованих системах, а також протоколи та механізми захисту конфіденційності користувачів у мережі.

У наступному розділі найпопулярніші методи зведені у порівняльній таблиці. А також обрано один із методів для подальшої реалізації.

3 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ЗАХИСТУ КОНФІДЕНЦІЙНОСТІ ІНФОРМАЦІЇ В ДЕЦЕНТРАЛІЗОВАНИХ СИСТЕМАХ

У цьому розділі представлено аналіз вимог до конфіденційності та безпеки блокчейну, перш ніж показати, як поточні технологічні досягнення відповідають їм, та виділити їхні недоліки.

3.1 Аналіз властивостей механізмів захисту

Вимоги блокчейна щодо конфіденційності та безпеки відрізняються від звичайних систем через його децентралізовану роботу. Дійсно, блокчейн забезпечує конфіденційність та безпеку конфіденційних даних, оскільки користувач може здійснювати транзакції за допомогою відкритих та закритих ключів, не розкриваючи своєї реальної особи. Однак публічний блокчейн не гарантує конфіденційності транзакцій, оскільки зміст транзакцій (наприклад, сума) є загальнодоступним [26]. Це призводить до витоку інформації, чутливої до конфіденційності. Крім того, біткоін-адреси користувачів можуть бути пов'язані з реальною особистістю користувачів [27]. Тому ми припускаємо, що для забезпечення конфіденційності та безпеки у блокчейні необхідно вибирати механізми на основі чітко сформульованих вимог. Нижче ми визначили та детально описали найбільш значущі вимоги.

– захист даних транзакції. Вміст транзакції (корисне навантаження), сама транзакція (наприклад, час та сума транзакції тощо) можуть потребувати захисту від несанкціонованого розкриття;

– захист особи користувача. Структура транзакцій встановлює прямий зв'язок із залученими користувачами за допомогою асиметричної криптографії (якщо з'ясувати, кому належить відкритий ключ). Як правило, користувачі блокчейну можуть виявити намір захистити зв'язок між своєю особистістю та транзакціями, які вони виконують, що робить захист особи одним з основних

питань для блокчейну;

– властивості безпеки. Необхідні типові властивості безпеки даних: конфіденційність для захисту від несанкціонованого доступу до даних, цілісність для впевненості в тому, що вміст є оригінальним, доступність, оскільки дані мають бути доступні на запит, і невідмовність, оскільки записані дані не можуть бути спростовані. За визначенням, реалізація блокчейна повинна забезпечувати такі властивості, тому завдання полягає в тому, щоб переконатися, що додаткові заходи, спрямовані на виконання інших вимог, не порушують жодної з цих властивостей;

– уникнення TTP. Централізовані бухгалтерські книги нинішніх транзакційних систем (наприклад, банків) схильні до вразливості єдиної точки відмови, тому їх легше скомпрометувати, ніж децентралізовані системи. Оскільки блокчейн допомагає усунути проблему TTP, додаткові рішення, що вирішують проблеми конфіденційності та безпеки, також повинні уникати опори на TTP.[28];

– підтримка блокчейну. В таблиці 3.1 буде показано, як вивчені механізми конфіденційності та безпеки підтримуються у реалізації блокчейну.

Таблиця 3.1 – Огляд вимог до блокчейну та існуючих механізмів захисту конфіденційності

Механізм захисту	Захист особи користувача	Захист даних транзакції	Властивості безпеки	Уникнення TTP	Підтримка блокчейну
CoinJoin	Немає	Є частково	Доступність, конфіденційність	Немає	Bitcoin BC
Chaumian CoinJoin	Є	Є	Доступність, конфіденційність	Немає	Bitcoin BC
CoinShuffle	Немає	Є	Доступність, конфіденційність	Є	Bitcoin BC, Ethereum BC
Blind Signature	Є	Є	Конфіденційність, цілісність	Є	Bitcoin BC, Permission BC

Продовження таблиці 3.1

Механізм захисту	Захист особи користувача	Захист даних транзакції	Властивості безпеки	Уникнення TTP	Підтримка блокчейну
Zero-knowledge Proof (zk-SNARKs)	Є	Є частково	Цілісність, конфіденційність, доступність	Є	Bitcoin BC, Zcash BC
Confidential Transactions	Є	Є	Конфіденційність	Немає	Bitcoin BC, Monero BC
Ring Confidential Transactions	Є частково	Є	Конфіденційність	Часткове (за вибором користувача)	Monero BC
CryptoNote	Є	Є частково	Цілісність, конфіденційність, доступність	Часткове (за вибором користувача)	Bytecoin BC, Monero BC
MimbleWimble	Є	Є	Цілісність, конфіденційність, доступність	Часткове (за вибором користувача)	Grin BC, Beam BC

Відповідно до цих вимог можна оцінити основні механізми збереження конфіденційності та безпеки, які підтримують анонімність транзакцій, підвищують анонімність даних та захист особи користувача, як показано в таблиці 3.1, та описати, як вони застосовуються до блокчейну.

Методи Confidential Transactions та Ring Confidential Transactions забезпечують конфіденційність даних та захист особи користувача за допомогою кільцевого підпису. Ці методи використовують тип криптографічного цифрового підпису, який спирається на групу користувачів (яка називається кільцем), з асиметричними ключами для підписання повідомлень. Після того, як повідомлення підписано, його можна розшифрувати лише за допомогою кільцевого підпису, і зловмисник не буде мати можливість визначити фактичного підписувача повідомлення всередині

групи. Як приклад реалізації протокол Ring CryptoNote приховує деталі транзакції (наприклад, суму, походження, пункт призначення) в децентралізованій криптовалюті Monero [29]. Однак кільцевий підпис вимагає від TTP управління ідентифікаторами користувачів, а вартість генерації та перевірки кільцевого підпису зростає через цифрові сертифікати [27]. Кільцевий підпис на основі ідентифікації дозволяє подолати ці проблеми та підвищити конфіденційність користувачів, а також запобігає атаці з повним розкриттям ключа [30].

Доказ із нульовим знанням (ZKP) дозволяє одному суб'єкту (користувачу, який перевіряє) довести іншому суб'єкту (верифікатору), що задане значення істинно, не розкриваючи жодної інформації, крім тієї, що сам доказ вірний. Рішення на основі ZKP є особливо цікавими для забезпечення цілісності даних та аутентифікації без розкриття інформації, оскільки вони забезпечують доказ затвердження без розкриття цього конкретного твердження. Вони також зберігають анонімність для конфіденційних даних та не покладаються на TTP. Однак, порівняно з іншими рішеннями, вони вимагають великих обчислювальних витрат для перевірки та генерування доказів [31].

Методи «змішування» приймають вхідні дані від кількох користувачів і надсилають їх наступному адресату відповідно до різних проміжків часу. Ці методи рандомізують порядок транзакційних даних, тому зловмиснику складніше відстежити комунікацію та транзакції між користувачами. Однак мережа блокчейну стає ТТП і має проблеми із затримкою порівняно з іншими механізмами. Одним із прикладів реалізації блокчейна є CoinJoin. З іншого боку, маршрутизація Onion та маршрутизація Tor дозволяють здійснювати анонімну комунікацію, забезпечуючи захист даних користувача без будь-яких TTP.

Асиметрична ключова криптографія на прикладі методу Blind Signatures, (криптографія з відкритим ключем, наприклад, RSA [32]), є ефективною як для ідентифікації користувача, так і для забезпечення

анонімності даних транзакції завдяки своїм властивостям безпеки, наприклад, конфіденційності та автентифікації даних [33]. Вона заснована на парі ключів (відкритим та закритим) для шифрування та розшифровування повідомлень. Даний метод спирається на цифрові підписи для перевірки того, що повідомлення надійшло від конкретного відправника (якщо воно зашифроване закритим ключем відправника) або для гарантії того, що тільки конкретний одержувач може його прочитати (якщо воно зашифроване відкритим ключем одержувача). Однак, у порівнянні з криптографією з симетричним ключем, вона вимагає великих обчислювальних витрат, що робить її менш придатною для шифрування великих обсягів даних. Насправді, зазвичай, для управління ідентифікаційними даними користувачів потрібно TTP, хоча управління ключами може бути реалізовано індивідуально.

Протокол CryptoNote – це протокол на основі якого побудовано сімейство анонімних криптовалют, найбільш відомими з яких є Bytecoin та Monero. Анонімність у CryptoNote реалізована за рахунок використання кільцевих підписів (приховують відправника) і одноразових адрес (приховують одержувача). Блокчейн на основі цього протоколу є стійким до аналізу тому, що дотримується двом вимогам забезпечення конфіденційності, а саме:

– відсутність можливості відстеження. Для кожної транзакції всі відправники рівноймовірні;

– незв'язаність. Для двох будь-яких вихідних транзакцій неможливо довести, що вони відправлені однією і тією самою особою.

Протокол MimbleWimble – це формат і протокол блокчейну, що надає виняткову масштабованість, приватність та знеособленість криптовалюти, спираючись на сильні криптографічні примітиви. Структура транзакцій вказує на ключові принципи MimbleWimble: непорушну гарантію приватності та конфіденційності. Даний протокол вже реалізований та використовується у таких децентралізованих системах як Grin та Beam. Блокчейни на основі MimbleWimble не залежать від TTP, але можуть використовувати його для

збільшення швидкості обробки транзакцій.

3.2 Аналіз характеристик механізмів захисту

Щоб більш детально порівняти існуючі механізми захисту конфіденційності, до розгляду додано наступні критерії: складність (СК), масштабованість (МШ), витрати на розмір транзакції (ВРТ), вартість пам'яті (ВП), ефективність (ЕФ), витрати на обчислення (ВО), витрати на зв'язок (ВЗв), витрати на затримку (ВЗт). У таблиці 3.2 наведено глобальний аналіз існуючих технологій за цими критеріями.

Складність – це загальний показник, який допомагає оцінити, наскільки «дорогим» буде рішення після його прийняття, виходячи з його математичної складності. Масштабованість дає уявлення про те, наскільки порівнювані рішення підходять для великомасштабних блокчейнів (Bitcoin DC, Ethereum BC, Monero BC тощо). Витрати на розмір відображають вартість застосування рішення з точки зору простору для зберігання даних. Вартість пам'яті показує обсяг пам'яті, який потрібний для застосування рішення, що допомагає виміряти продуктивність рішення. Ефективність визначає швидкість та середній час виконання, який потрібен механізму захисту для виконання поставленого завдання, що впливає на продуктивність розв'язання. Обчислювальні витрати показують вартість операцій механізму забезпечення конфіденційності та безпеки. Комунікаційні витрати вимірюються кількістю байт у кожному повідомленні, що передається мережею, а затримка означає скільки часу потрібно для передачі інформації в пункт призначення.

Таблиця 3.2 – Порівняльний аналіз підходів до забезпечення конфіденційності у децентралізованих системах

Механізм захисту	СК	МШ	ВРТ	ВП	ЕФ	ВО	ВЗв	ВЗт
CoinJoin	Залежить від кількості учасників транзакції	Так	Залежить від кількості учасників транзакції	Середня	Висока	Висока	–	Низькі
Chaumian CoinJoin	Залежить від кількості учасників транзакції	Так	Залежить від кількості учасників транзакції	Висока	Середня	Низька	Середні	Середні
CoinShuffle	Залежить від кількості учасників транзакції	Так	Залежить від кількості учасників транзакції	Середня	Висока	Висока	Високі	Низькі
Blind Signature	Середня	Так	Високі	Середні	Висока	Середня	Низькі	Середні
Zero-knowledge Proof (zk-SNARKs)	Висока	Ні	Низькі	Висока	Висока	Середня	–	Високі
Confidential Transactions	Середня	Ні	O(n), збільшується зі збільшенням розміру групи	Низькі	Середня	Середня	Низькі	Низькі
Ring Confidential Transactions	Низька	Ні	O(n), збільшується зі збільшенням розміру групи	Середні	Висока	Висока	–	Середні
CryptoNote	Висока	Ні	Високі	Високі	Висока	Висока	Середні	Високі
MimbleWimble	Висока	Так	Високі	Середні	Висока	Середня	–	Високі

Згідно з таблицею, метод Ring Confidential Transactions є відносно ефективним і працює з дуже низькою затримкою. Властивості безпеки (конфіденційність) та захист транзакцій даних [29] також забезпечуються без зменшення витрат на розмір підпису. Однак керування ключами та великий

розмір підпису є недоліками, оскільки кількість пар ключів лінійно залежить від кількості учасників кільця.

Доказ нульового знання (ZKP) [34] є перспективним рішенням для підтвердження транзакцій та забезпечення конфіденційності даних у блокчейні. Даний механізм добре підходить для забезпечення анонімної аутентифікації та захисту блокчейну від атаки "людина посередині" [35]. Проте доказ нульового знання є малоефективним для великомасштабних систем, оскільки потребує великого обчислювального часу та ресурсів пам'яті для створення та перевірки доказів.

Методи на основі механізму CoinJoin, запропоновані в [36], досягають високих результатів як в ефективності, так і в масштабованості. Дані методи мають низький розмір витрат на розмір транзакції, оскільки розмір вхідних даних лінійно залежить кількості мікс-серверів. Запропонована схема не може зменшити складність, затримку, витрати на пам'ять та обчислювальні витрати через високу вартість генерації вхідного шифротексту та вартість генерації ключів шифрування, що використовуються мікс-серверами. Можна виділити метод CoinShuffle, який не використовує мікс-сервери. За допомогою цього методу користувачі здійснюються міксування транзакцій самостійно. Це позитивно впливає на витрати на затримку, але негативно впливає на швидкість обчислення даних.

Метод Blind Signature забезпечує безпечний зв'язок між відправником та одержувачем за допомогою відкритих та закритих ключів [37]. Проаналізувавши даний метод ми можемо сказати, що він має високі обчислювальні витрати часу через час генерації ключів та часу шифрування/дешифрування, що робить шифрування/дешифрування складнішим для великих розмірів даних [38]. З позитивних сторін можна виділити високу ефективність та невисоку складність методу.

Провівши аналіз механізмів захисту на основі протоколу CryptoNote можна сказати, що дані механізми є високо ефективними, але мають високу складність, вартість пам’яті та витрати на зв’язок. Через це у блокчейнів на

основі протоколів CryptoNote можуть виникнути проблеми зі здатністю до масштабування.

Розглянувши протокол MimbleWimble можна сказати, що він є не менш ефективним ніж протокол CryptoNote, але він має набагато меншу вартість обчислень та пам’яті. Також забезпечення властивостей безпеки (цілісність, конфіденційність, доступність) не впливає на можливість протоколу до масштабування.

3.3 Аналіз блокчейнів на основі представлених механізмів захисту

Останній крок для оцінки існуючих підходів до захисту конфіденційності – порівняння найбільш поширених блокчейнів (криптовалют), які використовують вищеописані механізми захисту конфіденційності даних користувачів в децентралізованих системах.

В таблиці 3.3 використані наступні критерії:

– тип (наприклад публічний блокчейн, фреймворк, публічний блокчейн з елементами приватної мережі тощо);

– незмінність блокчейну – властивість, відсутність якої буде означати те, що до блокчейну можна вносити зміни;

– рівень анонімності користувачів. Користувачі блокчейну можуть виявити намір захистити зв'язок між своєю особистістю та транзакціями, які вони виконують;

– рівень підтримки та модифікації налаштувань – це рівень пристосованості блокчейну до впровадження змін та модифікацій;

– алгоритм консенсусу - це спосіб, завдяки якому децентралізовані вузли мережі досягають згоди щодо поточного стану даних у всіх блоках;

– підтримка смарт-контрактів. Смарт-контракт – криптографічний алгоритм, призначений для формування, контролю та надання інформації про володіння чимось (наприклад криптовалютою);

– алгоритм генерування гешу. Алгоритм хешування, займає важливу роль у сучасній криптографії, часто використовується для досягнення цілісності даних та аутентифікації суб'єкта, а також є гарантією безпеки для декількох криптосистем та протоколів;

– час генерації блоку – час генерації нового коректного блоку;

– час підтвердження транзакції – час потрібний для підтвердження транзакції (може залежати від завантаженості мережі);

– рівень масштабованості. Масштабованість відноситься до здатності системи розвиватися відповідно до зростаючого попиту;

– вірогідність атаки «51%». Атака «51%» – це вразливість PoW-блокчейнів, за допомогою якої зловмисник захоплює контроль над підтвердженням транзакцій та генерацією блоків;

– стійкість до DDoS атак. Метод виконання такої атаки полягає у наповненні мережі великою кількістю дрібних транзакцій;

– P2P комунікація. P2P – це технологія побудови розподіленої мережі, де кожен вузол може одночасно виступати як ролі клієнта (одержувача інформації), так і у ролі сервера (постачальника інформації);

– можливість мультипідпису – це технологія підписання транзакцій кількома приватними ключами для підвищення рівня безпеки та конфіденційності у процесі схвалення відправлення транзакцій;

– розмір транзакцій – це середній розмір транзакцій у блокчейні.

Таблиця 3.3 – Порівняння блокчейнів, які використовують вищеописані механізми захисту

Назва блокчейну (механізми захисту, які застосовуються)	Bitcoin (Confidential Transactions, CoinJoin)	Monero (Ring Confidential Transactions, ZKP, Stealth Addresses, Single-use Addresses)	Dash (CoinShuffle, Blind Signatures)	Zcash (ZKP (ZK-SNARKs), Blind Signatures)	Ethereum (ZKP, CryptoNote)	Grin (Mimble Wimble)
Тип	Публічний блокчейн	Публічний блокчейн	Публічний блокчейн	Публічний блокчейн	Публічний блокчейн	Публічний блокчейн
Незмінність блокчейну	Так	Так	Так	Так	Так	Так
Рівень анонімності користувачів	Низький	Високий	Середній	Високий	Середній	Високий
Рівень приховання даних транзакцій	Низький	Високий	Середній	Високий	Низький	Середній
Рівень підтримки та модифікації налаштувань	Дуже низький	Низький	Низький	Низький	Середній	Середній
Алгоритм консенсусу	Proof of Work	Proof of Work	Proof of Work + Proof of Stake	Proof of Work + Zk-SNARK	Proof of Work	Proof of Work
Має підтримку смарт-контрактів	Частково	Ні	Ні	Ні	Ні	Ні
Алгоритм генерування гешу	SHA-256	CryptoNight	X11	ZKP	Ethash	NiceHash
Час генерації блоку	10 хв.	2 хв.	2,5 хв.	2,5 хв.	16 сек.	2 хв.
Час підтвердження транзакції	1 год.	20 хв.	30 хв.	15 хв.	3 хв.	10 хв.
Рівень масштабованості	Низький	Середній	Середній	Низький	Середній	Високий
Вірогідність атаки «51%»	Низька	Середня	Середня	Середня	Низька	Висока
Стійкість до DDoS	Є	Є	Немає	Є	Є	Є
P2P комунікація	Є	Є	Є	Є	Є	Немає

Продовження таблиці 3.3

Назва блокчейну	Bitcoin	Monero	Dash	Zcash	Ethereum	Grin
Можливість мультипідпису	Є	Є	Є	Немає	Є	Є
Розмір транзакцій	Середній	Великий	Малий	Середній	Малий	Нижче середнього

Як можна побачити у таблиці 3.3 найбільш захищеними та збалансованими блокчейнами є Monero BC, Zcash BC та Grin BC. Ці блокчейни використовують наступні механізми захисту: Ring Confidential Transactions, ZKP, Stealth Addresses, Single-use Addresses, ZKP (ZK-SNARKs), Blind Signature, MimbleWimble. Проаналізувавши протоколи та методи захисту можна відмітити, що більшість механізмів використовують різні методи «засліплення» даних. Алгоритм Chaumian CoinJoin та протокол CryptoNote використовують метод Blind Signature, а протокол MimbleWimble – зобов’язання Педерсена. Також метод Blind Signature використовується у декількох блокчейнах окремо від інших алгоритмів чи протоколів.

Проаналізувавши таблиці 3.1, 3.2 та 3.3 можна відмітити, що алгоритм Blind Signature забезпечує високий рівень конфіденційності та є ефективним у багатьох випадках. Досвід впровадження інформаційних технологій в децентралізовані мережі свідчить про те, що даний метод є відносно простим для реалізації та не потребує великої кількості обчислювальних ресурсів.

В даному розділі проаналізовані вищеописані підходи до забезпечення конфіденційності користувачів в децентралізованих системах. Були розглянуті характеристики та властивості даних методів, а також був проведений огляд блокчейнів, які використовують ці механізми та протоколи.

Для реалізації в даній роботі був вибраний метод Blind Signature через його розповсюдженість, невелику складність реалізації та високу ефективність.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ BLIND SIGNATURE

4.1 Загальні відомості

Програма складається з 1 файлу – blind.exe. Розмір програми – 18,7 Мб. Програма розроблена у середовищі Visual Studio 2019, за допомогою мови програмування C++ з використанням наступних бібліотек: Cryptlib, Integer, Nbtheory, Osrng, Rsa, Sha.

4.2 Функціональне призначення

Дана програма призначена для демонстрації роботи алгоритму створення та верифікації «сліпого» підпису. В даній програмі продемонстровано генерація вхідних даних, генерація повідомлення, хешування, шифрування та підписання повідомлення.

4.3 Опис логічної структури

Алгоритм збору даних та їх обробки відповідає опису, наведеному у підрозділі 2.2.

Повний код програми наведено у Додатку А.

Нижче будуть розглянуті основні блоки програми.

На рисунку 4.1 наведено програмний код, який генерує ключову пару (приватний та публічний ключі). Для демонстраційних цілей було згенеровано 64 бітні ключі. Взагалі в даному алгоритмі повинні використовуватись ключі 1024 біт і більше.

```
AutoSeededRandomPool prng;
RSA::PrivateKey privKey;

privKey.GenerateRandomWithKeySize(prng, 64);
RSA::PublicKey pubKey(privKey);
```

Рисунок 4.1 – Приклад програмного коду генерації ключів

На рисунку 4.2 зображено програмний код, який за допомогою функцій GetModulus(), GetPublicExponent(), GetPrivateExponent() генерує модуль публічного ключа, публічну експоненту, приватну експоненту відповідно.

```
const Integer& n = pubKey.GetModulus();
const Integer& e = pubKey.GetPublicExponent();
const Integer& d = privKey.GetPrivateExponent();
```

Рисунок 4.2 – Приклад програмного коду генерації вхідних параметрів

На рисунку 4.3 наведено програмний код, за допомогою якого було згенеровано повідомлення, яке потім необхідно буде зашифрувати та підписати.

```
SecByteBlock orig((const byte*)"secret", 6);
Integer m(orig.data(), orig.size());
std::cout << "Message: " << std::hex << m << std::endl;
```

Рисунок 4.3 – Приклад програмного коду генерації повідомлення, яке необхідно підписати

На рисунку 4.4 зображено програмний код, за допомогою виконується хешування повідомлення. Хешування виконується за допомогою алгоритму SHA256. В даному алгоритму Blind Signature хешування повідомлення виконується з метою запобігання вразливості розкриття ключа. Дана особливість була описана у підрозділі 2.2.

```
buff1.resize(SIG_SIZE);
SHA256 hash1;
hash1.CalculateTruncatedDigest(buff1, buff1.size(), orig, orig.size());

// H(m) as Integer
Integer hm(buff1.data(), buff1.size());
std::cout << "H(m): " << std::hex << hm << std::endl;
```

Рисунок 4.4 – Хешування повідомлення, яке необхідно підписати

На рисунку 4.5 наведено блок програми, за допомогою якої повідомлення зашифровується.

```
Integer r;
do {
    r.Randomize(prng, Integer::One(), n - Integer::One());
} while (!RelativelyPrime(r, n));

Integer b = a_exp_b_mod_c(r, e, n);
std::cout << "Random: " << std::hex << b << std::endl;


Integer mm = a_times_b_mod_c(hm, b, n);
std::cout << "Blind msg: " << std::hex << mm << std::endl;
```

Рисунок 4.5 – Шифрування повідомлення

На рисунку 4.6 наведено блок програми, за допомогою якого повідомлення підписується.

```
Integer ss = privKey.CalculateInverse(prng, mm);
std::cout << "Blind sign: " << ss << std::endl;
```

Рисунок 4.6 – Підписання повідомлення

На рисунку 4.7 зображено блок коду, за допомогою якого ми перевіряємо підписане повідомлення.

```
Integer ck = pubKey.ApplyFunction(ss);
std::cout << "Check sign: " << ck << std::endl;
if (ck != mm)
    throw std::runtime_error("Alice cross-check failed");
```

Рисунок 4.7 – Перевірка підписаного повідомлення

На рисунку 4.8 наведено блок програми, за допомогою якого ми розшифровуємо повідомлення.

```
Integer s = a_times_b_mod_c(ss, r.InverseMod(n), n);
std::cout << "Unblind sign: " << std::hex << s << std::endl;
```

Рисунок 4.8 – Розшифрування підписаного повідомлення

На рисунку 4.9 зображено блок коду, за допомогою якого ми перевіряємо отримане підписане повідомлення.

```
Integer v = pubKey.ApplyFunction(s);
std::cout << "Verify: " << std::hex << v << std::endl;

size_t req = v.MinEncodedSize();
buff2.resize(req);
v.Encode(&buff2[0], buff2.size());


buff3.resize(SIG_SIZE);
SHA256 hash2;
hash2.CalculateTruncatedDigest(buff3, buff3.size(), orig, orig.size());

bool equal = buff2.size() == buff3.size() && VerifyBufsEqual(
    buff2.data(), buff3.data(), buff3.size());

if (!equal)
    throw std::runtime_error("Eve verified failed");

std::cout << "Verified signature" << std::endl;
```

Рисунок 4.9 – Перевірка підписаного повідомлення

4.4 Вхідні дані

У якості вхідних даних виступають відкриті та закриті ключі користувачів (генеруються програмно), модулі відритих та закритих ключів (генеруються програмно), повідомлення, що передається (генерується програмно).

4.5 Вихідні дані

Вихідними даними програми виступають згенероване повідомлення, захешоване повідомлення, зашифроване повідомлення, підписане зашифроване повідомлення, підписане повідомлення (рис. 4.10).

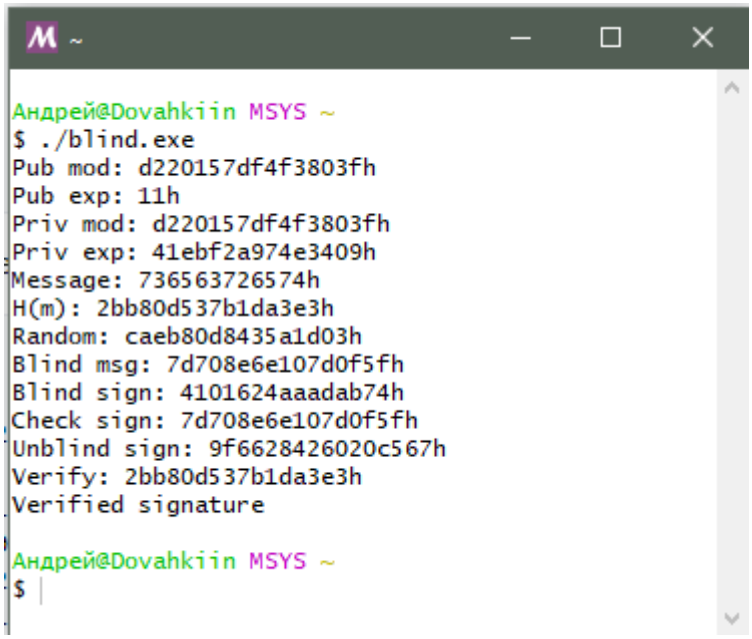

Рисунок 4.10 – Результат роботи програми

Метод Blind Signature, на основі якого розроблена дана програма, є нескладним для імплементування в різні децентралізовані системи. Наприклад, можна реалізувати алгоритм, який дозволяє генерувати сліпий підпис, сумісний з існуючим протоколом Bitcoin. Користувач генерує набір параметрів підпису та синтезує відкритий ключ для використання в транзакції Bitcoin. З метою здійснення транзакції, користувач зашифровує хеш своєї транзакції та відправляє іншому вузлу мережі для підпису, а потім розшифровує отримане від підписувача підписане та зашифроване повідомлення. Підписана транзакція публікується в блокчейні Bitcoin. Імплементація даного алгоритму значно збільшить рівень конфіденційності цієї децентралізованої системи.

ВИСНОВКИ

Підсумовуючи плюси і мінуси кожного методу захисту конфіденційності даних в складній системі блокчейн, яка повинна задовольняти вимоги щодо приватності користувачів з бажаними властивостями, ми хотіли б зробити наступні три зауваження.

По-перше, жодна технологія не є панацеєю для забезпечення конфіденційності в блокчейні. Таким чином, відповідні методи захисту приватних даних користувачів повинні обиратися на основі вимог клієнтів та контексту застосування. Загалом, поєднання декількох технологій працює більш ефективно, ніж використання однієї технології.

По-друге, не існує технології, яка не має дефектів або є досконалою у всіх аспектах. Коли новий підхід додається до складної системи, він завжди викликає появу інших проблем або нових форм атак. Тому вимагається уважне ставлення до потенційних збитків, спричинених інтеграцією деяких методів безпеки в блокчейні.

По-третє, завжди існує компроміс між забезпеченням конфіденційності даних та ефективністю. Потрібно просувати ті алгоритми та підходи, які поліпшують безпеку блокчейну і в той же час сприяють його розвитку та можливості практичного розгортання таких додатків з прийнятною продуктивністю.

У першому розділі розглянуті поняття конфіденційності та її складові. Було визначено, які атрибути активності користувачів та які дані про транзакції необхідно приховувати.

У другому розділі детально розглянуті стандарти, протоколи та методи захисту конфіденційності інформації в децентралізованих системах.

У третьому розділі проаналізовані вищеописані підходи до забезпечення конфіденційності користувачів в децентралізованих системах. Були розглянуті характеристики та властивості даних методів, а також був проведений огляд

блокчейнів, які використовують ці механізми та протоколи.

Для реалізації в даній роботі вибраний метод Blind Signature через його розповсюдженість, невелику складність реалізації та високу ефективність.

В четвертому розділі наведено опис програмної реалізації алгоритму Blind Signature. Дана програмна реалізація призначена для демонстрації роботи алгоритму створення та верифікації «сліпого» підпису. В даній програмі продемонстровано генерація вхідних даних, генерація повідомлення, хешування, шифрування та підписання повідомлення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bitcoin: A peer-to-peer electronic cash system [Електронний ресурс] / S. Nakamoto. – 2008. – Режим доступу: https://bitcoin.org/bitcoin.pdf.

2. Confidential Assets [Електронний ресурс] / Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille. – Режим доступу: https://blockstream.com/bitcoin17-final41.pdf.

3. Reid F. An Analysis of Anonymity in the Bitcoin System [Електронний ресурс] / F. Reid and M. Harrigan // Security and Privacy in Social Networks. – Springer New York, 2013. – P. 197–223. – Режим доступу: https://users.encs.concordia.ca/~clark/biblio/bitcoin/Reid%202011.pdf.

4. Herrera-Joancomartí J. Privacy in bitcoin transactions: New challenges from blockchain scalability solutions / J. Herrera-Joancomartí and C. Perez-Solá // Modeling Decisions for Artificial Intelligence: 13th International Conference, MDAI 2016. – Springer International Publishing, 2016.

5. Meiklejohn S. A fistful of bitcoins: characterizing payments among men with no names [Електронний ресурс] / Sarah Meiklejohn, Marjori Pomarole, Grant Jordan et al. // Proceedings of the 2013 conference on Internet measurement conference, 2013. – Режим доступу: https://cseweb.ucsd.edu/ ~smeiklejohn/files/imc13.pdf.

6. Ruffing T. P2P Mixing and Unlinkable Bitcoin Transactions. Anonymity of the people, by the people, and for the people / T. Ruffing, P. Moreno-Sanchez, A. Kate // IACR Cryptology ePrint Archive, 2017.

7. Сучасні механізми захисту конфіденційності інформації в децентралізованих системах [Електронний ресурс] – Режим доступу: https://ojs.ukrlogos.in.ua/index.php/liga/issue/ view/10.12.2021-ukr/688

8. G. Maxwell, "Coinjoin: Bitcoin privacy for the real world", 2013, [Електронний ресурс] – Режим доступу: https://bitcointalk.org/index.php ?topic=279249.

9. Ruffing T. CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin [Електронний ресурс] / Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. – Режим доступу: https://petsymposium.org/2014/papers/ Ruffing.pdf.

10. Алгоритм сліпого підпису [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Blind_signature.

11. Blum, Manuel; Feldman, Paul; Micali, Silvio (1988). Non-Interactive Zero-Knowledge and Its Applications. Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC 1988). p. 103–112.

12. Maxwell G. Confidential Transactions [Електронний ресурс] / Greg Maxwell. – Режим доступу: https://people.xiph.org/~greg/confidential_values.txt.

13. Van Saberhagen N. CryptoNote v 2.0 [Електронний ресурс] / Nicolas van Saberhagen – Режим доступу: https://cryptonote.org/whitepaper.pdf.

14. CryptoNote Standards [Електронний ресурс] – Режим доступу: https://cryptonote.org/standards/.

15. Stealth addresses [Електронний ресурс] – Режим доступу: http://www.mail-archive.com/bitcoindevelopment@lists.sourceforge.net/msg0361 3 .html.

16. Zero to Monero [Електронний ресурс] – Режим доступу: https://web.getmonero.org/library/Zero-to-Monero-1-0-0.pdf.

17. Merkle Root [Електронний ресурс] – Режим доступу: https://www.investopedia.com/terms/m/merkle-root-cryptocurrency.asp.

18. MIMBLEWIMBLE [Електронний ресурс] – Режим доступу: https://scalingbitcoin.org/papers/mimblewimble.txt.

19. Mimblewimble Commit [Електронний ресурс] – Режим доступу: https://mimblewimble.cash/20161006-WhitePaperUpdatee9f45ec.pdf.

20. Jonathan Bootle Bulletproofs: Short Proofs for Confidential Transactions and More [Електронний ресурс] – Режим доступу: https://eprint.iacr.org/2017/1066.pdf.

21. Cutting through the Blockchain noise [Електронний ресурс] – Режим доступу: https://medium.com/slalom-technology/cutting-through-the-blockchain-noise-aa41b0847ab9.

22. Mimblewimble Grin [Електронний ресурс] – Режим доступу: https://github.com/mimblewimble/grin/blob/master/doc/mmr.md.

23. Onion Routing [Електронний ресурс] – Режим доступу: https://cryptome.org/2014/08/onion-routing-security-2000.pdf.

24. TOR [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Tor.

25. Internet invisible project [Електронний ресурс] – Режим доступу: https://staas.home.xs4all.nl/t/swtr/documents/wt2015_i2p.pdf.

26. Henry, R.; Herzberg, A.; Kate, A. Blockchain access privacy: Challenges and directions. IEEE Secur. Priv. 2018, 16, p. 38–45.

27. Chow, S.S.; Yiu, S.M.; Hui, L.C. Efficient identity-based ring signature. In International Conference on Applied Cryptography and Network Security; Springer: Berlin/Heidelberg, Germany, 2005; p. 499–512

28. Ammous, S. Blockchain Technology: What Is It Good for? 2016. [Електронний ресурс] – Режим доступу: https://papers.ssrn.com/sol3/papers.cfm.

29. Noether, S. Ring SIgnature Confidential Transactions for Monero. IACR Cryptol. ePrint Arch., 2015, 1098.

30. Gamage, C.; Gras, B.; Crispo, B.; Tanenbaum, A.S. An identity-based ring signature scheme with enhanced privacy. In Proceedings of the 2006 Securecomm and Workshops, Baltimore, MD, USA, 28 August–1 September 2006; p. 1–5.

31. Bernabe, J.B.; Canovas, J.L.; Hernandez-Ramos, J.L.; Moreno, R.T.; Skarmeta, A. Privacy-preserving solutions for Blockchain: Review and challenges. IEEE Access 2019, 7, p. 164–169.

32. Jeeva, A.; Palanisamy, D.V.; Kanagaram, K. Comparative analysis of performance efficiency and security measures of some encryption algorithms. Int. J. Eng. Res. Appl. 2012, 2, p. 303–307.

33. Shah, S. Use of Blockchain as a Software Component to Send Messages Anonymously for a Data Trading Platform; Archemy: New Egypt, NJ, USA, 2017

34. Prabhakaran, M.; Sahai, A. Concurrent Zero Knowledge Proofs with Logarithmic Roundcomplexity [Електронний ресурс] – Режим доступу: http://eprint.iacr.org/2002/055.pdf.

35. Lu, L.; Han, J.; Liu, Y.; Hu, L.; Huai, J.P.; Ni, L.; Ma, J. Pseudo trust: Zero-knowledge authentication in anonymous P2Ps. IEEE Trans. Parallel Distrib. Syst. 2008, 19, p. 135–137.

36. Jakobsson, M.; Juels, A. An optimally robust hybrid mix network. In Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing, Newport, RI, USA, 26–29 August 2001; p. 284–292

37. Farah, S.; Javed, Y.; Shamim, A.; Nawaz, T. An experimental study on performance evaluation of asymmetric encryption algorithms. In Proceedings of the 3rd European Conference of Computer Science on Recent Advances in Information Science (EECS-12), Paris, France, 2–4 December 2012; p. 121–124

38. Maqsood, F.; Ahmed, M.; Ali, M.M.; Shah, M.A. Cryptography: A comparative analysis for modern techniques. Int. J. Adv. Comput. Sci. Appl. 2017, p. 442–448.