Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Факультет комп’ютерних наук
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

«Допущено до захисту»
В.о. завідувач кафедри БІСТ
Мелкозьорова О.М.
_____
« » 2024 р.

Пояснювальна записка
до кваліфікаційної роботи бакалавра
на тему: «Дослідження та розробка алгоритму верифікації інформації з використанням протоколу Доведення з нульовим розголошенням»

оцінка « » Керівник : ст. викл. Шеханін К.Ю.
Голова ЕК Рецензент: ст. викл. Лисицький К.Є.
Лемешко О.В. _____

Виконавець : студент групи КБ-42
Пецоляк М.М.

Харків – 2024

РЕФЕРАТ

Робота складається з 45 сторінок, 1 таблиці, 8 рисунків, 6 лістингів, 2 додатків та 9 джерел посилання.

Робота присвячена формуванню алгоритму верифікації інформації з використанням протоколів ZKP. В якості рішення проблеми було запропоновано 3 алгоритми верифікації на мові програмування Rust, з використанням бібліотеки Halo2, яку використовували в ZCash.

Метою роботи є аналіз сучасної архітектури існуючих ZKP-рішень, спрощення та прискорення роботи з бібліотеками ZKP-рішень шляхом розробки алгоритмів з їх використанням.

В результаті роботи були сформовані доцільні алгоритми ZKP-рішень для розглянутих проблем на основі бібліотеки Halo2, розглянута архітектура і завдання сучасних продуктів, які використовують ZKP в своїх системах, сформовані слабкі ти сильні сторони ZKP-алгоритмів на прикладах.

Практичною цінністю розроблених алгоритмів є дослідження та використання ZKP для формування схем доведення знань будь-якого роду без їх розкриття. Отримані результати допоможуть зробити роботу з бібліотекою Halo2 більш зручною та швидкою.

Ключові слова: ZK-SNARK, ZK-STARK, ВЕРИФІКАЦІЯ ДОВЕДЕННЯ, АЛГОРИТМ ZKP, БЛОКЧЕЙН, БЕЗПЕКА ІНФОРМАЦІЇ, КІБЕРБЕЗПЕКА

ABSTRACT

The work consists of 45 pages, 1 table, 8 figures, 6 listings, 2 appendices and 9 reference sources.

The work is devoted to the formation of an information verification algorithm using ZKP protocols. As a solution to the problem, 3 verification algorithms were proposed in the Rust programming language, using the Halo2 library, which was used in ZCash.

The purpose of the work is to analyze the modern architecture of existing ZKP-solutions, consider and develop information verification algorithms using the Halo2 library, further simplify and speed up work with ZKP-solution libraries.

As a result of the work, appropriate ZKP-solution algorithms were formed for the considered problems based on the Halo2 library, the architecture and tasks of modern products that use ZKP in their systems were considered, the weaknesses and strengths of ZKP-algorithms were formed using examples.

The practical value of the developed algorithms is the study and use of ZKP for the formation of knowledge proof schemes of any kind without their disclosure. The obtained results will help to make working with the Halo2 library more convenient and faster.

Key words: ZK-SNARK, ZK-STARK, VERIFICATION OF PROOF, ALGORITHM ZKP, BLOCKCHAIN, INFORMATION SECURITY, CYBERSECURITY

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 5
ВСТУП 6
1 АКТУАЛЬНИЙ СТАН СУЧАСНОГО АЛГОРИТМУ ZKP І ДОСВІД ЙОГО ВИКОРИСТАННЯ 8
1.1 Опис алгоритму ZKP 8
1.1.1 Загальна характеристика ZKP 8
1.1.2 Приклади реалізації та властивості ZKP 9
1.2 Технології на основі використання ZKP 15
1.2.1 zk-SNARK 18
1.2.2 zk-STARK 19
1.2.3 BulletProofs та порівняння протоколів 20
1.3 Досвід використання ZKP 21
1.3.1 Області використання ZKP 21
1.3.2 Сучасні моделі, які використовують ZKP 22
1.4 Висновки до першого розділу 24
2 РОЗРОБКА АЛГОРИТМУ ВЕРИФІКАЦІЇ З ZKP 27
2.1 Алгоритм верифікації на основі константи 27
2.2 Алгоритм верифікації на основі послідовностей 30
2.3 Алгоритм верифікації на основі ігор 32
2.4 Висновки до другого розділу 35
ВИСНОВКИ 37
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 39
ДОДАТОК А. ПРИКЛАД СФОРМОВАНОГО ДОВЕДЕННЯ 41
ДОДАТОК Б. АРХІТЕКТУРА ЧИПУ «КАМІНЬ – НОЖИЦІ – ПАПІР» 42

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

BPP	bounded-error, probabilistic, polynomial – обмежена помилка, ймовірнісний, поліноміальний
IoT	Internet of Things – інтернет речей
iZKP	interactive Zero-Knowledge Proof – інтерактивне доведення з нульовим розголошенням
niZKP	non-interactive Zero-Knowledge Proof – не інтерактивне доведення з нульовим розголошенням
ZKP	Zero-Knowledge Proof – доведення з нульовим розголошенням
zk-SNARK	Zero-Knowledge Succinct Non-Interactive Argument of Knowledge – короткий не інтерактивний аргумент знання з нульовим розголошенням
zk-STARK	Zero-Knowledge Scalable Transparent Argument of Knowledge – масштабований прозорий аргумент знання з нульовим розголошенням

ВСТУП

Питанню безпеки і конфіденційної роботи з інформацією в сучасному світі виділяється величезне фінансування і витрачається багато сил заради покращення все існуючих систем. Однак, крім покращення алгоритмів конфіденційної передачі інформації і зменшувати вірогідність перехоплення її порушником, важливе і доведення права власності на інформацію певною особою.

Головна задача алгоритмів, які застовують Доведення з нульовим розголошенням (ZKP) – розрахунок вірогідності наявності інформації у особи без необхідності розкриття цієї інформації. Напрямків використання таких алгоритмів – безліч, але в основному він використовується в блокчейн-системах.

Chain зазначає, що використання ZKP в блокчейнах може привести до покращення приватності і масштабованості системи. Завдяки схемі роботи ZKP, яка генерує доведення на основі вкладеної в нього інформації, декілька транзакцій можна зберігати у вигляді одного доведення, який у свою чергу зберігається в блокчейні. Разом з цим переваги ZKP відмічаються і Mangrovia, Chainlink та OpenZeppelin – пропонуючи свої рішення з використанням алгоритму такого доведення.

Використання ZKP доведення не закінчується тільки на блокчейн. Сфера використання таких алгоритмів стосується також і децентралізованого доведення особи людини, систем голосування, використання в інтернеті речей (IoT), міжнародних ланцюгів поставок тощо, зазначає Chainlink.

Актуальність роботи. Тема роботи стосується безпеки використання і передачі даних та розглядає сучасні підходи до рішення проблем приватності. ZKP-алгоритми надають можливість особам доводити

наявність певної інформації без її витоку, можуть спростити алгоритм верифікації особи людини, збільшити приватність і масштабованість сучасних блокчейн рішень.

Об’єкт дослідження. Об’єктом дослідження є ZKP-алгоритми.

Предмет дослідження. Предметом дослідження є алгоритм верифікації інформації з використанням ZKP.

Завдання дослідження. Метою роботи є розробка алгоритму верифікації інформації за допомогою ZKP-алгоритмів. Такий алгоритм має підтримувати створення доведення, перевіряти їх, виводити результат перевірки. Відповідно до цього сформовані наступні завдання:

1. Розглянути і описати концепцію ZKP;
2. Дослідити сучасні рішення, які використовують ZKP;
3. Провести порівняльний аналіз існуючих рішень;
4. Розробити приклади алгоритмів створення і перевірки доведення з використанням ZKP;
5. Визначити актуальність і доцільність алгоритмів;
6. Зробити висновки.

1 АКТУАЛЬНИЙ СТАН СУЧАСНОГО АЛГОРИТМУ ZKP І ДОСВІД ЙОГО ВИКОРИСТАННЯ

1.1 Опис алгоритму ZKP

1.1.1 Загальна характеристика ZKP

У криптографії підтвердження з нульовим знанням або протокол з нульовим знанням – метод, за допомогою якого одна сторона (довідник) може довести іншій стороні (верифікатору), що дане твердження є істинним, уникаючи при цьому передачі верифікатору будь-якої інформації, окрім простий факт істинності твердження. Інтуїція, яка лежить в основі доказів з нульовим знанням, полягає в тому, що тривіально довести володіння певною інформацією, просто розкривши її; Завдання полягає в тому, щоб довести це володіння, не розкриваючи інформацію чи будь-який її аспект.

У світлі того факту, що можна створити доказ певного твердження лише тоді, коли він володіє певною секретною інформацією, пов’язаною з твердженням, верифікатор, навіть переконавшись у правдивості твердження, все одно повинен залишатися нездатним довести заява третім особам.

Варто зазначити, що концепції ZKP з’являються ще під час вирішення задачі Візантійський генералів. Задача вирішується моделлю (P, V, S), в якій: P – той, що доводить; V – той, що перевіряє; S – твердження, яке виноситься на доведення. При вирішенні задачі система відповідала трьом умовам[1]:

- Повноті - якщо S справді істинне, то абонент P майже переконає абонента V визнати його істинним;

- Коректності – якщо S справді істинне, то абонент P майже переконає абонента V визнати його істинним;
- Стійкості – у результаті роботи протоколу (P, V, S) абонент V не збільшує свої знання про твердження S або, іншими словами, не зможе здобути ніякої інформації про те, чи S істинне.

У простій моделі нетривіальні докази з нульовим знанням (тобто для мов, що не належать до BPP) вимагають взаємодії між довідником і верифікатором. Ця взаємодія зазвичай передбачає вибір одного або кількох випадкових викликів верифікатором; випадкове походження цих викликів разом із успішними відповідями перевіряючого на них, незважаючи на це, разом переконує верифікатор у тому, що перевіряльник справді володіє заявленими знаннями. Якщо взаємодії не було, то верифікатор, отримавши стенограму виконання протоколу, тобто єдине повідомлення довідника, міг би відтворити цю стенограму третій стороні, таким чином переконавши третю сторону, що верифікатор також володіє секретною інформацією.

У моделях звичайного випадкового рядка та випадкового оракула існують не інтерактивні докази з нульовим знанням у світлі евристики Фіата-Шаміра. На практиці ці докази ґрунтуються на обчислювальних припущеннях (як правило, стійкість до зіткнень криптографічної хеш-функції).

1.1.2 Приклади реалізації та властивості ZKP

Одним з прикладів реалізації ZKP в абстрактних ситуаціях є давня задача «Печера Алі-Баби», яка ілюструє процес верифікації доведення. Історія була вперше опублікована в 1990 році Жаном-Жаком Квісквотером та іншими в їхній статті «Як пояснити своїм дітям протоколи з нульовим знанням».

Формулювання задачі наступне: існує певна печера та дві особи (особа А та особа Б), в печері знаходяться певні двері, які відкриваються тільки за секретну фразу, яку знає особа А. Яким чином можна довести особі Б, що особа А має доступ до цієї фрази, якщо не можна її казати? Задача має наступне рішення:

- Особи А та Б позначають ліворуч і праворуч шляхи від входу як вихід А і Б. Особа Б чекає біля печери, поки особа А заходить туди. Особа А йде шляхом виходу А або Б. Особі Б не дозволено побачити, яким шляхом вона йде. На рисунку 1.1 показана ілюстрація цього етапу;

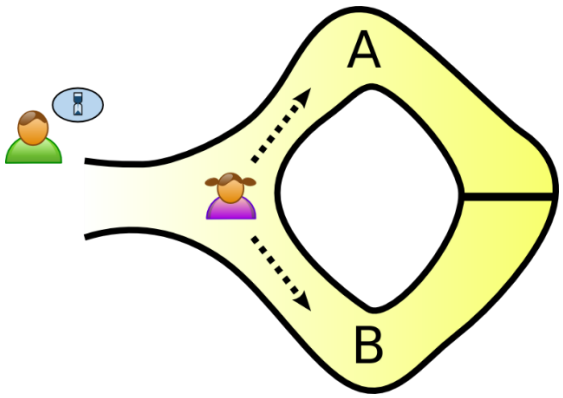

Рисунок 1.1 – Перший етап задачі про печеру.

- Після цього особа Б заходить у печеру та вигукує назву шляху, яким вона хоче повернутися, А або Б, вибраних навмання. Якщо особа дійсно знає слово, це легко: вона відкриває двері, якщо потрібно, і повертається по бажаному шляху. На рисунку 1.2 показана ілюстрація цього етапу;

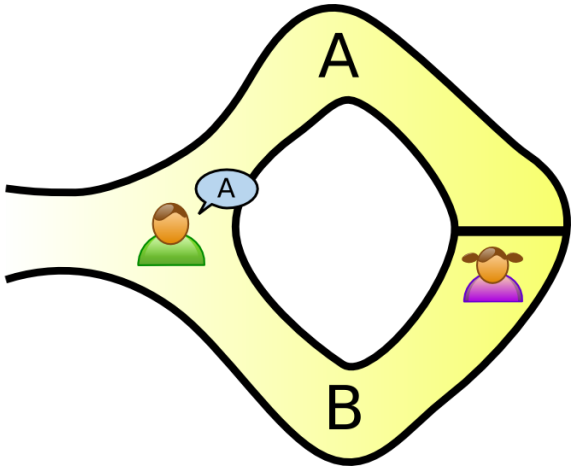

Рисунок 1.2 – Другий етап задачі про печеру.

- Припустимо, що особа А не знає цього слова. Тоді вона зможе повернутися вказаною дорогою, лише якщо особа Б назве ту саму дорогу, якою вона увійшла. Оскільки особа Б навмання вибирала вихід А або Б, у особи А був би 50% шанс правильно вгадати. На рисунку 1.3 наведена ілюстрація цього етапу;

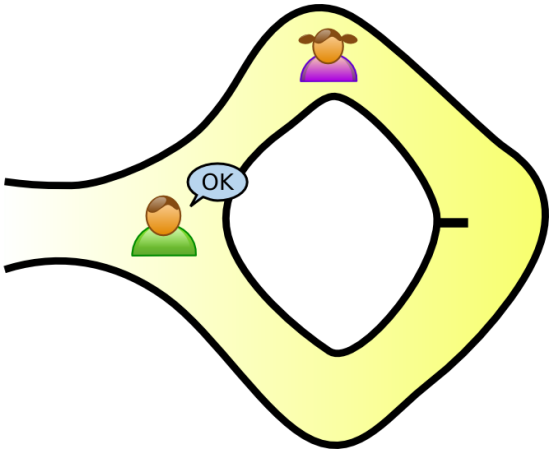

Рисунок 1.3 – третій етап задачі про печеру.

- Якщо виконувати одну й ту саму перевірку багато разів до вірогідність зменшується в геометричній прогресії. Наприклад вірогідність вгадати вибір особи Б 20 разів підряд сягає все менше ніж 1 до 1 000 000.

Таким чином, якщо особа А неодноразово з'являється на виході на яку вказує особа Б, то дуже ймовірно, що особа А справді знає секретне слово. Таким чином можна представити спосіб верифікації ZKP.

Варто зазначити деякі властивості такого доведення:

- Непотреба в сторонніх спостерігачах. Якщо особа Б носить приховану камеру, яка записує всю транзакцію, єдине, що камера зафіксує, це в одному випадку, коли особа Б обирає вихід, з якого потім вийде особа А. Такого роду запис не є доказом наявності у особи А секретної фрази, оскільки немає гарантії підробки відеозапису.
- Контроль знань. Один з важливих моментів є те, що таких метод доведення наявності секретної фрази є валідним тільки для осіб А та Б, будь-які інші особи, які можуть спостерігати за експериментом не можуть гарантувати знання особою А такої фрази, знову ж з причини можливої підробки експерименту. Така властивість ще називається нульовим знанням.

Варто зазначити, що якщо особа Б буде обирати вихід кидаючи монету на камеру, то цей протокол втрачає властивість нульового знання; підкидання монети на камері, ймовірно, було б переконливим для будь-якої людини, яка дивилася б запис пізніше. Таким чином, хоча це не розкриває таємницю слова особі Б, це дає можливість переконати світ загалом, що особа А має ці знання, що суперечить заявленим бажанням і приватності.

Інший варіант розглядання реалізації протоколу ZKP – задача двох кульок. Уявіть що ваш друг є червоно-зеленим дальтоніком, і у вас є дві кулі: одна червона та одна зелена, але в іншому однакові. Другу кульки здаються абсолютно однаковими. Він скептично ставиться до того, що кулі насправді можна розрізнити.

Задача включає в себе необхідність доведення другу, що кульки насправді різного кольору, але нічого іншого. При цьому розкривати яка кулька червона, а яка зелена – не можна.

Схема рішення такої задачі за рахунок ZKP-протоколу проста:

- Ви віддаєте два м’ячі другу, і він кладе їх собі за спину. Далі він бере один із м’ячів, дістає його з-за спини та демонструє. Потім він знову кладе його за спину, а потім вирішує відкрити лише одну з двох куль, вибираючи одну з двох навмання з рівною ймовірністю;
- Наступним кроком буде визначити, чи змінилась куля в руках друга – чи ні. Оскільки за умовою задачі головний герой не являється дальтоніком, то можна легко визначити факт заміни кулі на іншу. При цьому в очах друга – кулі абсолютно однакові, але він знає однозначно змінились вони, чи ні;
- Останнім кроком вирішення завдання буде просте повторення дій, скажімо 20 разів. Тоді вірогідність того, що вибір кульок був випадковий – дуже мала, а факт того, що кульки різного кольору – найбільш вірогідний.

Наведений вище доказ не має знань, оскільки друг ніколи не дізнається, яка кулька зелена, а яка червона. Під час виконання задачі ідентифікація кульок відбувалася лише за кольором без назви їх кольорів, що робить неможливим визначення справжнього кольору кульок для вашого друга.

Ще один важливий момент задачі – конкретизація доведення. Друга можна було надурити, сказавши що кульки мають чорний та білий колір, хоча насправді вони червоні на зелені. Тому варто зазначати доведення з додатковими кроками верифікації, або більшою конкретизацією. Наприклад, цієї проблеми можна було б позбавитися, якщо сказати не «Я хочу довести що одна кулька червона, а інша – зелена», а «Я хочу довести, що ці дві кульки мають різний колір».

Мінуси та обмеження ZKP. Вище розглянуті плюсі та переваги систем, побудованих на основі ZKP крім принципів нульового розголошення та секретності мають також і свої минуси. Загалом вони представляють собою результат використання правил системи на основі ZKP:

- Великі обчислювальні витрати;
- Необхідність встановлення в довірчому середовищі;
- Складність і подальший розвиток;
- Взаємодія між системами ZKP та стандартизація.

Розглянемо кожну з проблем більш детально. Великі обчислювальні витрати. Деякі протоколи ZKP можуть бути дорогими в обчислювальному плані та вимагати значної потужності обробки та часу для створення та перевірки доведення. Це може обмежити їх масштабованість і практичність, особливо в середовищах з обмеженими ресурсами.

В якості рішення такої проблеми пропонується виконання досліджень та розробок, які повинні бути спрямовані на підвищення ефективності та продуктивності протоколів ZKP. Це включає в себе оптимізацію алгоритмів, зменшення витрат на обчислення та націлені як на вивчення альтернативних підходів, таких як zk-SNARK і zk-STARK, так і на розробку нових сучасних та більш досконалих підходів.

Необхідність встановлення в довірчому середовищі. Деякі системи ZKP вимагають етапу надійного налаштування, на якому генеруються початкові параметри. Якщо це налаштування зламано або не виконано належним чином, це може підірвати гарантії безпеки та конфіденційності системи ZKP.

Для вирішення цієї проблеми необхідно докласти зусиль в сфері розробці ZKP із надійними налаштуваннями у вигляді гарантованості створення доцільних довірчих середовищ, або формування моделей, яким

довірче середовище не обов’язкове. Прикладом моделі, яка вирішила цю проблему є модель zk-STARK.

Складність та подальший розвиток. ZKP можуть бути концептуально складними, а їх реалізація потребує спеціальних знань і досвіду. Розробка та розгортання систем на основі ZKP може бути складним завданням, а помилки в реалізації можуть призвести до вразливості або небажаних наслідків.

Взаємодія між системами ZKP та стандартизація. Протоколи та системи ZKP все ще розвиваються, а стандартизації бракує. Взаємодія між різними реалізаціями ZKP може бути обмеженою, що може перешкоджати широкому впровадженню та інтеграції з існуючими системами.

В результаті можна побачити, що використання ZKP в рішеннях для своєї системи можуть нести як позитивні, так і негативні наслідки. Сучасні моделі, рішення яких побудовані на основі ZKP представляють з себе великі, складні та багатомодульні системи, які в більшості своєму не можуть взаємодіяти з іншими системами, які базуються на схожих рішеннях через відсутність чіткої стандартизації для таких систем. Натомість своїм мінусам, системи на основі ZKP пропонують набір інструментів, які гарантують приватність для користувачів, захищеність для їх інформації та масштабованість в більшості випадків ZKP-рішень.

1.2 Технології на основі використання ZKP

ZKP можна розділити на дві категорії: інтерактивні та не інтерактивні ZKP[2].

Інтерактивні ZKP (iZKP). iZKP вимагають взаємодії між довідником і верифікатором, щоб переконатися, що підтвердження знань виконується правильно. У iZKP довідник та верифікатор беруть участь у серії раундів, обмінюючись повідомленнями, щоб довести, що довідник володіє відповідними знаннями. На рисунку 1.4 наведена ілюстрація щодо iZKP.

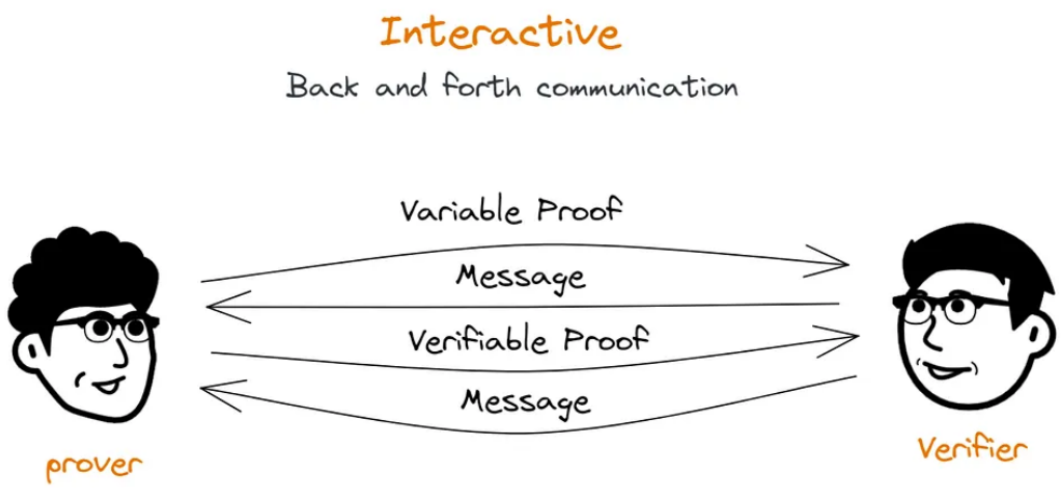

Рисунок 1.4 – Взаємодія в iZKP.

Одним із найбільш часто використовуваних iZKP – евристика Фіата-Шаміра, у якій випадкове значення генерується та використовується як параметр для перевірки. Потім довідник повинен використати свої таємні знання, щоб дати відповідь на виклик, який верифікатор може визначити як правильний. Якщо відповідь правильна, верифікатор приймає підтвердження, і взаємодія завершена.

Однією з ключових переваг iZKP є те, що вони дуже гнучкі та можуть використовуватися для підтвердження знання щодо багатьох різних типів інформації:

- Наприклад, їх можна використовувати, щоб довести, що хтось знає секретне значення, наприклад пароль;
- Ще одним способом використання такого доведення – переконливысть, що особа має певний рівень повноважень над цифровим активом, таким як криптовалюта;
- Такі доведення можна використовувати і в системах автентифікації користувачів для доведення особистості,

доведенням може слугувати яка-небудь ключ-карта, або біометричні дані.

Ще однією перевагою iZKP є те, що вони дуже безпечні. Використання випадкових викликів і вимога взаємодії між довідником і верифікатором ускладнюють для зловмисника скомпрометацію доказу.

Однак iZKP мають і деякі недоліки:

- Вони вимагають високого рівня довіри між довідником і верифікатором. Якщо будь-яка сторона скомпрометована, безпека доказу може бути скомпрометована.
- iZKP можуть займати багато часу та значну кількість обчислювальних ресурсів, що може зробити їх непрактичними для використання в деяких системах.

Неінтерактивні ZKP (niZKP). З іншого боку стоять niZKP – ZKP, які не вимагають взаємодії між довідником і верифікатором. Замість цього засіб перевірки створює доказ, який може бути перевірено верифікатором без будь-якої взаємодії. На рисунку 1.5 наведена ілюстрація щодо iZKP.

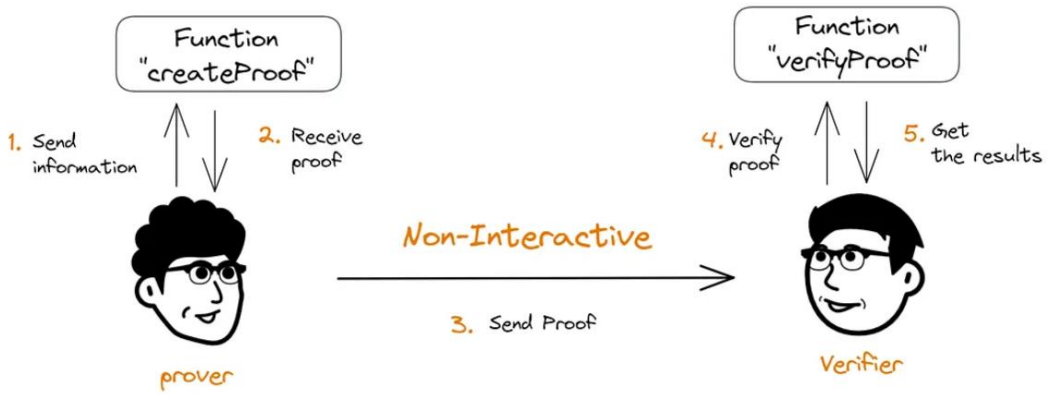

Рисунок 1.5 – Взаємодія в niZKP.

Одним із найпоширеніших niZKP є SNARK[3]. SNARK використовують розширені математичні алгоритми для створення доказу, який може перевірити верифікатор без будь-якої взаємодії. Доказ, як

правило, є коротким, сильно стислим представленням інформації, яку довідник хоче підтвердити, і верифікатор може використовувати це представлення, щоб підтвердити, що доказ дійсний.

Основною перевагою niZKP є те, що вони набагато ефективніші та масштабованіші[4], ніж iZKP. Оскільки немає взаємодії між довідником та верифікатором, niZKPs можна перевірити набагато швидше та з меншими обчислювальними ресурсами. Крім того, оскільки вони не вимагають взаємодії, niZKP добре підходять для використання в децентралізованих системах, таких як мережі блокчейн, де довіру між сторонами встановити важко.

Як і iZKP, niZKP також мають свої недоліки:

- Одним із головних недоліків є те, що вони менш гнучкі, ніж iZKP, і можуть використовуватися лише для підтвердження знання певних типів інформації.
- niZKPs можуть бути важчими для впровадження та потребувати вищого рівня технічної експертизи.

На основі інтерактивних та не інтерактивних ZKP розробили багато протоколів з різними профілями використання. Загально їх розділяють на наступні категорії: SNARK; STARK; BulletProofs[5].

1.2.1 zk-SNARK

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) був презентований в задокументованому форматі Елі Бен-Сассоном, Алессандро К'єзою, Даніелем Генкіним, Ераном Тромером і Мадарсом Вірзою в 2014 році[6].

Робота zk-SNARK розділяється на 3 фази:

- Початкова фаза. Використовує відкритий та закритий ключ для визначення початкових даних для довідника та верифікатора.

- Доведення ключа. На цьому етапі використовується закритий ключ для генерації доведення.
- Підтвердження доведення. На цьому етапі відкритий ключ використовується для підтвердження доведення.

Протоколи, які побудовані на основі zk-SNARK мають наступні властивості:

- Нульове розголошення. Верифікатор не має жодної іншої інформації, окрім заяви про валідність, наданої довідником.
- Швидкі. Згенерований розмір доведення невеликий, його легко та швидко перевіряти.
- Неінтерактивні. Протоколам не потрібна взаємодія між довідником та верифікатором.
- Захищенні. Зловмисник не може обдурити систему, не надавши відповідне доведення на підтримку твердження.

Недоліками таких протоколів є складне встановлення і підтримка системи а також необхідність швидких розрахункових приладів.

1.2.2 zk-STARK

Zero-Knowledge Scalable Transparent Argument of Knowledge (zk-STARK) був презентований в задокументованому форматі Елі Бен-Сассоном, Іддо Бентовим, Іноном Хореші та Майклом Рябзєвим в 2018 році[7].

Такі протоколи пропонують такий рівень прозорості, якого немає у zk-SNARK. Вони не вимагають довіреного налаштування, що робить їх більш прозорими та потенційно більш захищеними від квантових атак. Протоколи на основі zk-STARK мають наступні властивості:

- Нульове розголошення.
- Здатні до масштабування. З використанням zk-STARK, обчислення виконання транзакцій і зберігання даних можуть

здійснюватися поза робочої системи. Після цього збережені дані можуть буть підтверджені з використанням одного STARK доведення для перевірки їх дійсності в ланцюзі.

- Прозорі. Такі протоколи використовують загальнодоступні генератори псевдовипадкових послідовностей для системних параметрів. За рахунок такої архітектури відпадає необхідність в встановленні власних довірчих генераторів.
- Захищенні.

З недоліків можна зазначити великі розміри доказів, більше обчислювальних витрат.

1.2.3 BulletProofs та порівняння протоколів

BulletProofs протоколи – тип Non-Interactive ZKP протоколів, розроблений Бенедиктом Бунцем, Джонатаном Бутлом, Деном Бонехом, Ендрю Поелстрою та іншими в 2017 році. Він використовує певну логіку, що лежить в основі zk-SNARK і конфіденційних транзакцій, щоб покращити продуктивність доказу з точки зору розміру перевірки та часу перевірки.

В таблиці 1.1 представлені порівняльні характеристики вищеописаних протоколів zk-SNARK, zk-STARK, BulletProofs.

Таблиця 1.1 – Порівняльні характеристики протоколів

	zk-SNARK	zk-STARK	BulletProofs
Складність довідника	O(N * log(N))	O(N * log(N)^k)	O(N * log(N))
Складність верифікатора	O(1)	O(log(N)^k)	O(N)
Складність перевірки	O(1)	O(log(N)^k)	O(log(N))
Середній розмір одного доведення	50 MB	45 kB	1.5 kB

Також необхідно зазначити додаткові можливості деяких протоколів:

- zk-SNARK потребує виконання початкової фази довірчою особою або стороною;
- zk-STARK в порівнянні з іншими протоколами має захист від квантових атак;

Отже, кожний з протоколів має свої позитивні і негативні сторони тому вибір необхідно виконувати орієнтуючись на потреби конкретних рішень.

1.3 Досвід використання ZKP

1.3.1 Області використання ZKP

Доказ із нульовим знанням може використовуватися для захисту конфіденційності даних у різних випадках, таких як:

- Блокчейн. Прозорість публічних блокчейнів, таких як Bitcoin та Ethereum, дозволяє публічно перевіряти транзакції. Однак вона також має на увазі низький рівень конфіденційності та може призвести до деанонімізації користувачів. Докази з нульовим знанням можуть підвищити рівень конфіденційності публічних блокчейнів. Наприклад, криптовалюта Zcash заснована на Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (Zk-SNAKR), криптографічному методі нульового знання;
- Фінанси. ING використовує ZKP, що дозволяє клієнтам довести, що їхня секретна кількість лежить у відомому діапазоні. Наприклад, претендент отримання іпотечного кредиту може довести, що його дохід перебуває у допустимих межах, не розкриваючи точної зарплати.

- Онлайн-голосування. ZKP дозволяє виборцям голосувати анонімно та перевіряти, що їхній голос був включений у остаточний підрахунок голосів.
- Аутентифікація. ZKP може використовуватися для аутентифікації користувачів без обміну секретною інформацією, наприклад, паролями.
- Машинне навчання. ZKP дозволяє власнику алгоритму машинного навчання переконувати інших у результатах роботи моделі, не розкриваючи жодної інформації про саму модель ML.

Разом з використанням ZKP в вищевказаних областях випливають і деякі проблеми, які постають в результаті використання такої моделі: відсутність гарантії правдивості твердження і інтенсивність обчислювання.

Навіть якщо ймовірність перевірки верифікатором того, що доводить бреше, може бути суттєво мала, ZKP не гарантує стовідсоткової достовірності затвердження. Як було показано вище, ймовірність того, що перевіряючий бреше, зменшується на кожній ітерації процесу підбору куль, але ніколи не може досягти нуля. Отже, докази з нульовим знанням є справжніми доказами в математичному сенсі.

Алгоритми, що використовуються в протоколах ZKP, є обчислювально інтенсивними, оскільки вимагають великої кількості взаємодій між перевіряючим і доведеним (в інтерактивних ZKP) або великих обчислювальних можливостей (у неінтерактивних ZKP). Це робить ZKP непридатними для використання на повільних чи мобільних пристроях.

1.3.2 Сучасні моделі, які використовують ZKP

Протоколи ZKP в сучасному світі використовуються в різних сферах технологій і бізнесу, особливо в області криптовалют, кібербезпеки та блокчейн-технологій. Розглянемо декілька прикладів використання протоколів на основі реальних працюючих систем:

- Zcash. Це криптовалюта, яка використовує протоколи ZKP для забезпечення конфіденційності транзакцій. Користувачі можуть надсилати кошти один одному, при цьому залишаючи мінімальні сліди про транзакції, що здійснюються. На рисунку 1.5 показана порівняльна характеристика транзакцій в Bitcoin та Zcash.

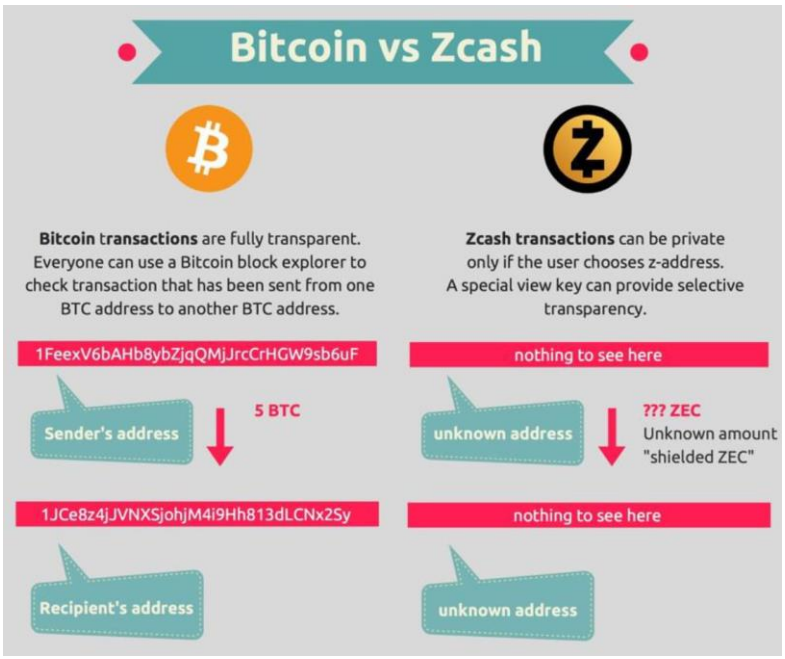

Рисунок 1.5 – Порівняння приватності транзакцій Bitcoin та Zcash.

- StarkWare. Ця компанія розробляє протоколи та рішення для розширення масштабу блокчейнів. Вони використовують протоколи ZKP для підтвердження правильності обчислень у блокчейні без розкриття даних.
- AZTEC Protocol. Це технологічна компанія, яка використовує протоколи ZKP для розробки приватних фінансових інструментів на базі блокчейну. Вони забезпечують конфіденційність транзакцій у мережах Ethereum та інших блокчейнів.
- StarkEx. Це рішення для масштабування торгівлі на блокчейні, яке використовує протоколи ZKP для верифікації правильності операцій та підтвердження даних без необхідності їх розкриття.

- Coda Protocol. Цей стартап використовує протоколи ZKP для створення блокчейну з мінімальним обсягом даних, необхідних для синхронізації мережі. Це дозволяє покращити швидкодію і масштабованість блокчейну.

Загалом, використанням ZKP в системах для підвищення приватності та захищенності інформації користувачів. Такі рішення надають можливості створення захищених каналів для передачі інформації, гарантують знання певної інформації особою в якості надання необхідного доведення, захищають інформацію від несанкціонованого доступу інших осіб, які не мають на неї право та не знають про неї. Рішення з використанням ZKP прибирають необхідність в автентифікації користувачів в системі перед проведенням вищеописаних операцій за рахунок існування відповідного доведення. Такі системи активно використовуються в блокчейні та можуть використовуватися в багатьох інших сферах, пов’язаних з захищеністю та збереженню конфіденційної інформації будь-якого роду.

1.4 Висновки до першого розділу

Отже, сучасні протоколи ZKP походять від задачі Візантійських генералів, яка започаткувала правило стійкості (нульового розголошення). Основною ідеєю таких протоколів є доведення іншому користувачу істинність певного твердження без передачі будь-якої іншої інформації. Описати характеристику та завдання протоколів ZKP можна за допомогою задач:

- Задачу про дві особи та двері з ключем;
- Задачу про двох друзів та кольорові кулі.

В результаті рішення таких задач формуються принципи, на яких побудовані сучасні системи на ZKP-рішеннях. В усіх них реалізовані

принципи нульового знання, вибіркового доведення та секретності інформації.

Така архітектура моделі надає широкий спектр можливостей і механізмів дій, які реалізовані у вигляді типізації архітектури ZKP: інтерактивні та не інтерактивні. Головною відмінністю цих архітектур в методі створення та верифікації доведення. Якщо в випадку з інтерактивними ZKP, довідник та верифікатор мають пряму взаємодію (довідник генерує та відправляє параметри на перевірку – отримує повідомлення від верифікатора – надає верифікатору доведення – отримує результат від верифікатор), то в неінтерактивних ZKP довідник надає свої значення до спеціальної функції генерації доведення, яке після цього відправляє верифікатору, який у свою чергу перевіряє згенероване доведення.

Існуючим прикладом реалізації архітектури ZKP є zk-SNARK та zk-STARK, які надають інструменти доведення інформації з дотриманням принципу нульового знання та мають свої плюси та мінуси. Головною різницею цих алгоритмів є те, що zk-SNARK потребує встановлення в довірчому середовищі, а zk-STARK потребує більших обчислювальних витрат.

В результаті цього протоколи ZKP можуть бути використані в низькі областей, такі як: блокчейн; фінанси; голосування; автентифікація; машинне навчання.

Сучасні стартапи і компанії в основному активно використовують такі протоколи для сфери блокчейн: Zcash; StarkWare; AZTEK; StarkEx; Coda Protocol. Існують ще багато сфер використання таких протоколів, в тому числі і в державних установах, системах автентифікації, забезпечення кваліфікованого та чесного голосування тощо.

Перспективи розвитку таких протоколів величезні, з розвитком швидкості обчислювальної технікі та еволюції алгоритмів швидких

розрахунків, будуть і пришвидшуватися розрахунки доказів протоколів ZKP, що дозволить його використовувати активно не тільки в блокчейн системах, а і в інших галузях, які мають забезпечувати конфіденційність інформації або їх користувачів.

2 РОЗРОБКА АЛГОРИТМУ ВЕРИФІКАЦІЇ З ZKP

2.1 Алгоритм верифікації на основі константи

Розробка алгоритму верифікації згенерованих доведень виконувалась мовою Rust з використанням бібліотеки Halo2, яка спрощує роботу з формування і перевірки доведень на основі введення системи Схем як рішення побудови доведення і його верифікації в системі[8]. На рисунку 2.1 показана архітектура побудови і перевірки доведення на прикладі двох осіб та початкових значень.

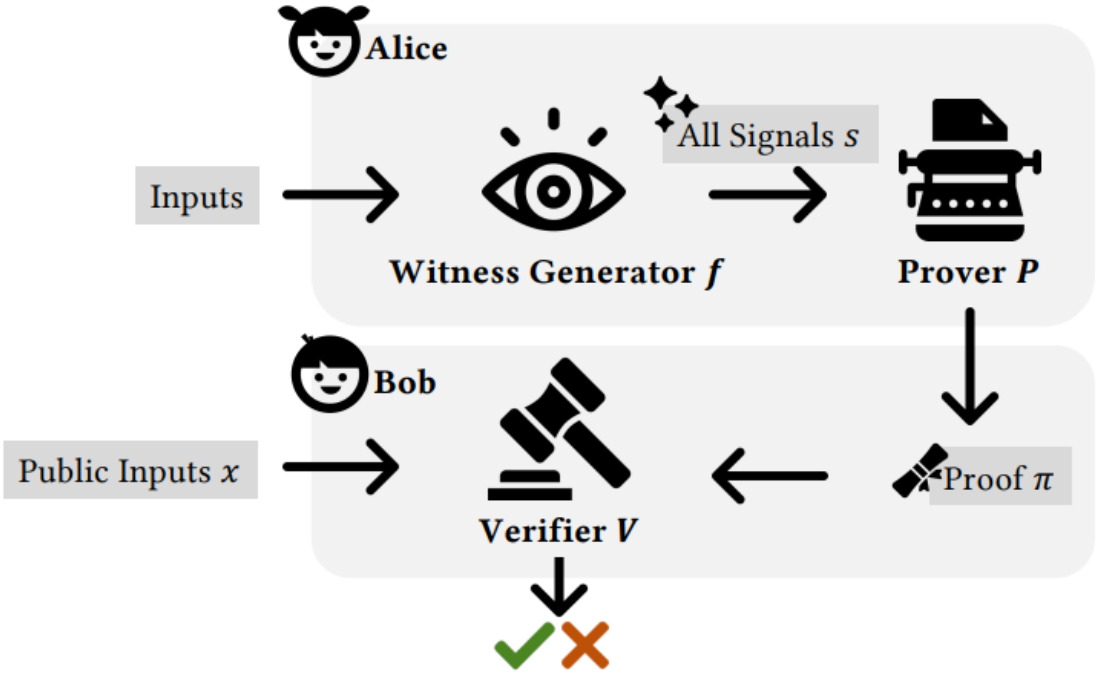

Рисунок 2.1 – Архітектура побудови і перевірки доведення.

На рисунку показана схема роботи довідника і верифікатора в системах, які побудовані на основі ZKP[9].

- «Аліса» генерує доведення використовуючи вхідні дані, формує Claim та Witness, які використовуються для генерації доведення;

- Після цього сформована комбінація проходить через певний довідник, який на виході генерує доведення;
- Доведення надходить до верифікатора, який за допомогою «Боба» та доступних йому вхідних даних робить висновок щодо дійсності твердження.

Розглянемо побудову і перевірку такої архітектури за допомогою бібліотеки Halo2 з використанням константи у вигляді вхідних даних. На лістингу 2.1 наведені сформовані правила для формування схеми.

Лістинг 2.1 – Формування правил для константи

```
impl<F: Field> Circuit<F> for MyCircuit<F> {
    type Config = FieldConfig;
    type FloorPlanner = SimpleFloorPlanner;

    fn without_witnesses(&self) -> Self {
        Self::default()
    }

    fn configure(meta: &mut ConstraintSystem<F>) -> Self::Config {
        let advice = [meta.advice_column(), meta.advice_column()];
        let instance = meta.instance_column();
        let constant = meta.fixed_column();

        FieldChip::configure(meta, advice, instance, constant)
    }

    fn synthesize(
        &self,
        config: Self::Config,
        mut layouter: impl Layouter<F>,
    ) -> Result<(), Error> {
        let field_chip = FieldChip::<F>::construct(config);

        let a = field_chip.load_private(layouter.namespace(|| "load a"), self.a)?;
        let b = field_chip.load_private(layouter.namespace(|| "load b"), self.b)?;

        let constant =
            field_chip.load_constant(layouter.namespace(|| "load constant"),
self.constant)?;

        let ab = field_chip.mul(layouter.namespace(|| "a * b"), a, b)?;
        let absq = field_chip.mul(layouter.namespace(|| "ab * ab"), ab.clone(),
ab)?;
        let c = field_chip.mul(layouter.namespace(|| "constant * absq"), constant,
absq)?;
```

```
        field_chip.expose_public(layouter.namespace(|| "expose c"), c, 0)
    }
}
```

В данному випадку після створення схеми відкритим вводом буде лише константа c. Вона ж і представляє з себе ключ за яким доведення буде верифіковано:

- Після створення схеми з числами constant = 7, a = 2, b = 3, маємо деяку константу c, яка представляє з себе відкритий ввод;
- В разі збігання відкритого вводу з деяким деяким секретним значенням (в нашому випадку це 7 * 2^2 * 3^2), доведення верифіковано, людина має доступ до секретної інформації;
- В разі неправильного виводу значення, доведення не верифіковано, людина нічого не знає про секретну інформацію.

Лістинг 2.2 показує формування доведення на основі константи з вищеописаної послідовності дій.

Лістинг 2.2 – Формування доведення на основі константи

```
use halo2_proofs::{dev::MockProver, pasta::Fp};
let k = 4;

let constant = Fp::from(7);
let a = Fp::from(2);
let b = Fp::from(3);
let c = constant * a.square() * b.square();

let circuit = MyCircuit {
    constant,
    a: Value::known(a),
    b: Value::known(b),
};

let mut public_inputs = vec![c];
let prover = MockProver::run(k, &circuit, vec![public_inputs.clone()]).unwrap();
assert_eq!(prover.verify(), Ok(()));

public_inputs[0] += Fp::one();
let prover = MockProver::run(k, &circuit, vec![public_inputs]).unwrap();
assert!(prover.verify().is_err());
```

В першому випадку тестування успішне, верифікатор перевірив підпис і повідомив про позитивний результат. В іншому випаду базове значення константи змінюється на 1, після перевірки верифікатор повідомив про негативний результат, що є правильним значенням для тесту, отже доведення для створеної схеми працює.

2.2 Алгоритм верифікації на основі послідовностей

Розглянемо алгоритм формування і перевірки доведення на основі послідовності Фібоначчі. Лістинг 2.3 показує сформовані правила для схеми на основі послідовності Фібоначчі.

Лістинг 2.3 – Формування схеми на послідовності Фібоначчі

```
impl<F: Field> Circuit<F> for MyCircuit<F> {
    type Config = FiboConfig;
    type FloorPlanner = SimpleFloorPlanner;

    fn without_witnesses(&self) -> Self {
        Self::default()
    }

    fn configure(meta: &mut ConstraintSystem<F>) -> Self::Config {
        let col_a = meta.advice_column();
        let col_b = meta.advice_column();
        let col_c = meta.advice_column();
        let instance = meta.instance_column();

        FiboChip::configure(meta, [col_a, col_b, col_c], instance)
    }

    fn synthesize(&self, config: Self::Config, mut layouter: impl Layouter<F>) ->
Result<(), Error> {
        let chip = FiboChip::construct(config);
        let (prev_a, mut prev_b, mut prev_c) = chip.assign_first_row(
            layouter.namespace(|| "first_row"),
            self.a, self.b
        )?;

        chip.expose_public(layouter.namespace(|| "private a"), &prev_a, 0)?;
        chip.expose_public(layouter.namespace(|| "private b"), &prev_b, 1)?;

        for _i in 3..10{
            let c_cell = chip.assign_row(
```

```
                    layouter.namespace(|| "next row"),
                    &prev_b,
                    &prev_c,
                )?;
                prev_b = prev_c;
                prev_c = c_cell;
            }

            chip.expose_public(layouter.namespace(|| "out"), &prev_c, 2)?;
        }
}
```

В даному випадку після створення схеми відкритими значеннями будуть константи a, b та out. Вони будуть представлять вхідні значення під час процесу верифікації доведення:

- Для створення послідовності Фібоначчі необхідно перші 2 значення в послідовності налаштувати як 0 та 1, після чого кожне наступне значення буде дорівнювати сумі двох минулих;
- Згідно з написаних правил, таблиця «Відбитків» буде представляти з собою цю саму послідовність з 1-го елемента до 10-го, яке буде дорівнювати 34;
- Якщо вхідні значення будуть відповідати відкритим описаним значенням в системі (0, 1, 34), то верифікатор дасть позитивний відгук, особа знає про таємну інформацію;

Лістинг 2.4 демонструє процес перевірки сформованої схеми на основі вищеописаних дій.

Лістинг 2.4 – Перевірка сформованої схеми

```
let k = 4;
let a = Fp::from(0);
let b = Fp::from(1);
let out = Fp::from(34);

let circuit = MyCircuit {
    a: Value::known(a), b: Value::known(b),
};


let mut public_inputs = vec![a,b,out];
```

```
let prover = MockProver::run(k, &circuit, vec![public_inputs.clone()]).unwrap();
assert_eq!(prover.verify(), Ok(()));

public_inputs[2] += Fp::one();
let prover = MockProver::run(k, &circuit, vec![public_inputs]).unwrap();
assert!(prover.verify().is_err());
```

В результаті перевірки програма не видає помилок, отже доведення можна вважати доцільним для використання в сформованій схемі.

2.3 Алгоритм верифікації на основі ігор

Подібні алгоритми створення схем доведень та їх верифікації можна реалізувати і на більш складних системах. Наприклад – в теорії ігор. Для формування схеми я обрав правила ігри «Камінь – Ножиці – Папір», в яких два гравця обирають один з трьох варіантів, після чого відповідно від результату обирається переможець.

За правила до гри була обрана стандартна конфігурація гри:

- Дві однакові фігури – нічия;
- Камінь перемагає ножиці;
- Ножиці перемагають папір;
- Папір перемагає камінь.

На рисунку 2.2 показана ілюстрація вищеописаних правил в грі «Камінь – Ножиці – Папір».

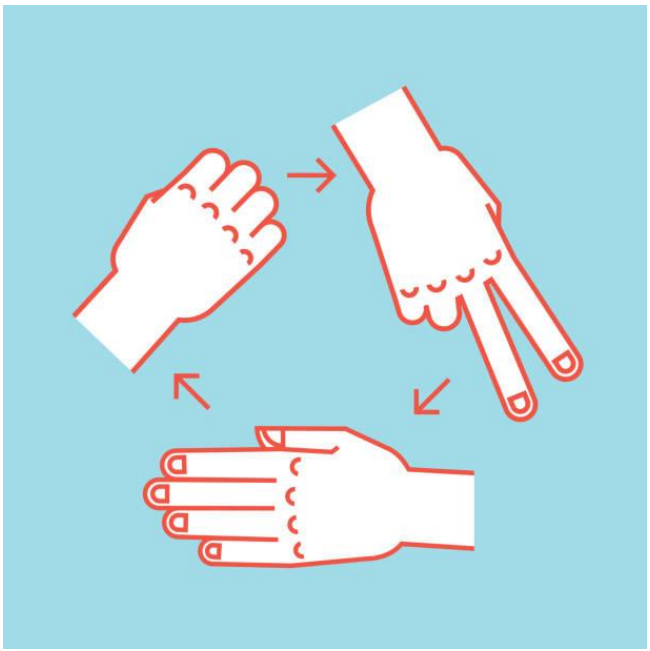

Рисунок 2.2 – Ілюстрація ігрового процесу «Камінь – Ножиці – Папір».

Представимо кожну фігуру у вигляді значень: 0, 1 та 2. Тоді для визначення перемоги чи нічиї достатньо застосувати наступні формули:

- x – y = 0 – нічия.
- (y + 2 – x) * (y – 1 – x) = 0 – перемога другого гравця (y).

В результаті сформованих умов можемо згенерувати правила для написання схеми доведення. На лістингу 2.5 показаний результат сформованих правил у грі для бібліотеки Halo2.

Лістинг 2.5 – Формування схеми на грі «Камінь – Ножиці – Папір»

```
impl<F: PrimeField, const N: usize> Circuit<F> for AdventCircuit<F, N> {
    type Config = AdventConfig<F>;
    type FloorPlanner = SimpleFloorPlanner;

    fn without_witnesses(&self) -> Self {
        Self {
            xs: [Value::unknown(); N],
            ys: [Value::unknown(); N],
        }
    }

    fn configure(meta: &mut ConstraintSystem<F>) -> Self::Config {
        let col_x = meta.advice_column();
```

```
        let col_y = meta.advice_column();
        let col_accum = meta.advice_column();
        let instance = meta.instance_column();
        let constant = meta.fixed_column();

        AdventChip::<F,N>::configure(meta, [col_x, col_y, col_accum], instance,
constant)
    }

    fn synthesize(&self, config: Self::Config, mut layouter: impl Layouter<F>) ->
Result<(), Error> {
        let chip = AdventChip::construct(config);

        let out_cell = chip.assign(layouter.namespace(|| "rps"), self.xs,
self.ys)?;
        chip.expose_public(layouter.namespace(|| "out"), &out_cell, N-1)?;

        Ok(())
    }
}
```

В створеній схемі чип будується на творчому методі construct, у якому розписані правила розрахування перемоги сторін. Вихідний код чипа показаний в Додатку Б. В змодельованій схемі вихідним значенням буде параметр out, який є вихідним рахунком для всіх гравців за правилами:

- В разі перемоги нараховується 6 очок;
- В разі нічиї нараховується 3;
- В разі програшу нараховується 0.

На лістингу 2.6 реалізований процес перевірки схеми на основі гри з використанням вищеописаних правил.

Лістинг 2.6 – Перевірка схеми на основі гри «Камінь – Ножиці – Папір»

```
let k = 4;

let xs = [0,1,2];
let ys = [1,0,2];
let out = 15;

let mut public_inputs = vec![Fp::zero(); 3];
public_inputs[2] = Fp::from(out);

let circuit = make_circuit(xs, ys, out);
```

```
let prover = MockProver::run(k, &circuit, vec![public_inputs.clone()]).unwrap();
assert_eq!(prover.verify(), Ok(()));

public_inputs[2] = Fp::one();
let prover = MockProver::run(k, &circuit, vec![public_inputs.clone()]).unwrap();
assert!(prover.verify().is_err());
```

В результаті маємо два набору фігурок: (0, 1, 2) та (1, 0, 2), відкрите значення 15. Для розрахунку звичайного відкритого значення достатньо використати формулу для перемоги та нічиї. В випадку перемоги необхідно зарахувати 6 очок, в випадку програшу – 3, також за визначеною в архітектурі формулою необхідно зараховувати 1 + y, де y – поточне значення в раунді кожного разу. Тоді маємо:

- Перший раунд: 6 + 1 + 1 = 8;
- Другий раунд: 8 + 1 = 9;
- Третій раунд: 9 + 3 + 2 + 1 = 15.

Результати збігаються, верифікатор надає позитивний відгук. В іншому випадку, як і в минулих алгоритмах до вихідного значення додається одиниця. Верифікатор надає негативний відгук, доведення працює коректно.

2.4 Висновки до другого розділу

Побудова і перевірка доведення в ZKP тісно зв’язані одним з одним, оскільки при побудові доведення генеруються публічні значення, які мають бути надані при перевірці. В разі збігання цих значень, особа вважається носієм інформації, на яку було створено доведення.

Алгоритм формування доведення в Halo2 базується на формуванні Схеми, яка представляє собою поведінку та список правил в сформованому доведенні. Для кожної такої схеми обираються вхідні приватні та публічні значення:

- Приватними значеннями в схемі є вхідні в схему параметри;

- Публічними значеннями в схемі є розраховані всередині схеми значення, які потім використовують для верифікації доведення.

Всього було сформовано 3 різні схеми:

- Схема на константі;
- Схема на послідовності чисел;
- Схема на грі.

За рахунок сформованих алгоритмів, була аргументована доцільність використання доведень як гарантії знання про інформацію конкретних осіб. В подальшому сформовані алгоритми можна використовувати для ознайомлення з бібліотекою Halo2 та роботою ZKP-рішень в сучасних мовах програмування.

ВИСНОВКИ

В роботі основну роль виконують протоколи доведення з нульовим розголошенням – ZKP, які покращують приватність і масштабованість сучасних блокчейн систем.

Виконана класифікація ZKP за реалізацією протоколів, перераховані сучасні рішення і компанії, які використовують такі доведення. Визначені слабкі і сильні сторони доведення з нульовим розголошенням для найпопулярніших протоколів.

Загалом існує два типу ZKP: інтерактивні (Interactive) та не інтерактивні (Non-Interactive), в свою чергу вони поділяються на протоколи з реалізаціями (zk-SNARK, zk-STARK, BulletProofs, Hyrax тощо). Кожне таке рішення пропонує доведення наявності інформації у особи без її викриття відповідно до правил побудови ZKP. Такі особливості в роботі розглянуті для протоколів zk-SNARK, zk-STARK та BulletProofs.

В результаті виконання роботи були сформовані алгоритми створення доведень на основі використання бібліотеки Halo2 на мові програмування Rust. Бібліотека представляє з себе набір інструментів для формування Схем – набору правил формування доведення з приватними та публічними параметрами. В разі успішності збігу всіх публічних параметрів особою, яка бажає довести факт знання певного твердження, вона проходить верифікацію та вважається носієм цього твердження.

В якості тверджень були обрані три наступні системи:

- Певне правило алгебраїчної операції над числами. Публічні параметри – результат операції;
- Певна послідовність чисел. Публічні параметри – останнє число послідовності;

- Певна сесія гри в «Камень – Ножиці – Папір». Публічні параметри – результат ігор для другого гравця у вигляді очок.

Для цих тверджень були створені відповідні Схеми, проведені тестування як в разі правильних публічних параметрів, так і неправильних. В разі введення правильних параметрів – система вважала доведення дійсним, тобто особа являється носієм інформації. В разі введення неправильних параметрів – система верифікації надавала негативний відгук, особа не являється носієм інформації.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Доведення з нульовим розголошенням – Юз-Кейси та Додатки [Стаття] / Компанія «Chainlink» // Відео та архітектура. – 2024. – Режим доступу до статті:

https://chain.link/education-hub/zero-knowledge-proof-use-cases

2. Обіцянка доведення з нульовим розголошенням – Масштабованість, Швидкість, Приватність даних та Безпека [Стаття] / Компанія «OpenZeppelin» // Вимоги до ZKP. – 2024. – Режим доступу до статті:

https://www.openzeppelin.com/zkp

3. Доведення з нульовим розголошенням – як це працює і чому це важливо [Стаття] / Компанія «MangroviaBlockchain» // Приклади використання ZKP. – 2024. – Режим доступу до статті:

https://www.mangrovia.solutions/en/zero-knowledge-proof-how-it-works-and-why-its-important/

4. Як доведення з нульовим розголошенням можуть покращити приватність та масштабованість Блокчейну? [Стаття] / Компанія «Chain» // Масштабованість та приватність ZKP в Блокчейн-рішеннях. – 2023. – Режим доступу до статті:

https://medium.com/@chaincom/chain-insights-how-zero-knowledge-proofs-can-enhance-blockchains-privacy-and-scalability-8b72dad4f230

5. Опитування щодо доведення з нульовим розголошенням в блокчейні [Дослідження] / IEEE Network // Проблеми та покращення сучасних ZKP-рішень. – 2021. – Режим доступу до дослідження:

https://ieeexplore.ieee.org/abstract/document/9520375

6. SNARKs на мові програмування C: Верифікація програмних рішень з використанням нульового знання [Стаття] / [Eli Ben-Sasson, Alessandro

Chiesa, Daniel Genkin, Eran Tromer, та Madars Virza] // Оптимізація SNARK. – 2014. – Режим доступу до статті:

https://www.iacr.org/archive/crypto2013/80420214/80420214.pdf

7. Машстабоване нульове знання без довірчого середовища [Стаття] / [Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, та Michael Riabzev] // Переваги та недоліки STARK. – 2018. – Режим доступу до статті:

https://www.iacr.org/archive/crypto2019/116940201/116940201.pdf

8. MiMC для нулового знання [Електронний ресурс] / Primesym // Блоковий шифр для SNARK додатків. – 2019. – Режим доступу до ресурсу:

https://byt3bit.github.io/primesym/mimc/

9. Практичний аналіз захисту схем доведення з нульовим розголошенням [Стаття] / [Hongbo Wen, Jon Stephens, Yanju Chen, Kostas Ferles, Shankara Pailoor та інш.] // Побудова та верифікація доведення. – 2023. – Режим доступу до статті:

https://eprint.iacr.org/2023/190.pdf

ДОДАТОК А. ПРИКЛАД СФОРМОВАНОГО ДОВЕДЕННЯ


```
"trace_commitment":
"0x7a75155baa6711984e5133f2e462696d64efe8b100000000000000000000000000",
"constraint_commitment":
"0x4a1fe6cc62fddef3764e3a90859b120513cdc0cb000000000000000000000000000",
"trace_oods_values":
["0x02847fa2d3fc723c706a2723ed72ee3e3c1f7e784a22c19d85c370f99b1451bb"],
"constraint_oods_values":
["0x0717854e39b956977359cd416ff3d7c2eeb895450910a5848a50698d45180262"],
"fri_commitments":
["0xfb3344f9d6e4b7a108b5c1ec21b296d00b44953200000000000000000000000000"],
"last_layer_coefficients":
["0x02635f9869a2fe55d8001fc1683db790369e8dbd587b25bda21041872543e86c"],
"pow_nonce": "0x00000000000000c8","trace_values":
["0x029e64bf65829a905c360429392506cb8a461deecb2927eaac9bb0f4f388815a",
"0x07bdfc9cb48ea2ef8e6a2d0b6ed99b58b30838aa23abac00e4d762fc51b01201"],
"trace_decommitment":
["0x056149e30d1f4d00a3c9fbd6c6daf93475b9e21134d6d81553644a41c7ea6e27",
"0x0041b205be1344a17195d2f4912664a74cf7c755dc5453ff1b28983a69c2dd80",
"0xf7fbf70f04f64c1923029af2395e69590e43190600000000000000000000000000",
"0x7e6821de58a81ed0d78e50ebbeca8f26d429e78f000000000000000000000000",
"0x6ae0f192189b14f240d95e0c8aac68b14038cdb600000000000000000000000000",
"0xbeb74646bcf2b5aae00c7edb23540a01656bd5ce000000000000000000000000",
"0x7a9bece36e3109b57779b5769c627eaf5b614cda000000000000000000000000",
"0x9f92fb3ef67f7e747b93a440807c0e6fb7fb6baf000000000000000000000000"],
"constraint_values":
["0x03fba848cd948c4ed2eea6c8b73007298da09ab8adeb257d7c22b8cf35e58d6d",
"0x0296dbc4f17a2b1918b7c8d2eb038d2419a8e3a0a760271c388b4a1af121129e"],
"constraint_decommitment":
["0x05635e2cf94e1ac8745af04e8a03a79007f8e5f905c339cbbb335965ceddb55e",
"0x06c82ab0d5687bfe2e91ce44563021957bf09d110c4e382cfecac81a13a2302d",
"0x8eb26a914a60f2fdd0f2d017e30ef0e969b7f36500000000000000000000000000",
"0x537609ba98cc263319815b18f666e60e8fe807fb000000000000000000000000",
"0xd42263067416a3ea271dc8a1caa4a63e6c584db800000000000000000000000000",
"0xfaee27d5804a7dfae429a2abb382483564c1e960000000000000000000000000",
"0x6cf6164ea69a25d23c8841894346b7c66312ad670000000000000000000000000",
"0x1a892fb74f3522b4d7846dfe54301083b8e4ad5d000000000000000000000000"],
"fri_values":
[[
"0x0131afcc34d17f2aec000fe0b41edbc81b4f46deac3d92ded10820c392a1f436",
"0x0131afcc34d17f2aec000fe0b41edbc81b4f46deac3d92ded10820c392a1f436"]],
"fri_decommitments":
[["0x9bddf694fa77858fc252e5542a812a5d7f5669fc000000000000000000000000",
"0x9bddf694fa77858fc252e5542a812a5d7f5669fc000000000000000000000000",
"0x8e9cc63480610eccf4a8d9400820543c661ab4c800000000000000000000000000",
"0x8e9cc63480610eccf4a8d9400820543c661ab4c800000000000000000000000000",
"0x22b5cd2647eb8429cf4a0498b00850c3a52c702a000000000000000000000000",
"0x22b5cd2647eb8429cf4a0498b00850c3a52c702a000000000000000000000000"]]
```

ДОДАТОК Б. АРХІТЕКТУРА ЧИПУ «КАМІНЬ – НОЖИЦІ – ПАПІР»

```
impl<F: PrimeField, const N: usize> AdventChip<F, N> {
    fn construct(config: AdventConfig<F>) -> Self {
        Self { config }
    }

    fn configure(meta: &mut ConstraintSystem<F>, advice: [Column<Advice>; 3],
instance: Column<Instance>, constant: Column<Fixed>) -> AdventConfig<F> {
        let [col_x, col_y, col_accum] = advice;

        let selector: Selector = meta.selector();

        meta.enable_equality(col_accum);
        meta.enable_equality(instance);

        meta.enable_constant(constant);

        let y_wins = IsZeroChip::configure(
            meta,
            |meta| meta.query_selector(selector),
            |meta| {
                let x = meta.query_advice(col_x, Rotation::cur());
                let y = meta.query_advice(col_y, Rotation::cur());
                (y.clone() + const_val(2) - x.clone()) * (y - const_val(1) - x)
            }
        );

        let is_draw = IsZeroChip::configure(
            meta,
            |meta| meta.query_selector(selector),
            |meta| {
                let x = meta.query_advice(col_x, Rotation::cur());
                let y = meta.query_advice(col_y, Rotation::cur());
                y - x
            }
        );

        meta.create_gate("round", |meta| {
            let s = meta.query_selector(selector);

            let x = meta.query_advice(col_x, Rotation::cur());
            let y = meta.query_advice(col_y, Rotation::cur());
            let accum = meta.query_advice(col_accum, Rotation::cur());

            let out = meta.query_advice(col_accum, Rotation::next());

            vec![
```

```
                s.clone() * (out - (y_wins.expr() * F::from(6) + is_draw.expr() *
F::from(3) + y.clone() + const_val(1) + accum)),
                s.clone() * x.clone() * (x.clone() - const_val(1)) * (x.clone() -
const_val(2)),
                s.clone() * y.clone() * (y.clone() - const_val(1)) * (y.clone() -
const_val(2)),
            ]
        });

        AdventConfig { advice: [col_x, col_y, col_accum], selector, instance,
is_zero_chips: [y_wins, is_draw] }
    }

    fn assign(&self, mut layouter: impl Layouter<F>, xs: [Value<F>; N], ys:
[Value<F>; N]) -> Result<AssignedCell<F,F>, Error> {
        layouter.assign_region(
            || "rps game",
            |mut region| {
                let [col_x, col_y, col_accum] = self.config.advice;
                let [y_wins, is_draw] = &self.config.is_zero_chips;
                let mut accum_value: Value<F> = Value::known(F::ZERO);
                let mut out_cell = Err(Error::Synthesis);

                for row in 0..N {
                    let [x, y] = [xs[row], ys[row]];

                    self.config.selector.enable(&mut region, row)?;

                    if row == 0 {
                        region.assign_advice_from_constant(
                            || "zero",
                            col_accum,
                            0,
                            F::ZERO
                        )?;
                    }

                    region.assign_advice(
                        || format!("x[{}]", row),
                        col_x,
                        row,
                        || x
                    )?;

                    region.assign_advice(
                        || format!("y[{}]", row),
                        col_y,
                        row,
                        || y
                    )?;
```

```
                    let y_wins_chip = IsZeroChip::construct(y_wins.clone());
                    let y_wins_value = (y + Value::known(F::from(2)) - x) * (y -
Value::known(F::ONE) - x);
                    let y_wins = y_wins_chip.assign(&mut region, row,
y_wins_value)?;

                    let is_draw_chip = IsZeroChip::construct(is_draw.clone());
                    let is_draw_value = y - x;
                    let is_draw = is_draw_chip.assign(&mut region, row,
is_draw_value)?;

                    let round_score = y_wins.zip(is_draw).and_then(|(y_wins,
is_draw)| {
                        let partial_score = if y_wins { 6 } else if is_draw { 3 }
else { 0 };
                        Value::known(F::from(partial_score)) + y +
Value::known(F::ONE)
                    });

                    accum_value = accum_value + round_score;
                    out_cell = region.assign_advice(
                        || format!("out[{}]", row),
                        col_accum,
                        row + 1,
                        || accum_value
                    );
                };

                out_cell
            })
    }

    pub fn expose_public(&self, mut layouter: impl Layouter<F>, cell:
&AssignedCell<F,F>, round: usize) -> Result<(), Error> {
        layouter.constrain_instance(cell.cell(), self.config.instance, round)
    }

}

struct AdventCircuit<F: PrimeField, const N: usize> {
    xs: [Value<F>; N],
    ys: [Value<F>; N],
}

impl<F: PrimeField, const N: usize> Circuit<F> for AdventCircuit<F, N> {
    type Config = AdventConfig<F>;
    type FloorPlanner = SimpleFloorPlanner;

    fn without_witnesses(&self) -> Self {
        Self {
            xs: [Value::unknown(); N],
```

```
            ys: [Value::unknown(); N],
        }
    }

    fn configure(meta: &mut ConstraintSystem<F>) -> Self::Config {
        let col_x = meta.advice_column();
        let col_y = meta.advice_column();
        let col_accum = meta.advice_column();
        let instance = meta.instance_column();
        let constant = meta.fixed_column();

        AdventChip::<F,N>::configure(meta, [col_x, col_y, col_accum], instance,
constant)
    }

    fn synthesize(&self, config: Self::Config, mut layouter: impl Layouter<F>) ->
Result<(), Error> {
        let chip = AdventChip::construct(config);

        let out_cell = chip.assign(layouter.namespace(|| "rps"), self.xs,
self.ys)?;
        chip.expose_public(layouter.namespace(|| "out"), &out_cell, N-1)?;

        Ok(())
    }
}
```