КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

Кваліфікаційна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою "Інформатика"
спеціальності 122 Комп'ютерні науки на тему:

РЕАЛІЗАЦІЯ ПРОТОКОЛУ АНОНІМНИХ ПЕРЕКАЗІВ НА ОСНОВІ ДОКАЗІВ З НУЛЬОВИМ РОЗГОЛОШЕННЯМ

Виконав студент 4-го курсу
Лев ПОТЬОМКІН _____
(підпис)

Науковий керівник:
Член-кореспондент НАН України, професор
Анатолій АНІСІМОВ _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри математичної інформатики

Протокол № _____ _____ 2023 р.

Завідувач кафедри
Василь ТЕРЕЩЕНКО _____
(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи: 42 сторінок, 2 ілюстрації, 1 таблиця, 15 використаних джерел.

Ключові слова: ДОКАЗИ З НУЛЬОВИМ РОЗГОЛОШЕННЯМ, АНОНІМНІ ПЕРЕКАЗИ, БЛОКЧЕЙН, КРИПТОГРАФІЯ, СМАРТ КОНТРАКТИ, ДЕРЕВО МЕРКЛА.

Об’єктом роботи є протокол анонімних переказів криптовалюти на базі блокчейну Ethereum.

Метою кваліфікаційної роботи є створення й публікація протоколу для анонімних переказів з відкритим вихідним кодом для покращення гнучкості таких переказів в публічних блокчейнах.

Інструментами створення є безкоштовний, вільно поширюваний редактор коду `Neovim` та мова розмітки тексту `Typst`, мови програмування `Noir`, `TypeScript`, `Solidity`. Використано фреймворк `Hardhat`, бібліотеку `ethers.js` та сервіс `Etherscan`.

Результат роботи: розроблено та опубліковано реалізацію протоколу анонімних переказів під ліцензією MIT.

ЗМІСТ

Скорочення та умовні позначення 5

Вступ 6

Розділ 1. Огляд доказів з нульовим розголошенням 8

1.1. Означення та властивості 8

1.2. Приклад: схема Шнорра 8

1.2.1. Обґрунтування протоколу Шнорра 9

1.3. Перетворення Фіата-Шаміра 10

1.4. zk-SNARK та довірене налаштування 11

1.5. Порівняння SNARK, STARK та Bulletproof 13

Розділ 2. Огляд існуючих протоколів анонімних переказів 15

2.1. ZCash 15

2.2. Monero 16

2.3. MimbleWimble 17

2.4. Aztec 19

2.4.1. Aztec Connect 20

Розділ 3. Реалізація 22

3.1. Опис протоколу 22

3.2. Множина анонімності 25

3.3. Релейери та анонімність у мережі 26

3.4. Дерево Меркла 27

3.4.1. Неявне дерево Меркла 28

3.4.2. Інкрементальне дерево Меркла 30

3.5. Протокол Aave 31

3.5.1. Ризики 32

3.5.2. Flash Loans 33

3.5.3. Інтеграція 34

3.5.4. Розрахунок потенційних прибутків 36

Висновки 38

Перелік джерел посилання 40

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

DAO – Decentralized Autonomous Organization, децентрелізована автономна організація;

DeFi – Decentralized Finance, децентралізовані фінанси;

DLP – Discrete Logarithm Problem, задача дискретного логарифму;

ERC20 – широко використовуваний стандарт токенів в блокчейні Ethereum;

EVM – Ethereum Virtual Machine, середовище виконання смарт-контрактів в блокчейні Ethereum;

L2 – Layer 2, рішення другого рівня;

MiMC – Minimal Multiplicative Complexity, сімейство хешів з мінімальною мультиплікативною складністю;

MPC – Multiparty Computation, протокол багатосторонніх обчислень;

PLONK – Permutations over Lagrange-bases for Oecumenical Non-interactive arguments of Knowledge;

SMT – Sparse Merkle Tree, неявне дерево Меркла;

SNARK – Succinct Non-Interactive ARgument of Knowledge;

STARK – Scalable Transparent ARgument of Knowledge;

UTXO – Unspent Transaction Output, невитрачений залишок транзакції – відокремлена сума криптовалюти з конктретним власником;

ZKP – Zero-Knowledge Proof, доказ з нульовим розголошенням;

ZKC – Zero-Knowledge Circuits, схеми з нульовим розголошенням;

ВСТУП

У сучасному цифровому світі анонімність та конфіденційність стають все більш важливими, особливо в контексті фінансових операцій на публічних блокчейнах. Рівень приватності постійно знаходиться під загрозою через розширення можливостей збору даних та нав’язування технологій слідкування. Станом на 2023 рік існують різні криптовалютні платформи та протоколи, які надають анонімність та конфіденційність при проведенні транзакцій, кожен зі своїми ризиками, гарантіями безпеки, унікальними функціями та недоліками. Однією з ключових технологій, що забезпечує таку анонімність, є докази з нульовим розголошенням.

Актуальність роботи полягає у постійно зростаючій потребі захисту персональних даних та анонімності у цифровому просторі. Протоколи анонімних переказів, засновані на доказах з нульовим розголошенням, дозволяють реалізувати безпеку і конфіденційність обміну коштами. Це має особливу важливість для таких областей, як фінанси та криптовалюти, де конфіденційність транзакцій є критичною. Незважаючи на те, що багато аспектів цих протоколів вже досліджено, ще залишається багато недосконалостей та можливостей оптимізації, вивчення яких є актуальним.

Основна мета роботи полягає в аналізі наявних протоколів анонімних переказів, вдосконаленні та реалізації модифікації існуючого протоколу на основі доказів з нульовим розголошенням

для покращення його гнучкості та сумісності з екосистемою блокчейну, оцінці ефективності нової реалізації.

Об’єктом дослідження є протоколи анонімних переказів та докази з нульовим розголошенням. У процесі дослідження використані теоретичні та практичні методи, включаючи аналіз наукової літератури, математичні моделі, симуляції та реалізацію прототипів.

Можливі сфери застосування результатів роботи включають криптовалютні платформи, банківські установи, фінтех-компанії та інші організації, які займаються електронними платежами та фінансовими операціями.

Дана робота базується на реалізації Tornado Cash –- відомого протоколу для анонімних транзакцій на блокчейні Ethereum. Основні принципи та ідеї його архітектури адаптовані та розширені для даної роботи, зокрема для використання мови програмування `Noir` та інтеграції протоколу Aave.

РОЗДІЛ 1. ОГЛЯД ДОКАЗІВ З НУЛЬОВИМ РОЗГОЛОШЕННЯМ

1.1. Означення та властивості

Доказ нульового розголошення можна розглядати як взаємодію між двома комп'ютерними програмами. Одну з них називають Прувер, а другу Верифікатор. Прувер переконує Верифікатора, що певне математичне твердження є істинним. Будь-який доказ нульового розголошення має задовольняти три властивості:

1. **Повнота** (Completeness): якщо Прувер чесний (твердження істинне), то врешті-решт він переконає Верифікатора.
2. **Обґрунтованість** (Soundness): Прувер може переконати Верифікатора лише в тому випадку, якщо твердження істинне.
3. **Нульове розголошення** (Zero knowledge): Верифікатор не дізнається жодної інформації, окрім факту істинності твердження.

1.2. Приклад: схема Шнорра

Схема Шнорра [1] є одним із найпростіших та найбільш широко використовуваних протоколів ZKP. Вона дозволяє стороні довести, що вона знає секретнне число, не розкриваючи його або будь-яку інформацію стосовно нього. Схема використовується, зокрема, в HTTPS та SSH.

Протокол наступний:

1. **Підготовка**: Перед початком протоколу сторона, яка доводить знання ключа (прувер), та сторона, якій доводиться (верифікатор), домовляються про певне велике просте число p і примітивний корінь g по модулю p. Верифікатор знає $y = g^x \bmod p$. Прувер хоче довести, що знає x, залишивши його в секреті.
2. **Комітмент**: Прувер обирає випадкове число r і відправляє $t = g^r \bmod p$ верифікатору.
3. **Виклик**: Верифікатор відправляє пруверу випадкове число c.
4. **Відповідь**: Прувер обчислює $s = r + cx \bmod p - 1$ і відправляє s верифікатору.
5. **Перевірка**: Верифікатор перевіряє, що $g^s \bmod p$ дорівнює $ty^c \bmod p$. Якщо рівність виконується, то верифікатор приймає доказ. В іншому випадку доказ відхиляється.

У цьому протоколі прувер ніколи не розкриває свій секрет x, але все одно може довести, що він його знає. Одночасно верифікатор може бути впевнений, що прувер знає секрет, не отримуючи жодної додаткової інформації про цей секрет.

Цей протокол використовує криптографічне припущення про нерозв’язність DLP.

1.2.1. Обґрунтування протоколу Шнорра

Нехай існує алгоритм, що може переконати Верифікатора, що він знає x не маючи його. Тоді нехай цей алгоритм успішно пройшов

два виклики c_1, c_2 Верифікатора для фіксованого x – тобто відправив s_1, s_2. Тоді

$$s_1 - s_2 \bmod p - 1 = c_1 x - c_2 x \bmod p - 1$$

$$\Rightarrow \frac{s_1 - s_2}{c_1 - c_2} \bmod p - 1 = x \bmod p - 1$$

Отже, цей алгоритм еквівалентний добуванню дискретного логарифму з $y = g^x = g^{\frac{s_1 - s_2}{c_1 - c_2}}$, що і треба було довести.

Повнота доводиться тривіально за допомогою малої теореми Ферма.

1.3. Перетворення Фіата-Шаміра

Перетворення Фіата-Шаміра [2] – це важлива техніка в криптографії, яка дозволяє перетворити інтерактивний протокол ZKP на неінтерактивний. Це важливо, тому що інтерактивність може бути проблематичною для багатьох реальних застосувань, особливо в онлайн-системах, де велика кількість взаємодій може бути обтяжливою.

У стандартному інтерактивному ZKP, прувер та верифікатор обмінюються декількома повідомленнями. У випадку протоколу Шнорра, наприклад, прувер починає з надсилання комітменту, потім верифікатор відправляє випадкове число як виклик, і прувер відповідає на цей виклик.

Перетворення Фіата-Шаміра дозволяє усунути необхідність в інтерактивності шляхом заміни виклику верифікатора на криптографічно безпечний хеш від комітменту прувера. Таким чином, прувер може створити доказ без взаємодії з верифікатором: він створює свій комітмент, обчислює хеш від цього комітменту, використовує цей хеш як виклик і відповідає на цей виклик. Згодом верифікатор може перевірити доказ, перевіривши, що хеш від комітменту прувера справді відповідає виклику і що відповідь правильно відповідає на виклик.

Це перетворення робить можливим застосування ZKP в неінтерактивних контекстах, таких як криптовалюти та блокчейн. Проте воно залежить від існування криптографічно безпечних хеш-функцій, які вважаються стійкими до злому.

1.4. zk-SNARK та довірене налаштування

Сімейство пруф-систем zk-SNARK [3] є одним з найпопулярніших на сьогодняшній день, оскільки дозволяє генерувати короткі докази, які можна швидко верифікувати – незалежно від розміру обчислень, що доводяться. Проте, вони мають великий спільний недолік.

Довірене налаштування (*trusted setup*) в zk-SNARK є важливою частиною процесу, оскільки воно генерує початкові параметри, які використовуватимуться під час створення та перевірки доказів.

Налаштування має бути виконано безпечним і надійним способом, оскільки від цього залежить безпека системи.

Одним із поширених способів проведення довіреного налаштування є протокол багатосторонніх обчислень (MPC):

1. Кожен учасник самостійно генерує частину параметрів налаштування та зберігає свою частину в таємниці.
2. Кожен учасник використовує свою секретну частину, щоб створити публічну частину параметрів налаштування.
3. Публічні частини від усіх учасників об’єднуються для створення остаточних параметрів для системи zk-SNARK.
4. Приватні частини учасників знищуються.

Властивість безпеки цієї установки полягає в тому, що доки принаймні один учасник успішно знищить свою секретну частину, остаточні параметри не можуть бути використані для створення хибних доказів.

Одним із прикладів такого підходу на практиці була «церемонія» ZCash, яка являла собою протокол багатосторонніх обчислень за участю шести учасників. Протокол був розроблений таким чином, що навіть якби п’ятеро учасників змовилися, щоб зберегти свої секрети та створити фальшиві докази, зусилля були б марними, доки принаймні один учасник правильно знищив свій секрет. Церемонія була проведена з високим рівнем свідомості безпеки, включаючи заходи для запобігання side-channel attacks, які могли стати причиною витоку секретів.

Хоча цей процес суттєво зменшує ризик, пов’язаний із довіреним налаштуванням, він не усуває його повністю. Таким чином, системи, які не вимагають надійних установок, як-от zk-STARK і Bulletproofs, можуть бути більш привабливими в певних контекстах.

1.5. Порівняння SNARK, STARK та Bulletproof

Тут описані приклади сучасних пруф-систем з їх перевагами та недоліками (Таблиця 1).

zk-SNARK. Частина **S** (*succinct*, стислий) означає, що докази невеликі та їх швидко верифікувати, що є ключовою перевагою. Однак вони вимагають окремий *trusted setup* для кожної схеми ZKC, що є суттєвим недоліком. Підвид SNARKів під назвою PLONK потребує один *trusted setup* для всіх схем не більше певного розміру (*universal trusted setup*), що однозначно є покращенням, хоч і не повним вирішенням проблеми.

zk-STARK [4]. Вони схожі на zk-SNARK, але не вимагають довіреного налаштування, що допомагає уникнути вищезгаданого ризику безпеки. Вони також стійкі до квантових алгоритмів, адже їх безпека залежить тільки від обраної хеш-функції. Однак докази, згенеровані zk-STARK, більші, ніж ті, що згенеровані zk-SNARK, що може бути недоліком.

Bulletproof [5]. Це ще один тип ZKP, який не вимагає довіреного налаштування. Вони в основному використовуються для

покращення конфіденційності в мережі Bitcoin. Розмір Bulletproof’a логарифмічно зростає як із розміром твердження, яке потрібно підтвердити, так і з параметром безпеки, що робить їх більш здатними до масштабування. Однак вони не такі стислі, як zk-SNARK або zk-STARK, тобто потребують більше обчислювальних ресурсів для верифікації.

Таблиця 1 – Порівняльна характеристика пруф-систем

	SNARK	**STARK**	**Bulletproof**
Алгоритмічна складність: Прувер	$O(n \log(n))$	$O(n \text{ poly-log}(n))$	$O(n \log(n))$
Алгоритмічна складність: Верифікатор	$O(1)$	$O(\text{poly-log}(n))$	$O(n)$
Розмір доказу	$O(1)$	$O(\text{poly-log}(n))$	$O(\log(n))$
Потрібен trusted setup	так	ні	ні
Стійкість до квантових алгоритмів	ні	так	ні
Криптографічні припущення	DLP + secure bilinear pairing	хеші, стійкі до колізій	DLP

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ ПРОТОКОЛІВ АНОНІМНИХ ПЕРЕКАЗІВ

2.1. ZCash

ZCash (ZEC) [6] це відкрита децентралізована криптовалюта, яка була запущена в 2016 році. Вона була створена з метою надання більш приватної альтернативи іншим криптовалютам, зокрема Bitcoin. В основу ZCash покладено технологію zk-SNARKs, описану в попередньому розділі.

ZCash дозволяє користувачам вибирати, чи робити свої транзакції приватними (*shielded transactions*), адже вони дорожчі за публічні транзакції та вимагають від користувача генерації доказу ZKP.

У ZCash, технологія zk-SNARK дозволяє створити транзакцію з доказом, що відправник має достатньо коштів для проведення транзакції, але без розкриття суми транзакції, відправника та отримувача. Це гарантує приватність транзакції.

Отже, як це працює на практиці? Нижче наведено кроки, які відбуваються при використанні конфіденційного переказу в ZCash:

1. **Створення комітменту**: Відправник генерує секретний ключ та публічний ключ. Відправник тоді бере вхідні UTXO, суму, яку вони хочуть відправити, та адресу отримувача, та створює «комітмент». Комітмент є просто хешем всіх цих деталей, який потім записується в блокчейн.

2. **Створення доказу**: Відправник тоді генерує ZKP, що вони знають секретний ключ, який відповідає публічному ключу в комітменті, і що вхідні UTXO містять достатньо коштів для виконання транзакції.
3. **Верифікація**: Коли відправник передає доказ в мережу, вузли в мережі можуть перевірити доказ без відкриття комітменту. Якщо доказ є вірним, транзакція додається в блокчейн, але самі деталі транзакції залишаються прихованими. Завдяки технології zk-SNARKs, ZCash забезпечує приватність транзакцій, не жертвуючи безпекою або цілісністю своєї мережі.

2.2. Monero

Monero (XMR) [7] – це приватна, децентралізована і безпечна криптовалюта, яка була впроваджена в квітні 2014 року. Вона стала популярною, завдяки наголосу на анонімності та приватності. Це забезпечено через застосування декількох криптографічних технологій:

1. **Кільцеві Підписи** (*Ring Signatures*): використовуються для забезпечення анонімності відправника. Вони дозволяють відправнику підписати транзакцію з групою можливих підписувачів, тим самим забезпечуючи анонімність, оскільки неможливо визначити, хто з групи дійсно є відправником.

2. **Сховані адреси** (*Stealth Addresses)*: Monero використовує сховані адреси для забезпечення анонімності отримувача. Коли транзакція виконується, генерується унікальна одноразова адреса, яка асоціюється з кожною транзакцією на стороні отримувача. Це гарантує, що зовнішній спостерігач не може прослідкувати дві транзакції до того ж отримувача.
3. **Кільцеві конфіденційні транзакції** (*RingCT*): використовуються для приховування суми транзакції. Це розширення технології кільцевих підписів, що використовує комітменти Педерсена, які є гомоморфними за додаванням (сума двох комітментів дорівнює комітменту суми). Вони дозволяють довести, що сума входів транзакції дорівнює сумі виходів, без необхідності відкривати фактичні суми.
4. **Криптовалютна мережа Kovri**: Monero планує впровадження мережі Kovri, що дозволить забезпечити анонімність на рівні IP-адреси, використовуючи технологію оніон-маршрутизації. Ця технологія вже давно використовуються, наприклад, в браузері Tor.

2.3. MimbleWimble

Mimblewimble [8] – це криптовалютний протокол, представлений у 2016 році. Він відрізняється тим, що надає високий

рівень приватності і значно більшу ефективність за масштабування в порівнянні з багатьма традиційними криптовалютами.

MimbleWimble використовує декілька криптографічних технологій для забезпечення анонімності переказів.

1. Приховування транзакцій

У MimbleWimble приховування транзакцій здійснюється за допомогою комітментів Педерсена (Pedersen Commitments), так само як і в Monero. Комітмент Педерсена – це гомоморфний за додаванням криптографічний примітив, який дозволяє зберігати значення таким чином, що воно приховане, але все ще може бути використане в математичних обчисленнях. Тобто ви можете додавати зашифровані числа, і отримати той самий (зашифрований) результат, якщо б ви виконували ці обчислення на незашифрованих числах.

Коли користувач виконує транзакцію, він створює комітмент для кожного залишку (UTXO) транзакції. Ці комітменти потім використовуються для перевірки того, що загальна сума входів транзакції дорівнює загальній сумі залишків. Це гарантує, що транзакція є дійсною, без необхідності відкривати реальні суми переказів.

2. Об’єднання транзакцій

MimbleWimble також використовує технологію, відому як CoinJoin, для об’єднання транзакцій. CoinJoin – це процес, в якому

кілька користувачів об’єднують свої транзакції в одну, щоб змішати інформацію про відправників і отримувачів.

Коли транзакції об’єднуються за допомогою CoinJoin, всі входи та виходи об’єднаних транзакцій перемішуються разом. Це означає, що неможливо визначити, який вхід відповідає якому виходу, що допомагає забезпечити анонімність.

Важливо зазначити, що об’єднання транзакцій в MimbleWimble відбувається на рівні протоколу, що означає, що всі транзакції автоматично об’єднуються, на відміну від деяких інших технологій CoinJoin, які вимагають активної участі користувачів.

Основою цих механізмів є криптографія на еліптичних кривих, на яких будуються ключові криптографічні примітиви, що використовуються в MimbleWimble.

2.4. Aztec

Протокол Aztec [9] – це рішення другого рівня (*L2*), орієнтоване на конфіденційність, розгорнуте в мережі Ethereum. Для досягнення своїх цілей, протокол використовує ZKP, зокрема zk-SNARKи, схоже на те, як це робить протокол ZCash. На жаль, з 21.03.2021 протокол більше не приймає депозити – розробники вирішили його згорнути.

Основною метою Aztec є забезпечення рівня конфіденційності та приватності на Ethereum, який зазвичай недоступний у транзакціях за замовчуванням. Крім того, Aztec створено для надання переваг масштабованості. Використання zk-SNARK

допомагає стиснути дані транзакцій, що робить їх потенційно ефективнішими (а отже і дешевшими), ніж традиційні транзакції Ethereum.

На відміну від ZCash, що є достатньо примітивним протоколом, Aztec підтримує унікальні функції, такі як:

1. **Приватні активи**. Aztec підтримує не тільки приватні перекази ефіру, а й ERC20 токенів. Це дозволяє використовувати конфіденційні аналоги будь-яких цифрових активів, що реалізують даний стандарт.
2. **Модель аккаунтів** замість UTXO. Це дозволяє бути більш сумісними з протоколом Ethereum ефективніше обробляти депозити та виводи активів з протоколу.
3. Інтеграція з DeFi протоколами через **Aztec Connect**.

2.4.1. Aztec Connect

Aztec Connect – це розширення протоколу Aztec, що дозволяє взаємодіяти з DeFi протоколами на Ethereum, використовуючи конфіденційні активи, що знаходяться в Aztec L2. Це було можливим завдяки підтримці спеціальних інфраструктурних смарт-контрактів «мостів», що інтегрували б Aztec з іншими протоколами, такими як Uniswap, Aave та ін.

Алгоритм приблизно такий:

1. Користувач, або кілька користувачів Aztec хочуть використати DeFi протокол «X».
2. Активи цих користувачів збираються та використовуються в одній транзакції на Ethereum, що виконується від імені протоколу Aztec – тож неможливо зрозуміти, хто саме з усіх користувачів скористався функцією. Що саме виконується у транзакції, залежить від протоколу «X».
3. Генеруються та валідуються ZKP про можливість виконання цієї транзакції користувачами.
4. Усі отримані активи в результаті транзакції (якщо є), розподіляються між користувачами-ініціаторами, як це прописано на контракті-мості.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ

3.1. Опис протоколу

За основу протоколу була взята архітектура Tornado Cash [10]. Протокол може бути розгорнутий на будь-якому EVM-сумісному блокчейні та складається з таких смарт-контрактів:

- `Tornado`, що буде мати інтерфейс для депозитів та виводів, а також буде підтримувати дерево Меркла (описано в розділі 3.4.) з комітментів користувачів.
- `Hasher`, що буде містити реалізацію криптографічної хеш-функції MiMC для використання у дереві Меркла.
- `Verifier`, що буде виступати у ролі верифікатора ZKP для виводів коштів.

Усі депозити та виводи мають мати однакову валюту та деномінацію, адже інакше можна буде при виведенні відслідкувати користувача, що робив депозити, компрометуючи анонімність переказу.

Алгоритм взаємодії наступний:

1. Користувач генерує два набори з 32 байтів s та n. s називається секретом, n називається обнулювачем (*nullifier*). Обраховується комітмент $C = H(s \parallel n)$, де H – хеш-функція MiMC.
2. Користувач робить депозит у смарт-контракт `Tornado`, надаючи контракту комітмент C.

3. Смарт-контракт зберігає C у дереві Меркла та оновлює його кореневий хеш.
4. Користувач відправляє надійним каналом значення s та n отримувачу коштів (офф-чейн, тобто поза блокчейном).
5. Отримувач обчислює $N = H(n)$ та генерує ZKP P, який доводить, що:
   - дійсно $N = H(n)$
   - отримувач знає такі s та n, що $C = H(s \parallel n)$
   - C належить дереву Меркла з кореневим хешем R (за допомогою шляху Меркла)
6. Отримувач відправляє N, R та P на смарт-контракт `Tornado` із запитом на вивід коштів.
7. Смарт-контракт перевіряє, що обнулювач N ще не був використаний, кореневий хеш R співпадає зі значенням на контракті, та делегує верифікацію доказу P контракту `Verifier`. Якщо всі умови виконані, контракт відправляє кошти отримувачу.

Варто зазначити, що контракт зберігає декілька попередніх кореневих хешів на випадок, якщо кілька користувачів одночасно хочуть зробити депозит та вивести кошти. Без цього, депозит змінив би хеш, транзакція виводу провалилась би, користувач втратив би гроші, витрачені на комісію та був би вимушений генерувати новий доказ.

У контракті `Hasher` хеш MiMC [11] використовується через його низьку мультиплікативну складність та відносну простоту реалізації в ZKC. Цей компроміс потрібен тому, що він буде обчислюватись на смарт-контракті $2N$ разів під час кожного депозиту для оновлення дерева Меркла (де N – висота дерева), що може стати причиною великих комісій на транзакцію у EVM-сумісних блокчейнах. Також, ZKC працюють набагато ефективніше з мультиплікативними хешами аніж з перестановочними. Використання, наприклад, хешу `keccak256` може спричинити надмірно великі витрати на генерацію доказу, хоч цей хеш і є одним з найдешевших у EVM, але він є перестановочним.

У контракті `Verifier` використовується пруф-система PLONK, що має кілька переваг над стандартними zk-SNARK (такими як Groth16, що використовується в оригінальній реалізації Tornado Cash):

- Через універсальне довірене налаштування, що підтримується в PLONK [12], можна використовувати один і той самий верифікатор для усіх екзмеплярів протоколу (різні валюти, різні деномінації і т.і.).
- ZKC можна писати на сучасній мові `Noir`, що є більш високорівневою та гнучкою ніж `Circom` з оригінальної реалізації.
- Докази досить легко генерувати прямо у браузері, через зручну прив’язку бібліотеки `barretenberg` до WebAssembly.

Очевидно, що без знання секретних s та n неможливо згенерувати ZKP для отримання коштів. Також, очевидно, що без знання s неможливо віднайти C знаючи N, а отже неможливо пов’язати між собою транзакції депозиту та виводу коштів. Це і робить перекази через даний протокол анонімними.

3.2. Множина анонімності

Множина анонімності (*anonymity set*) – це група транзакцій, які неможливо відрізнити одна від одної. Коли користувачі вносять або знімають кошти за допомогою Tornado, їхні транзакції змішуються з транзакціями інших користувачів, щоб підвищити конфіденційність і заплутати слід транзакцій.

Множина анонімності представляє собою загальну множину транзакцій, які потенційно можуть бути змішані з конкретною транзакцією. Чим більша множина анонімності, тим більший рівень приватності надається користувачам. Якщо множина анонімності велика, стає складніше відстежувати потік коштів і пов’язувати конкретні депозити з конкретними виводами.

Для протоколу Tornado розмір множини анонімності для транзакції виводу в будь-який момент часу дорівнює кількості користувачів, що здійснили депозит до цього часу, адже будь-хто з них може здійснювати вивід.

3.3. Релейери та анонімність у мережі

Релейери – це життєво необхідний компонент системи, що допомагає зберігати анонімність користувачів, функціонуючи як проксі між ними та протоколом.

Щоразу, коли отримувачу коштів потрібно вивести їх з контракту, він має відправити ранзакцію у блокчейн. Так як блокчейн-транзакції не безкоштовні, на акаунті отримувача вже мають бути кошти, аби він міг оплатити ними комісію за транзакцію. Проте, його кошти наразі знаходяться на смарт-контракті, і він не хоче компрометувати свою особистість, переводячи собі гроші для оплати комісії публічно.

Релейери вирішують цю проблему. Релейери – це треті особи, що не задіяні безпосередньо у переказі між відправником і отримувачем, а лише слугують як проксі, що відправляють транзакцію виводу з наданими їм даними, аби запобігти будь-яким публічним переказам отримувачу. За свої послуги вони можуть залишити собі певний відсоток від суми виводу. Варто зазначити, що усі дані транзакції, а також сума комісії та адреса отримувача визначаються заздалегідь та автентифікуються у ZKP, тож релейер не може їх змінити.

Варто також зазначити, що протокол не вирішує проблем конфіденційності офф-чейн. Якщо бути необережним, транзакції можна буде відслідкувати за їх метаданими, що отримують вузли

Ethereum чи інтернет-провайдери – такі як IP-адресу, сигнатуру веб-клієнта, та ін. Якщо бути необережними, ці метадані можна буде пов’язати з іншими транзакціями, таким чином відновивши зв’язок між ними та скомпрометувавши анонімність.

3.4. Дерево Меркла

Дерево Меркла [13], що також відоме як хеш-дерево, це структура даних, що представляє собою ідеальне двійкове дерево (тобто таке повне двійкове дерево, в якому листи лежать на однаковій відстані від кореня). Воно назване на честь Ральфа Меркла, що винайшов його в 1979 році.

Дерево Меркла складається з двох основних типів елементів:

- **Внутрішні вузли**: Це елементи дерева, які мають рівно два нащадки і містять хеш від конкатенації значень нащадків $H_{\mathrm{AB}} = H(H_A \parallel H_B)$.
- **Листи**: Це кінцеві вузли дерева, які містять фактичні дані або хеш від фактичних даних (в залежності від реалізації).

Основна відмінність від хеш-таблиці полягає в тому, що одна гілка хеш-дерева може бути завантажена у необхідний час, а цілісність кожної гілки може бути перевірена негайно, навіть якщо все дерево ще недоступне.

Основною перевагою дерева Меркла висоти N є можливість довести за $O(N)$ належність дереву певного елемента, маючи лише

кореневий хеш (*root hash*) та хеші сусідніх вузлів на шляху до відповідного листа. Цей алгоритм називають доказом Меркла, а масив сусідніх хешів – шляхом Меркла (Рисунок 1).

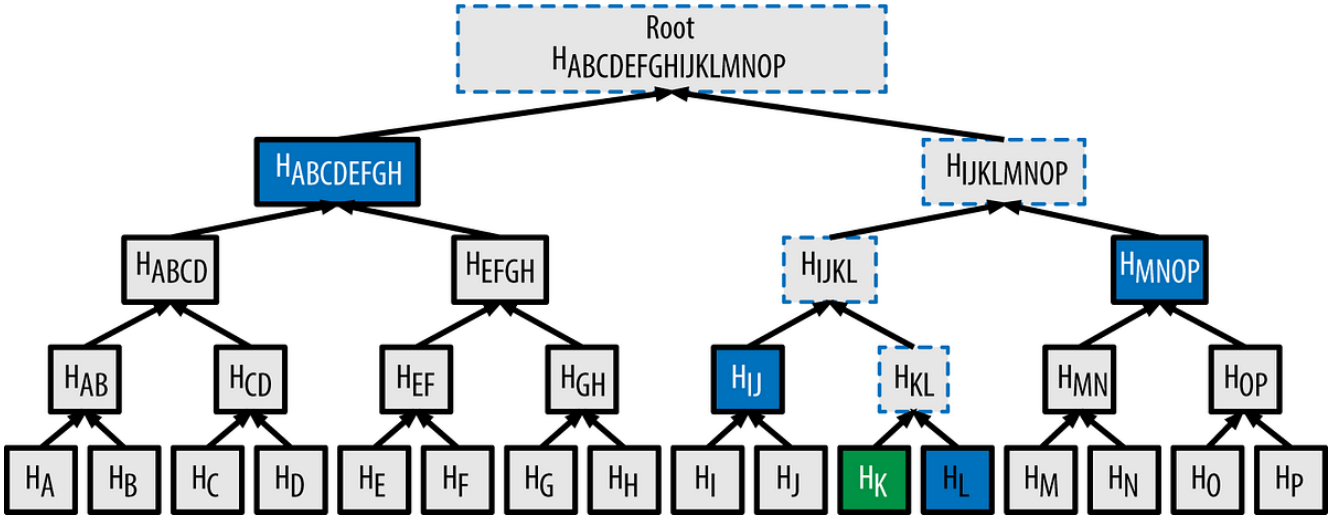

Рисунок 1 – Схема доказу Меркла

Дерево Меркла важливе в контексті блокчейну, оскільки воно дозволяє ефективно зберігати великі обсяги даних у якості рут-хешу (тобто хешу кореневого вузла) дерева Меркла, побудованого на цих даних. Це дозволяє здійснювати перевірку на належність (*membership proof*), надаючи шляхи Меркла.

Далі опишемо конкретні варіанти реалізації дерева Меркла, їх переваги та недоліки.

3.4.1. Неявне дерево Меркла

Неявне (або розріджене) дерево Меркла (*Sparse Merkle Tree*, SMT) – це варіант реалізації дерева Меркла, який дозволяє зберігати лише

$O(N \cdot K)$ вузлів де K – це кількість непустих листів. На відміну від наївної реалізації, де збрігається кожен з $2^N - 1$ вузлів.

Стандартне дерево Меркла має зберігати усі свої вузли явно. З іншого боку, неявне дерево Меркла розроблено таким чином, щоб бути ефективним, якщо дерево представляє велику кількість даних, більшість із яких є порожніми або мають значення за замовчуванням.

Це особливо корисно для застосунків в блокчейні, де дерево може відображати стан цілої книги з адресами для мільйонів або мільярдів облікових записів, більшість із яких може взагалі не мати жодних збережених значень.

SMT побудовано таким чином, що воно має достатньо листкових вузлів для представлення всіх можливих ключів даних (адрес, слотів тощо), але лише вузли на шляху від кореня до листа, зі значеннями, відмінними від дефолтних, потрібно зберігати або створювати, що економить багато місця.

Для дефолтних значень можна заздалегідь порахувати хеш на i-тому рівні дерева за рекуррентною формулою $H_i^{\text{default}} = H\big(H_{i-1}^{\text{default}} \parallel H_{i-1}^{\text{default}}\big)$, і використовувати їх там, де у вузла немає явно збереженого значення. Таких хешів буде рівно N.

3.4.2. Інкрементальне дерево Меркла

Інкрементальне дерево Меркла – це варіант реалізації дерева Меркла, що використовується для застосунків, де важливо тільки підтримувати актуальний рут-хеш дерева, а також підтримувати операцію поступового додавання елементів.

Через те, що немає вимоги зберігати усі вузли, а також через те, що елементи додаються поступово (тобто, спочатку – у перший лист, наступний – у другий лист, і т.д.), ми можемо сильно зменшити необхідну пам'ять для підтримки такого дерева – від $O(N \cdot K)$ до $O(N)$.

Адже перманентна пам'ять (*storage*) на EVM-сумісних блокчейнах дорога, це сильно зменшує ціну підтримки такого дерева на смарт-контракті.

Алгоритм додавання елементу наступний:

- Будемо підтримувати масив H^{filled}, де будемо зберігати останні оновлені вузли на кожному рівні дерева, але тільки якщо вони є лівими піддеревами батьківського вузла.
- На кожному рівні дерева, починаючи з листів, будемо обчислювати новий хеш батьківського вузла:
  - якщо новий вузол – лівий, то хеш дорівнює $H_{i+1}^{\text{new}} = H\left(H_i^{\text{new}} \parallel H_i^{\text{default}}\right)$
  - якщо новий вузол – правий, то хеш дорівнює $H_{i+1}^{\text{new}} = H\left(H_i^{\text{filled}} \parallel H_i^{\text{new}}\right)$

- якщо новий вузол – лівий, також оновлюємо $H_i^{\text{filled}} := H_i^{\text{new}}$

3.5. Протокол Aave

Aave [14] – це децентралізований протокол позик, розгорнутий на блокчейні Ethereum. Aave є скороченням від слова «привид» на фінській мові, що символізує прозорість в криптоекосистемі.

Aave має відкритий вихідний код та є некастодіальним, тобто не має активів під керуванням фізичних чи юридичних осіб. Протокол дозволяє користувачам здійснювати операції позики, кредитування та вкладення криптовалюти над близько 20 видами ERC20-токенів без необхідності підтвердження особистості або взаємодії з фінансовими інститутами.

Особливості Aave:

1. **Вклади та позики.** Користувачі можуть внести свої криптовалютні активи в пули ліквідності Aave, заробляючи відсотки доходу. Вони також можуть позичати активи, платячи відсотки за позику. Це дає можливість надавати надійне P2P кредитування без посередників. Відсотки комісії, що користувачі платять за позики, розподіляються поміж користувачами-вкладчиками.
2. **Flash Loans** (Миттєві позики). Це одна з унікальних особливостей Aave. Flash Loans дозволяють користувачам позичати будь-яку кількість активів без забезпечення, за умови, що позика

відшкодовується в одному і тому ж блоку Ethereum. Це відкриває широкий спектр можливостей для арбітражу, перебудови портфоліо та ін.

3. **aTokens.** Коли користувачі вносять активи в Aave, вони отримують відповідні aTokens. aTokens автоматично накопичують відсотки в реальному часі, що дозволяє користувачам бачити свої доходи на постійній основі.
4. **Стабільні та нестабільні ставки.** Aave дозволяє користувачам вибрати між стабільними та нестабільними ставками при позиці активів.

Aave пропонує багато можливостей як для користувачів, так і розробників екосистеми Ethereum, адже Aave досить легко інтегрувати у інші протоколи – як, наприклад, зробили Yearn та Aztec.

3.5.1. Ризики

Як і всі інші DeFi протоколи, включаючи той, що описаний в даній роботі, Aave має кілька потенційних ризиків, які користувачам варто враховувати. Ось декілька з них:

1. **Ризик смарт-контрактів.** Якщо в коді смарт-контракту Aave або в будь-якому з його контрактів-залежностей є помилки, це може призвести до втрати коштів. Це може бути результатом помилок у

коді або зловмисного взлому. Aave проводила аудити свого коду, але це не гарантує повної відсутності помилок.

2. **Ризик ліквідації.** Це ризик, що активи, які ви залишили в якості застави, можуть значно впасти в ціні. Якщо це станеться, ваша позика може стати небезпечно великою в порівнянні з заставою, і ваші заставні активи можуть бути ліквідовані, щоб покрити позику. Варто зазначити, що цей ризик присутній тільки для користувачів, що позичали активи, а не для вкладчиків.
3. **Регуляторний ризик.** Уряди по всьому світу продовжують розвивати своє регулювання криптовалют, і будь-які майбутні зміни в законодавстві можуть вплинути на роботу Aave або на можливість користувачів використовувати Aave.
4. **Ризик оракулів.** Для відслідковування відношень цін токенів в заставі та у позиці, Aave використовує т. зв. оракули – смарт-контракти, що отримують дані із зовнішнього світу. Від цін, що вони надають, залежить, чи будуть здійснюватись ліквідації. Отже, якщо якимось чином дані оракулів будуть недоступні чи невірні, можуть бути хибно здійснені (або не здійснені) ліквідації позицій користувачів.

3.5.2. Flash Loans

Миттєва позика (*flash loan*) є інноваційною функцією, що була вперше запроваджена протоколом Aave в DeFi просторі. Це тип

позики, що дозволяє користувачам позичити будь-яку доступну кількість активів без застави, проте позика повинна бути повернута в рамках однієї транзакції Ethereum.

Це працює наступним чином:

1. Користувач бере Flash Loan.
2. Користувач використовує ці кошти для виконання певних операцій, як-то арбітраж, переконфігурація портфоліо, само-погашення позики тощо.
3. Користувач повертає позику з невеликою платою за користування в межах тієї ж транзакції.

Якщо позика не повертається, або повертається не повністю, транзакція відкидається і стан блокчейна повертається до стану перед транзакцією (*revert*), адже будь-яка транзакція в Ethereum є атомарною (тобто, вона або виконується повністю, або не виконується взагалі).

Цей механізм дозволяє користувачам використовувати значні капіталовкладення для короткострокових операцій без необхідності великих інвестицій, а також знижує ризики пов’язані з ціною активів.

3.5.3. Інтеграція

Факт того, що кошти користувачів можуть досить довго знаходитись на контракті протоколу, це спонукає до думки про

використання цих коштів для генерації додаткових доходів. Ці доходи потім можуть бути використані кількома способами, такими як:

- Оплата послуг релейерів
- Фандинг підтримки та покращення протоколу розробниками
- Створення фонду протоколу та керування ним через DAO
- Розподілення доходів поміж користувачами протоколу
- Донати на ЗСУ

Варто зазначити, що розподілення доходів неможливо здійснювати пропорційно до часу, що вкладчики тримали кошти на протоколі, адже підчас виводу коштів неможливо зрозуміти, коли (і ким) було здійснене початкове вкладення.

Генерувати доходи для протоколу можемо двома способами:

1. Вкладення в Aave

Кожен депозит ETH у протокол будемо автоматично вкладати в Aave, тобто конвертувати в aWETH. Таким чином, ми збільшуємо ризик через взаємодію зі складною системою смарт-контрактів та оракулів, проте отримуємо близько 2% річних доходів за наші вкладення.

При виводі з протоколу, ми аналогічним чином закриватимемо нашу позицію на Aave, таким чином конвертуючи aWETH у ETH, який ми і повертатимемо користувачу. Залишок конвертації (тобто отримані на вкладенні відсотки) залишимо на контракті для майбутнього використання.

2. Flash Loans

Так як дуже імовірно, що на протоколі у будь-який момент часу буде значна кількість коштів, має сенс реалізувати там функцію `flashLoan`, подібну до тої, що є у Aave. Вона даватиме змогу користувачам миттєво позичити весь баланс контракту протоколу без застави, за умови що позика буде погашена у межах тієї самої транзакції. За кожну таку позику братитемо комісію 0.3%, що і буде джелелом доходу.

3.5.4. Розрахунок потенційних прибутків

Проаналізувавши події (*events*) `Deposit` та `Withdrawal` на смарт-контракті Tornado Cash, а також події `FlashLoan` на смарт-контракті Aave за допомогою Etherscan API, можемо просимулювати роботу протоколу з наведеними вище інтеграціями та розрахувати потенційний прибуток від них.

Прибутки від Aave будуть відповідати наступній формулі:

$$P(t_0) = I \cdot \int_0^{t_0} b(t)\ \mathrm{dt}$$

де

$P(t_0)$ – прибутки на момент часу t_0

I – процентна ставка за депозитом у Aave

$b(t)$ – баланс контракту протоколу на момент часу t

Нижче наведено графік потенційних прибутків протоколу при річній процентній ставці 2% (Рисунок 2).

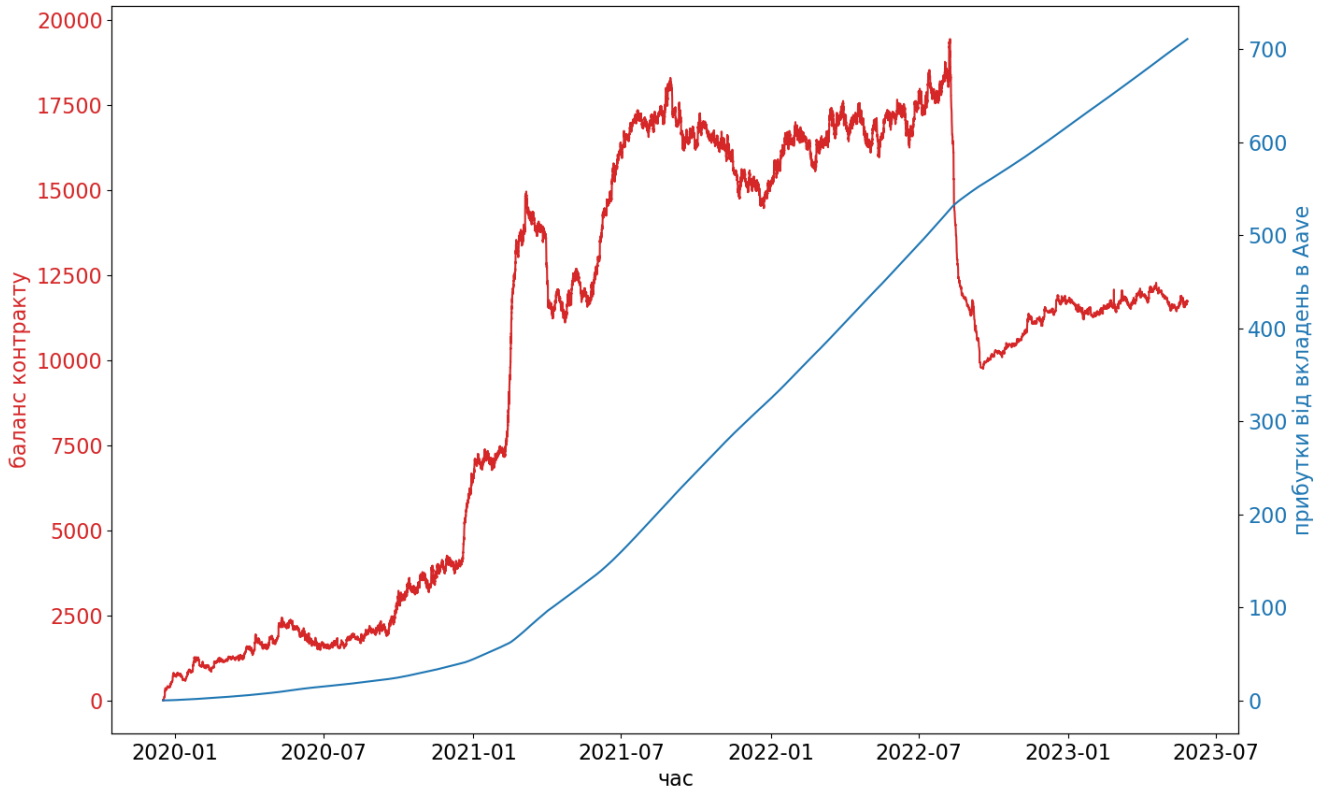

Рисунок 2 – Потенційні прибутки від інтеграції з Aave

Прибуток від реалізації `flashLoan` буде залежати від риночної активності та не може бути напряму розрахований, проте ми можемо розрахувати річний прибуток, отриманий протоколом Aave та екстраполювати на наш протокол. Прибуток становить 212 ETH за минулий рік.

Весь вихідний код протоколу, тести та скрипти, використані для симуляцій опубліковані на GitHub [15] під ліцензією MIT.

ВИСНОВКИ

У даній роботі було детально розглянуто властивості різних протоколів доказів з нульовим розголошенням, наведені приклади сучасних протоколів та складено їх порівняльну характеристику. Також, було проведено огляд різних протоколів для анонімних переказів та описано технології, що вони використовують, їх переваги та недоліки.

Основну мету – покращення реалізації існуючого протоколу для анонімних переказів на основі доказів з нульовим розголошенням – виконано. Протокол реалізовано на сучасній маловідомій мові програмування та з використанням сучасної пруф-системи, що збільшило їх адопцію та буде прикладом їх використання для майбутніх розробників. Реалізовано покращення, що включають інтеграцію протоколу Aave та функцію миттєвих позик. Оцінено їхню ефективність – розраховані потенційні прибутки для протоколу після впровадження даних вдосконалень.

Весь вихідний код протоколу, тести та скрипти, використані для симуляцій опубліковані на GitHub під ліцензією MIT. Протокол впроваджено на тестову мережу Goerli для публічного тестування.

Дослідження, проведені у роботі, можуть бути використані іншими розробниками протоколів, фін-тех компаніями та фінансовими установами. Сам протокол може використовуватись

користувачами, що бажають зберігти ефективність капіталу під час анонімних переказів.

Подальші дослідження за цією темою можуть розглядати інтеграцію інших DeFi інструментів у протоколи анонімних переказів, методи зменшення ризиків та оптимізацію обчислень, виконуваних на смарт-контрактах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. C. P. Schnorr, “Efficient signature generation by smart cards,” *J. Cryptology*, no. 4, 1991. [Електронний ресурс]. Режим доступу до ресурсу: https://link.springer.com/content/pdf/10.1007/BF00196725.pdf
2. A. Fiat, and A. Shamir. (1986). How to prove yourself: Practical solutions to identification and signature problems. Presented at Advances Cryptology -- CRYPTO '86. [Електронний ресурс]. Режим доступу до ресурсу: https://link.springer.com/content/pdf/10.1007/3-540-47721-7_12.pdf
3. E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, “Succinct non-interactive zero knowledge for a von neumann architecture,” Cryptology ePrint Archive, 2019. [Електронний ресурс]. Режим доступу до ресурсу: https://eprint.iacr.org/2013/879.pdf
4. E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, “Scalable, transparent, and post-quantum secure computational integrity,” Cryptology ePrint Archive, 2018. [Електронний ресурс]. Режим доступу до ресурсу: https://eprint.iacr.org/2018/046.pdf
5. B. Bünz, J. Bootle, et al., “Bulletproofs: Short proofs for confidential transactions and more,” Cryptology ePrint Archive, 2017. [Електронний ресурс]. Режим доступу до ресурсу: https://eprint.iacr.org/2017/1066.pdf

6. D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, “Zcash protocol specification,” 2022. [Електронний ресурс]. Режим доступу до ресурсу: https://zips.z.cash/protocol/protocol.pdf
7. K. M. Alonso, and J. H. Joancomarti, “Monero: Privacy in the blockchain,” Cryptology ePrint Archive, 2018. [Електронний ресурс]. Режим доступу до ресурсу: https://eprint.iacr.org/2018/535.pdf
8. G. Yu, “Mimblewimble non-interactive transaction scheme,” 2020. [Електронний ресурс]. Режим доступу до ресурсу: https://eprint.iacr.org/2020/1064.pdf
9. Z. J. Williamson, “The AZTEC protocol,” 2018. [Електронний ресурс]. Режим доступу до ресурсу: https://github.com/AztecProtocol/aztec-v1/blob/master/AZTEC.pdf
10. A. Pertsev, R. Semenov, and R. Storm, “Tornado cash privacy solution v1.4,” 2019. [Електронний ресурс]. Режим доступу до ресурсу: https://berkeley-defi.github.io/assets/material/Tornado%20Cash%20Whitepaper.pdf
11. M. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen, “Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity,” Cryptology ePrint Archive, 2016. [Електронний ресурс]. Режим доступу до ресурсу: https://eprint.iacr.org/2016/492.pdf
12. A. Gabizon, Z. J. Williamson, and O. Ciobotaru, “Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of

knowledge,” Cryptology ePrint Archive, 2022. [Електронний ресурс]. Режим доступу до ресурсу: https://eprint.iacr.org/2019/953.pdf

13. R. Dahlberg, T. Pulls, and R. Peeters, “Efficient sparse merkle trees,” Cryptology ePrint Archive, 2016. [Електронний ресурс]. Режим доступу до ресурсу: https://eprint.iacr.org/2016/683.pdf
14. E. Boado, and S. Kulechov, “AAVE protocol whitepaper v2.0,” 2020. [Електронний ресурс]. Режим доступу до ресурсу: https://github.com/aave/protocol-v2/blob/master/aave-v2-whitepaper.pdf
15. Л. Потьомкін, “Реалізація протоколу анонімних переказів на основі доказів з нульовим розголошенням,” 2023. [Електронний ресурс]. Режим доступу до ресурсу: https://github.com/ly0va/thesis