**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
**ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ**
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ

«На правах рукопису»
УДК _____

«До захисту допущено»
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
“15” травня 2018р.

# Магістерська дисертація

**на здобуття ступеня магістра**

зі спеціальності 113 «Прикладна математика»
(код і назва)

на тему: Криптографічні протоколи електронної готівки на основі технології блокчейн

Виконав: студент 6 курсу, групи ФІ-63м
(шифр групи)

Чорний Олег Миколайович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник професор, д.т.н., с.н.с., Кудін А. М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

**Київ – 2018 року**

**Національний технічний університет України**
**«Київський політехнічний інститут**
**імені Ігоря Сікорського»**
Фізико-технічний інститут
Кафедра математичних методів захисту інформації

Рівень вищої освіти – другий (магістерський)

Спеціальність 113 «Прикладна математика»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
«___»_____2018 р.

# ЗАВДАННЯ
## на магістерську дисертацію студенту

Чорному Олегу Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації: Криптографічні протоколи електронної готівки на основі технології блокчейн,
науковий керівник дисертації: Кудін Антон Михайлович, д.т.н., с.н.с.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 02.07.2018р. № 1058-с

2. Термін подання студентом дисертації: 16.05.2018р.

3. Об'єкт дослідження: системи електронних платежів

4. Предмет дослідження: застосування децентралізованих систем зберігання та обробки даних для розробки систем електронних платежів

5. Перелік завдань, які потрібно розробити: 1) проаналізувати існуючі методи вирішення поставленої задачі 2) визначити дані, які необхідно зберігати у блокчейні 3) розробити процедури та алгоритми обробки даних, які зберігаються у системи, акцентуючи увагу на безпеці та збереженню анонімності користувачів та проведених платежів 4) проаналізувати розроблене рішення, зробити висновки

6. Орієнтовний перелік ілюстративного матеріалу: 74 сторінки, 9 рисунків, 13 джерел

7. Орієнтовний перелік публікацій: матеріали XVI Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики».

8. Консультанти розділів дисертації

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

9. Дата видачі завдання: _____

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Термін виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| 1 | Вибір теми наукового дослідження | 01.09.17 – 30.09.17 | Виконано |
| 2 | Аналіз існуючих методів вирішення проблеми, їх недоліків | 01.10.17 – 31.10.17 | Виконано |
| 3 | Вибір даних, які зберігаються у смарт-контракті, що лежить в основі системи | 01.11.17 – 31.12.17 | Виконано |
| 4 | Розробка процедур і алгоритмів обробки даних для смарт-контракту | 01.01.18 – 28.02.18 | Виконано |
| 5 | Аналіз властивостей, отриманої системи | 01.03.18 – 31.03.18 | Виконано |
| 6 | Оформлення звіту | 01.04.18 – 30.04.18 | Виконано |
| | | | |
| | | | |

Студент _____ О. М. Чорний
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ А. М. Кудін
(підпис) (ініціали, прізвище)

# РЕФЕРАТ

Кваліфікаційна робота містить: 74 сторінки, 9 рисунків, 13 джерел.

У роботі досліджуються системи електронних платежів та можливості їх вдосконалення. Метою є зменшення вимог до обчислювальних ресурсів, необхідних для підтримки функціонування таких систем. Предметом дослідження є застосування децентралізованих систем обробки та зберігання інформації до розробки систем електронних платежів.

Було запропоновано модифікацію системи електронних платежів Чаума, яка дозволяє передавати монети між користувачами системи. При цьому система забезпечує безпеку всіх операцій, повну анонімність користувачів та неможливість подвійної витрати монет.

ЕЛЕКТРОННІ ПЛАТЕЖІ, БЛОКЧЕЙН, СМАРТ-КОНТРАКТИ

# ABSTRACT

74-pages work contains 9 illustrations, 13 literature links.

Electronic payments systems and possibilities for their improvement are researched in this paper. The aim of the work is to reduce requirements for the computational resources necessary to support functioning of such systems. The subject of the research is application of decentralized systems of data storing and processing to the development of electronic payment systems.

Modification of the Chaum electronic payment system, which allows system's users to transfer coins to each other was proposed. System ensures security of all operations, full anonymity of users and prevents double spending.

ELECTRONIC PAYMENTS, BLOCKCHAIN, SMART CONTRACTS

**ЗМІСТ**

# ВСТУП

**Актуальність роботи.** Системи електронних платежів є однією із найбільш важливих складових будь-якої комерційної діяльності, оскільки вони здатні підвищити як швидкість так і зручність виконання розрахунків. Останні розробки у сфері децентралізованих систем зберігання та обробки даних дозволяють використовувати їх для розробки нових та вдосконалення існуючих систем електронних платежів.

**Метою дослідження** є застосування децентралізованих систем зберігання та обробки даних до побудови систем електронних платежів з метою зменшення вимог до обчислювальних ресурсів, необхідних банку для підтримки функціонування такої системи.

**Задачею дослідження** є вдосконалення системи електронних платежів Чаума для забезпечення можливості анонімної передачі монет між користувачами із використанням технології децентралізованого зберігання і обробки даних Ethereum. Для розв’язання задачі необхідно вирішити такі завдання:

1) проаналізувати існуючі методи вирішення поставленої задачі

2) визначити дані, які необхідно зберігати у блокчейні

3) розробити процедури та алгоритми обробки даних, які зберігаються у системи, акцентуючи увагу на безпеці та збереженню анонімності користувачів та проведених платежів

4) проаналізувати розроблене рішення, зробити висновки

*Об’єкт дослідження***:** системи електронних платежів.

*Предмет дослідження***:** застосування децентралізованих систем зберігання та обробки даних для розробки систем електронних платежів.

При розв’язанні поставлених завдань використовувались наступні *методи дослідження***:** методи математичного і комп’ютерного моделювання, дискретної математики, теорії алгоритмів.

**Наукова новизна одержаних результатів** полягає в тому, що було вдосконалено систему електронних платежів Чаума, додавши до неї можливість анонімної передачі монет між користувачами.

**Практичне значення одержаних результатів** полягає в тому, що запропоновані алгоритми та математичну модель можна використовувати для розробки повноцінної системи електронних платежів.

**Апробація результатів та публікації.** Основні результати було опубліковано у матеріалах XVI Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики».

# 1 ТЕОРЕТИЧНІ ВІДОМОСТІ

У даному розділі розглядаються системи електронних платежів, підходи до їх побудови, притаманні сучасним системам переваги та недоліки. Далі розглядається технологія блокчейн та, побудована на її основі, платформа смарт-контрактів Ethereum, яка може бути використана для вдосконалення розглянутих платіжних систем.

## 1.1 Системи електронних платежів

### 1.1.1 Класичні гроші

Добре відомо, що люди, які населяли Землю мільйони років тому і жили переважно племенами, виконували всю роботу і забезпечували себе всім необхідним для життя самостійно. Проте розвиток людства поступово призвів до виникнення ремесел та розподілення видів праці представниками різних родів та общин. В свою чергу, це означало, що для отримання всього необхідного людям необхідно було обмінюватися результатами своєї праці.

Первісна економіка спиралась на такі поняття як *дарування*, *борг* та *бартер*. Останній із них якраз і означає домовленість сторін про обмін майном однакової вартості (з точки зору сторін обміну). Враховуючи всі недоліки бартерного обміну, з часом виникла ідея виділити один певний товар, який міг би виконувати роль загального еквіваленту і використовуватись для проведення обміну щонайменше однією із сторін. Мова йде про виникнення такого поняття як *гроші*.

Спочатку широко використовувались так звані *товарні гроші*, коли в якості грошей виступав певний товар, який мав самостійну цінність і користь.

Головним прикладом такого виду грошей виступають дорогоцінні метали – срібло, золото, бронза. Використання суцільних злитків металу мало певні незручності, пов'язані із необхідністю точно вимірювати його вагу та визначати його пробу. Для попередження спроб шахрайства із часом виникли *монетні двори*, в яких викарбувані *монети* відмічали публічним клеймом. Іншою зручністю монет була їх висока вартість при відносно малій вазі, що було дуже зручно для проведення обміну.

Подальший розвиток економіки та банківської справи призвів до появи так званих *фіатних грошей*. Номінальна вартість таких грошей встановлюється і гарантується державою, в той час як їх власна вартість у порівнянні із номінальною нехтовно мала. Основною формою таких грошей є *банкноти*, виготовлені із паперу.

В зв'язку із історичним розвитком на сьогодні гроші мають такі функції.

– **Засіб обігу –** гроші використовуються як засіб оплати за товари і послуги, є посередником у обміні товарами та забезпечують їх обіг.

– **Міра вартості –** гроші використовуються для вимірювання і порівняння вартості товарів між собою на основі їх ціни.

– **Засіб платежу –** гроші використовуються для реєстрації та погашення боргових зобов'язань між суб'єктами економічних відносин.

– **Засіб накопичення –** невикористані гроші дозволяють переносити купівельну спроможність в майбутнє.

– **Світові гроші –** гроші функціонують як міжнародний купівельний та платіжний засіб, дозволяючи реалізовувати економічні взаємовідносини між країнами.

### 1.1.2 Комерційна діяльність

У дещо спрощеному вигляді, будь-яку економіку можна розглядати як процес виробництва товарів та послуг і *комерційної діяльності*, що пов'язана зі збутом товару з метою отримання прибутку [1]. Типовий бізнес-процес традиційної комерції складається із кроків зображених на рисунку 1.1.


**Рисунок 1.1** – Модель комерційної транзакції

1) Виробник товару створює новий продукт чи послугу і виводить його на ринок – самостійно чи за участю постачальників.

2) Проводиться рекламна кампанія, основною метою якої є ознайомлення якомога більшої кількості потенційних споживачів із новим товаром.

3) У разі виникнення потреби у певному товарі, споживач виконує пошук існуючих варіантів, порівнює знайдені альтернативи і, врешті, здійснює вибір на користь однієї із них.

4) На етапі переговорів із постачальником товару узгоджуються умови, ціна та термін доставки.

5) На наступному кроці формується замовлення, яке включає в себе перелік товарів та відповідних їм цін.

6) На основі отриманого замовлення, постачальник формує і надсилає споживачу *оферту* – пропозицію укласти угоду, що містить перелік домовлених умов надання послуг чи постачання продукції.

7) У разі прийняття усіх перелічених у оферті умов, споживач відповідає на неї *акцептом* – формальною згодою укладання угоди із постачальником.

8) Наступним етапом є проведення оплати, зручним для обох сторін способом. Оплата товару повинна бути взаємопов'язана із фізичною доставкою товару споживачу.

9) Далі відбувається власне постачання продукції або надання домовлених послуг постачальником споживачу.

10) Під час завершення угоди формуються документи, що підтверджують факт її проведення (рахунок-фактура, акт, накладна тощо).

11) В подальшому є можливою взаємодія учасників угоди одне з одним, пов’язана із питаннями післяпродажного обслуговування.

При роздрібному продажі товарів відносно невеликої вартості оферта, частіше всього, представляє собою рахунок, який передається споживачу поштою або особисто при зустрічі. Акцепт у такому випадку є оплатою отриманого рахунку.

Загальний технологічний розвиток привів до поступової автоматизації деяких етапів комерційних транзакцій, що дозволило суттєво підвищити швидкість їх проведення та знизити вартість послуг, за рахунок чого компаніям вдалось збільшити загальний рівень власного прибутку. Залучення обчислювальної техніки та мереж передачі даних до комерційної діяльності призвело до появи нового виду комерції – електронної комерції.

### 1.1.3 Електронна комерція

*Електронна комерція* (E-commerce) – це вид комерційної діяльності, проведення процесів якої суттєвим чином покладається на використання засобів обчислювальної техніки та комп'ютерних мереж. На рисунку 1.2 показано основних учасників електронної комерції та їх взаємодію між собою [1].

Стрімкий ріст популярності та розповсюдженості електронної комерції пов'язаний із ростом кількості користувачів глобальної мережі Інтернет. Найбільш яскравими прикладами компаній, які використовують Інтернет для здійснення своєї діяльності є компанія Amazon, орієнтована на продаж товарів масового вжитку, та компанія eBay, що займається проведенням онлайн-аукціонів.


**Рисунок 1.2** – Взаємодія учасників електронної комерції

Найчастіше розглядають наступних учасників електронної комерції:

– бізнес (B – Business)

– споживач (C – Consumer)

– держава (G – Government)

Залежно від характеру їх взаємодії розрізняють наступні типи систем електронної комерції[9]:

– B2C (Business to Consumer) – найбільш розповсюджений вид систем, у яких постачальник товару надає свої послуги безпосередньо кінцевим споживачам, користуючись для цього, наприклад, мережею Інтернет.

– B2B (Business to Business) – в такому випадку обидві сторони комерційних відносин є представниками бізнесу.

– C2C (Consumer to Consumer) – види систем, в яких кінцеві споживачі надають послуги або продають вироби іншим кінцевим споживачам. Як покупець так і продавець не є підприємцями у юридичному сенсі цього слова.

– B2G (Business to Government), G2B (Government to Business), G2C (Government to Consumer), C2G (Consumer to Government) – різні види взаємовідносин між бізнесом, споживачами та державою.

Ключовим питанням, пов'язаним з впровадженням і надійним функціонуванням системи електронної комерції, є забезпечення грошових розрахунків між постачальником і споживачем у електронному вигляді. Для досягнення цієї мети необхідно впровадження платіжних систем спеціального вигляду, що представляють собою електронні аналоги традиційних платіжних систем з використанням готівки, чеків, кредитних карт, банківських переказів. Принципова відмінність електронного платежів від традиційних полягає в тому, що весь процес від початку і до кінця відбувається в електронній (цифровій) формі, що породжує абсолютно нові вимоги до їх безпеки, що відображається на всіх етапах життєвого циклу таких систем.

### 1.1.4 Платіжні системи

*Платіжною системою* називається система, яка забезпечує перекази коштів від одного суб’єкта економіки до іншого, включаючи всі установи, інструменти, людей, процедури, правила і технології, які роблять проведення таких операцій можливим.

Історично сформувались наступні види платіжних систем [2]:

– системи готівкових платежів (cash-like)

– системи платежів по чекам та кредитним картам (cheque-like або credit-like)

– системи грошових переказів (money-transfer/remittance-like)

– системи з дебетними дорученнями (debit order)

*Системи електронних платежів* (далі СЕП) імітують роботу традиційних платіжних систем, працюючи у розподіленому обчислювальному середовищі.

У СЕП будь-якого виду завжди можна виокремити чотирьох основних учасників: *платника A*, *отримувача платежу B*, *банк платника* (або *емітент*) і *банк отримувача* (або *еквайер*). Банк емітент відповідальний за випуск платіжних засобів (наприклад, пластикових карт) і є гарантом виконання фінансових зобов'язань, пов'язаних з їх використанням. Банк еквайер займається обслуговуванням отримувачів платежів, які приймають до оплати вищезгадані засоби. Інколи до розгляду додаються ще *процесингові центри* – організації, що забезпечують учасникам розрахунків інформаційну та технологічну взаємодію.

На рисунку 1.3 в узагальненому вигляді показано рух коштів та банківської документації для систем усіх перелічених вище видів. Розглянемо детальніше кожну із них.

**Системи готівкових платежів.** Виконання платежу у такій системі складається із таких кроків.

1) Платник *A* виконує *зняття коштів із рахунку*, під час виконання якої він отримує від банку готівкові кошти, а банк зменшує суму на його рахунку на відповідну величину.

2) Платник *A* використовує отриману готівку для розрахунку за отримані від отримувача *B* товари або послуги. Ця операція називається *платежем*.

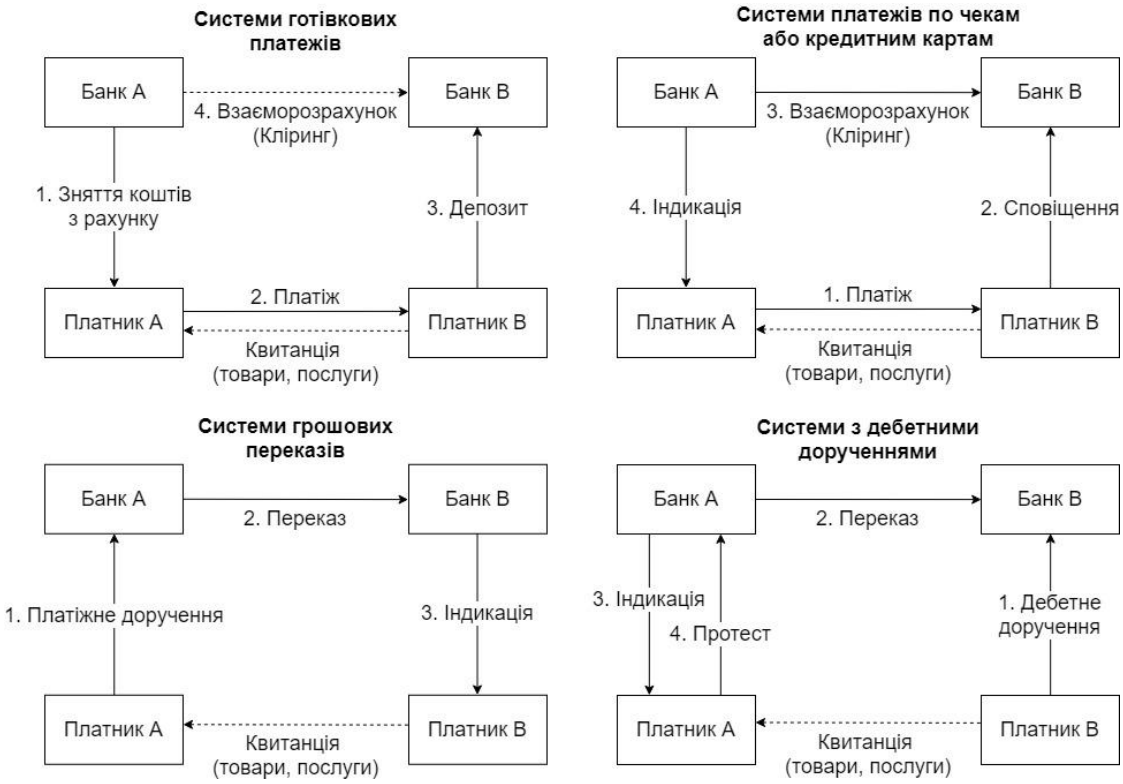


**Рисунок 1.3** – Типи платіжних систем

3) Отримувач *B* вносить готівку на рахунок в своєму банку, або, іншими словами, виконує операцію *депозиту.*

4) У СЕП необхідно додатково виконати *взаєморозрахунок* (або *кліринг*) між банками платника і отримувача. В ході виконання даної операції приводяться у відповідність кількість реальних та «електронних» коштів кожного банку, а також на рахунках платника та отримувача.

Готівкова є найбільш розповсюдженим способом розрахунку під час угод, що укладаються приватними особами або для розрахунків на відносно невелику суму коштів.

**Системи платежів по чекам та кредитним картам.** Принцип функціонування таких систем дещо відрізняється від систем з готівковими платежами.

1) Платник *A* передає отримувачу платежу *B* деяке посвідчення того, що він дійсно має намір провести платіж на вказану суму.

2) Отримувач платежу *B* передає це посвідчення до свого банку.

3) Використовуючи отримане посвідчення, банк отримувача звертається до банку платника для отримання реальних грошей (відбувається операція клірингу).

4) Банк платника надсилає платнику $A$ сповіщення про здійснений ним платіж.

Варто зауважити, що чеки є засобом виконання платежів із найбільшою собівартістю. Середня вартість обробки одного чека може досягати 1 долара США, що включає повний комплекс заходів по його виробництву та обслуговуванню.

**Системи грошових переказів.** Даний тип систем відрізняється від попереднього тим, що ініціатором платежу в них є платник.

1) Платник $A$ передає в свій банк платіжне доручення, в якому зазначається отримувач платежу, його банк та сума переказу.

2) Банк платника виконує взаємний розрахунок із банком отримувача.

3) Банк отримувача $B$ повідомляє його про надходження коштів.

Грошові перекази найчастіше використовуються для здійснення масових платежів, наприклад для виплати заробітної плати працівникам.

**Системи з дебетними дорученнями.** Для проведення платежу учасниками системи виконуються наступні дії.

1) Отримувач платежу $B$ повідомляє свій банк про намір отримати платіж від платника $A$.

2) Банк отримувача проводить взаєморозрахунок із банком платника.

3) Банк платника повідомляє його про те, що з його рахунку було списано певну суму коштів.

Очевидно, що така система може бути вразливою до зловживань з боку отримувачів платежів, тому зазвичай вводиться ще один додатковий етап, в ході якого, платник дозволяє отримувачу платежу запустити процес оплати. Такий спосіб оплати найбільш поширений при проведенні періодичних

платежів (наприклад щомісячна абонплата за користування певним видом послуг).

Окрім наведеної вище класифікації також розрізняють *автономні* (*off-line*) СЕП та СЕП, що працюють *в режимі реального часу* (*on-line*).

СЕП, що працюють в режимі реального часу мають більше можливостей з точки зору забезпечення безпеки операцій, оскільки дозволяють вести постійне спостереження за умовами роботи і можливими ризиками. Але це передбачає наявність надійної та постійно доступної телекомунікаційної мережі, забезпечити наявність якої можливо далеко не завжди.

Автономні СЕП мають очевидні переваги з точки зору комунікації між учасниками систем, проте при проектуванні таких систем постає інша суттєва проблема: *подвійна витрата коштів* (*double-spending*). Автономність частіше всього означає наявність у користувача системи певного технічного засобу, який зберігає інформацію про доступні для нього кошти. Це означає, що існує теоретична можливість, після проведення оплати, повертати даний засіб у стан, в якому він знаходився до виконання даної операції, дозволяючи тим самим витратити ті самі кошти знову. На практиці застосовують такі способи вирішення даної проблеми або їх комбінацію.

– *Запобігання подвійної витрати* коштів шляхом створення пристроїв, які неможливо повернути до попереднього стану.

– *Виявлення подвійної трати та покарання* винних осіб. Тут виявлення подвійної витрати відбувається постфактум, тому можливі певні труднощі у випадку, якщо подвійна витрата відбулась за допомогою засобу викраденого у його законних власників.

Враховуючи переваги та недоліки обох типів систем, на практиці досить часто використовують комбіновані системи, пристрої яких більшість часу працюють автономно, проте зобов’язані зв’язуватись із центральним сервером у певних випадках, наприклад при перевищені сумою платежу деякого ліміту. Додатково, платіжні пристрої таких систем періодично влаштовують сеанси

зв’язку, в ході яких вони відправляють інформацію про виконані транзакції та отримують оновлення параметрів безпеки.

### 1.1.5 Дематеріалізовані гроші

#### 1.1.5.1 Електронні гроші

*Електронні гроші* (*electronic money*) або *цифрові гроші* (*digital money*) визначаються як грошова вартість, що зберігається на електронному пристрої, який приймається як засіб здійснення платежу особами відмінними від його видавця. Іншими словами, передбачається існування деякого електронного пристрою, який зберігає інформацію про доступні кошти у двійковій формі. Причому, це може бути як обчислювальний пристрій загального призначення, який використовує спеціальне програмне забезпечення для проведення платіжних операцій та мережі для обміну супутньою інформацією (наприклад персональний комп’ютер чи смартфон), так і спеціальний технічний засіб, призначений виключно для зберігання та обробки інформації пов’язаної із виконанням платежів (наприклад смарт-карта).

Одиниці платежу, що зберігаються в картах або в програмному забезпеченні, купуються шляхом внесення готівки на відповідний банківський рахунок. Використання грошей такого вигляду обмежується тими організаціями, які приймають платежі у даній формі.

Електронні гроші можуть бути як *централізованими*, коли існує центральна точка контролю всіх рахунків та проведених транзакцій, так і *децентралізованими*, де зберігання інформації та виконання платіжних операцій виконується розподіленою мережею обчислювальних пристроїв одного рангу. Прикладом децентралізованих електронних грошей можуть

слугувати *криптовалюти*, які за останні кілька років здобули неймовірну популярність по всьому світу.

### 1.1.5.2 Віртуальні гроші

*Віртуальні гроші* (*virtual money*) – це спеціальний тип нерегульованих цифрових грошей, випуск і контроль яких здійснюється їх розробниками, і використовуються представниками спеціальної віртуальної спільноти. Ці гроші не випускаються центральним банком і не обов’язково прив’язані до фіатних грошей, але вони приймаються певними особами як засіб оплати і можуть зберігатись, передаватись або торгуватись у електронному вигляді.

Найбільш яскравим прикладом грошей такого вигляду є віртуальні валюти, які використовуються для покупки та продажу віртуальних товарів чи послуг у таких мережевих спільнотах, як відеоігри, соціальні мережі та ін. Отримати такі гроші можна, наприклад, за допомогою спеціальних сервісів, які обмінюють їх на реальні або електронні гроші чи виконуючи певний, специфічний для відповідного середовища, набір вимог, передбачений його розробниками (наприклад проходження квестів у онлайн-іграх та ін.).

Особливим типом віртуальних гаманців є телефонні карти, що випускаються телефонними компаніями. Це передплачені картки, призначенням яких є оплата послуг телефонного зв'язку відповідного оператора.

### 1.1.6 Властивості систем електронних платежів

**Анонімність.** Дана властивість може бути сформульована по-різному для кожного конкретного випадку. Частіше всього вимагається тільки анонімність платника, хоча є технічна можливість будувати таку СЕП, яка б забезпечувала анонімність отримувача платежу або відразу обох учасників.

У більшості випадків передбачається використання звичайних, не анонімних, рахунків, тому операція зняття готівки або електронних коштів із такого рахунку не є анонімною. Але подальше використання отриманих грошей є анонімним, що означає відсутність прямого зв'язку між особистістю платника і платежами, які ним виконуються.

У разі віддаленої банківської транзакції для забезпечення анонімності платежів необхідно також забезпечувати анонімне спілкування, оскільки після ідентифікації клієнта, наприклад, дзвінка по телефону, будь-які стратегії маскування платежів втрачають сенс.

Анонімність може бути *реальна* та *одностороння*. Перший тип означає досягнення анонімності учасника СЕП від усіх, без виключення, інших учасників, навіть якщо вони кооперуються з метою отримання інформації про нього. Одностороння анонімність є слабшою і гарантує анонімність тільки одного із двох учасників протоколу по відношенню до іншого. При виконанні платежу це, наприклад, означає, що платник є анонімним тільки перед банками, або тільки перед отримувачем (за умови, що вони не кооперуються).

Розглянемо анонімність в контексті банківських карт та електронних гаманців.

– Пластикові карти, які не містять ідентифікатора власника є анонімними. Приклад: телефонні картки. Банківські картки не є анонімними, оскільки на них зазначено номер карти, ім'я власника та відповідний номер рахунку.

– Досягнути анонімності операції перезарядки електронного гаманця можливо, якщо поповнювати його, наприклад, готівкою.

Зазначимо також те, що чим більше учасників, серед яких приховується особа анонімного учасника, тим краще. Якщо коло осіб, які потенційно могли виконати одну й ту саму дію, стає занадто малим, то ризик виявлення зростає. Ідеальним варіантом могла би бути анонімність серед усіх клієнтів одного банку або анонімність серед усіх користувачів певної СЕП. В реальності, досягнення анонімності, зазвичай, можливе тільки серед користувачів, які виконали платіж в деякий часовий проміжок або користувачів, що зняли із свого рахунку монету певного номіналу.

**Невідслідковуваність** означає, що учасник системи не тільки анонімний, але й між проведеними ним операціями, немає жодного логічного зв’язку, який би дозволив визначити належність цих дій до одного й того ж учасника. Наприклад, при використанні смарт-карт, записи про проведені операції зберігаються в «захищеній зоні» для аудиту. Але використання надійного криптографічного алгоритму не дозволить третій особі відслідкувати платежі і пов'язати два різних платежі, виконаних із використанням однієї карти.

У протилежному випадку, кожна окрема дія учасника СЕП може бути анонімною, але сторонній спостерігач має змогу визначити, що декілька дій були здійснені одним і тим же учасником. Навіть система, побудована коректно з технічної точки зору, може допускати викриття особи учасника, через можливість пов’язати його дії за якимись сторонніми, непрямими ознаками. Наприклад, за номерами здійснених телефонних дзвінків телефонна компанія може викрити особу свого анонімного абонента.

Забезпечення гарантій доставки товарів, так само як і можливість вирішення конфліктних ситуацій, суперечить вимозі невідслідковуваності. Питання підтвердження транзакцій є досить складним через те, що законодавство щодо гарантій і конфіденційності має деякі відмінності в різних країнах.

Для досягнення невідслідковуваності часто вдаються до прийому, де кожен учасник СЕП отримує псевдонім, дійсний протягом виконання однієї операції або протягом всього часу взаємодії із системою.

### 1.1.7 Анонімні СЕП, що працюють в режимі реального часу

В подальшому, ми зупинимось на розгляді анонімних СЕП [2], які працюють в режимі реального часу. Розглянемо приклади таких систем за принципом «від простого до складного», завершуючи найбільш складною системою, яка імітує «електронну монету» і базується на використанні спеціальної схеми сліпого цифрового підпису.

#### 1.1.7.1 Анонімні рахунки

СЕП, у яких ведуться анонімні рахунки не пов'язані із реальними особами учасників. Ідентифікатори замінюються псевдонімами, що перешкоджає встановленню особи учасників системи. З іншого боку, банк, що обслуговує систему, не дозволяє знижувати баланс таких рахунків нижче нуля, аби уникнути збитків у випадку «зникнення» анонімних учасників.

Перед тим як розглянути операції СЕП із анонімними рахунками, введемо деякі позначення. Нехай у нас є деяка асиметрична система електронного цифрового підпису $\Sigma = (M, K, S, Sign, Verify)$, де $M$ – множина повідомлень, $K$ – множина ключових пар, $S$ – множина підписів, $Sign: M \times K \to M \times S$ – функція підпису (для ЕЦП із додаванням), $Verify: M \times S \times K \to \{0,1\}$ – функція перевірки підпису. Також позначимо $Cert_{pk}$ сертифікат відкритого ключа $pk$.

Тепер розглянемо власне самі операції.

**Відкриття рахунку**. Щоб відкрити анонімний рахунок, учасник СЕП просто генерує пару ключів $(pk, sk)$ для схеми електронного цифрового підпису і пересилає відкритий ключ $pk$ у банк, який з цього моменту вважається псевдонімом учасника. Банк присвоює створеному рахунку номер $N$ та засвідчує цей факт своїм підписом, додаючи сертифікат свого відкритого ключа: $Sign(\text{"opened"} \parallel N \parallel pk, sk_B)$, $Cert_{pk_B}$.

**Платіж між анонімними рахунками**. Одержувач платежу $R$ повідомляє платнику номер свого рахунку $N_R$. Платник може виконати переказ на рахунок одержувача будь-яким зручним способом, прийнятим в СЕП, наприклад надіслати до банку підписане платіжне доручення з порядковим номером:

$$Sign(\text{"Transfer"} \parallel N \parallel N_R \parallel seq\_no \parallel amount, sk),$$

де $N$ – номер анонімного рахунку платника, $N_R$ – номер рахунку отримувача, $seq\_no$ – порядковий номер платіжного доручення, $amount$ – сума переказу.

Банк перевіряє платіжне доручення, використовуючи відомий йому псевдонім платника, і виконує його.

Хоча така СЕП забезпечує анонімність як платника так і отримувача, степінь анонімності в ній мала, без невідслідковуваності. Проблема стійкості системи до втрати грошових коштів вирішується в мінімальній формі: якщо хто-небудь з учасників втрачає свій секретний ключ $sk$, він втрачає доступ до рахунку, оскільки інших способів довести належність рахунку немає (якщо не вдатись до спеціальних заходів).

### 1.1.7.2 Анонімно переказувані стандартні величини

Термін *стандартні величини* звучить трохи дивно, коли мова йде про системи електронних платежів. Але таким є переклад з англійської мови назви структури даних, що використовується для представлення грошей, запропоноване авторами цієї СЕП (від англійського standard value). Стандартна величина подібна одноразовому рахунку, володіння яким передається через цю величину і дозволяє уникнути відслідковування платежів, характерних для попередньої схеми. Стандартна величина отримала таку назву тому, що являє собою цифровий еквівалент фіксованої кількості грошей, тобто є заздалегідь визначеною, однаковою, «стандартною» величиною. Це випадкове натуральне число, яке можна вважати номером «електронної банкноти», і позначається символом $\nu$. Взагалі стандартні величини можна завжди уявляти собі як цифрові аналоги звичайних паперових грошових банкнот певного номіналу (наприклад, 10, 100 або 1000 грн.), кожна з яких має унікальний серійний номер, що підтверджує її справжність, але не дозволяє при звичайних платежах встановити, ким і кому вона була заплачена і яким був її подальший шлях.

Розглянемо, як в такій системі виконується зняття із рахунку.

1) Щоб отримати стандартну величину, учасник системи $P$ генерує пару ключів $(pk_P, sk_P)$ системи цифрового підпису і пересилає банку підписане секретним ключем повідомлення із проханням видати йому стандартну величину, до якого додає свій відкритий ключ: $Sign(\text{"I want to receive standard value"}, sk_P), pk_P$.

2) Банк перевіряє наявність коштів на рахунку $P$ і, у випадку успіху, направляє йому повідомлення наступного вигляду: $Sign(\nu \parallel \text{"to"} \parallel pk_P, sk_B)$, $Cert_{pk_B}$. З цього моменту учасник $P$ стає власником стандартної величини $\nu$.

Тепер розглянемо виконання платежу:

3) Одержувач платежу $R$ обирає нову пару ключів $(pk_R, sk_R)$. З цього моменту $pk_R$ стає псевдонімом даного учасника. Далі він надсилає платнику $P$ наступне повідомлення: $Sign$ ("Receive ν for purpose Z as $\mathrm{pk_R}$", $sk_R$). Повідомлення підписано ключем, що вже є відомим платнику $P$. Воно необхідно як засіб обережності на випадок розбору можливих конфліктних ситуацій і суперечок про факт отримання грошей учасником $R$.

4) Отримавши попереднє повідомлення платник відправляє банку платіжне доручення: $Sign$("Transfer" ‖ $\nu$ ‖ "$to$" ‖ $pk_R, sk_P$). Для формування підпису використовується ключ $sk_P$, який платник згенерував під час отримання стандартної величини $\nu$.

5) Банк знаходить $\nu$ в базі даних стандартних величин, що були видані учасникам СЕП, і відновлює відкритий ключ $pk_P$ її поточного власника.

6) Банк перевіряє, що отримане від учасника $P$ платіжне доручення коректне і підписано ключем, що відповідає відкритому ключу, знайденому ним на попередньому кроці.

7) Банк переміщає $pk_P$ до списку минулих власників $\nu$ і зберігає платіжне доручення як доказ на випадок можливих суперечок.

8) Банк записує $pk_R$ як відкритий ключ поточного власника стандартної величини.

9) Банк підписує підтвердження переказу і надсилає його платнику: $Sign$("New owner of" ‖ $\nu$ ‖ "$is$" ‖ $pk_R, sk_B$).

10) Платник пересилає повідомлення банку одержувача.

11) Якщо на попередньому кроці одержувач платежу не отримав повідомлення, він може звернутися безпосередньо в банк самостійно і отримати інформацію про здійснений платіж.

12) За необхідності отримувач $R$ може надіслати платнику $P$ квитанцію або відразу надати йому товар або послугу, за яку той заплатив йому гроші.

Якщо $R$ відмовляється від факту проведення платежу, $P$ може використовувати банківське підтвердження з кроку (9) разом з повідомленням

отримувача з кроку (3), аби довести, що він переказав гроші на псевдонім саме цього отримувача.

Як бачимо, описана операція досить громіздка. Депозит стандартної величини на звичайний рахунок виконується операцією, схожою на платіж, з тією відмінністю, що банк вводить себе в якості чергового власника стандартної величини $\nu$, а останній справжній її власник отримує замість переданої банку стандартної величини реальні гроші.

У цій схемі платник і отримувач анонімні, а платежі одного і того ж учасника невідслідковувані. Разом із тим існує певний зв'язок між усіма платежами з однією і тією ж стандартною величиною $\nu$, так само, як це має місце в платежах звичайними паперовими грошима.

### 1.1.7.3 Система електронних платежів Чаума

**Опис системи.** Дана система анонімних електронних платежів була запропонована у 1982 році американським вченим Девідом Чаумом [3] і базується на схемі сліпого цифрового підпису.

*Сліпим підписом* називається такий цифровий підпис, процедура генерації якого не розкриває змісту документу підписуючій стороні. Розглянемо схему сліпого цифрового підпису на основі криптосистеми RSA.

Нехай є абоненти A і B. Абонент B має криптосистему RSA із відкритим ключем $(n, e)$ і приватним ключем $(n, d)$. Абонент A має хоче отримати цифровий підпис документа $M$ ключем абонента B, так аби абонент B не зміг отримати про документ M жодної інформації. Абоненти виконують такі кроки.

1) A обирає випадкове число $r,\ 1 < r < n,\ (r, n) = 1.$

2) A обчислює $k = r^e \ mod\ n,\ M' = kM \ mod\ n$ і передає повідомлення $M'$ абоненту B на підпис. $k$ називається *осліплюючим множником*.

3) B обчислює $S' = Sign\big(M', (n,d)\big) = M'^d \bmod n$ і відправляє $S'$ абоненту A.

4) A обчислює $S = S'r^{-1} \bmod n = M'^d r^{-1} \bmod n = M^d k^d r^{-1} \bmod n = M^d r^{ed} r^{-1} \bmod n = M^d r r^{-1} \bmod n = M^d \bmod n$ – дійсний цифровий підпис повідомлення $M$ ключем абонента B.

Опишемо тепер схему виконання електронного платежу на певну фіксовану суму на основі схеми сліпого підпису RSA. Нехай є абоненти A (платник), B (отримувач платежу) та Банк. Банк має криптосистему RSA із відкритим ключем $(n, e)$ і приватним ключем $(n, d)$. Передбачається, що абоненти та банк спілкуються по захищеному каналу зв'язку (наприклад на основі протоколу SSL/TLS) та виконують взаємну автентифікацію. Також назвемо *монетою* $C = \langle SN, S \rangle$ кортеж, першою компонентою якого є число $0 \le SN < n$, яке ми назвемо *серійним номером* монети, а другою є цифровий підпис серійного номера $S = Sign\big(SN, (n,d)\big)$ виконаний ключем Банку. *Номіналом монети* назвемо суму платежу, який виконується з її допомогою.

Виконуються такі кроки.

1) A випадковим чином генерує серійний номер $0 \le SN < n$.

2) A обирає випадкове число $r$, $1 < r < n$, $(r, n) = 1$ і обчислює $SN' = SN\, r^e \bmod n$.

3) A надсилає на підпис банку осліплений серійний номер $SN'$.

4) Банк генерує підпис $S' = SN'^d \bmod n$ і знімає із рахунку абонента A відповідну суму коштів.

5) A обчислює $S = S'r^{-1} \bmod n = SN^d \bmod n$, отримуючи таким чином дійсний цифровий підпис банку на серійному номері.

6) A формує монету $C = \langle SN, S \rangle$ та надсилає її B. B перевіряє підпис $S$ за допомогою відкритого ключа банку, і, якщо він вірний, надає абоненту A домовлені послуги.

7) B надсилає монету $C$ банку для перерахування коштів на свій рахунок.

8) Банк в свою чергу перевіряє підпис $S$ серійного номера $SN$ монети $C$ і, у разі успішної перевірки, зараховує на рахунок абонента B відповідну суму коштів а також запам'ятовує серійний номер $SN$, аби запобігти спробам подвійного використання.

Таким чином, банк виконує переказ коштів із рахунку абонента A на рахунок абонента B, не маючи при цьому ніякої змоги пов'язати їх між собою, тим самим забезпечуючи анонімність даного платежу.

**Недоліки та способи їх усунення.** Першим недоліком описаної схеми є фіксований номінал кожної монети, що приводить до певних незручностей при спробах виконання платежів на довільні суми. Існує кілька підходів до вирішення даної проблеми, проте найпростішим є наступний. Використовуються монети кількох фіксованих номіналів (наприклад 1, 5, 10, 25 і 100 умовних одиниць) та відповідні їм ключові пари для виконання сліпого підпису з боку Банку. При цьому, номінал включається до складу монети – $C = \langle SN, S, D \rangle$ і повідомляється Банку абонентом A на третьому кроці описаного вище алгоритму виконання електронного платежу. В свою чергу, Банк виконує підпис ключем $(n_D, d_D)$, що відповідає отриманому від абонента A номіналу $D$. В усіх подальших кроках алгоритму перевірка підпису серійного номеру $SN$ проводиться із використанням відповідного відкритого ключа $(n_D, e_D)$.

Іншим недоліком схеми електронних платежів Чаума є неможливість передавати монету $C$ від одного абонента системи до іншого без взаємодії із Банком. Це пов'язано із тим, що єдиним засобом захисту системи від подвійної трати монети є реєстр використаних монет зі сторони Банку. Уявімо, що абонент A отримав від банку підписану монету $C$ і виконує з її допомогою оплату одночасно абонентам X та Y. Якщо ми дозволяємо абонентам передавати монети іншим абонентам без взаємодії із банком, то жоден із отримувачів монети $C$ не буде в змозі виявити шахрайство з боку абонента A. Єдиним способом захисту є надання послуг абоненту A тільки після

повідомлення від Банку про успішний обмін монети $C$ на відповідну її номіналу суму коштів. В такому випадку спроба використання однієї і тієї ж монети $C$ для проведення оплати одночасно двом користувачам $X$ та $Y$ завершиться успіхом першої операції та невдачею другої з них.

Таким чином, абоненти не можуть зберігати отримані монети для проведення платежів іншим абонентам. Оскільки Банк зобов'язаний бути учасником кожного платежу створюється додаткове навантаження на його програмне та апаратне забезпечення. Більш того, Банк повинен бути постійно доступним, що значно ускладнює підтримку його внутрішніх систем.

Одне із рішень даної проблеми було запропоновано у роботі [4]. Суттєвим його недоліком є використання досить складних математичних конструкцій, що ускладнює його реалізацію та подальший аналіз його властивостей.

Інше рішення, засноване на використанні технології блокчейн було запропоновано у роботі [5]. Серед недоліків даної роботи необґрунтоване використання деяких складових, що призводить до ускладнення запропонованої системи та підвищення накладних витрат пов'язаних із її функціонуванням.

Метою даної роботи є модифікація системи електронних платежів Чаума таким чином, аби дозволити передачу монет між абонентами системи, забезпечуючи анонімність усіх операцій та неможливість подвійного використання монет.

## 1.2 Смарт-контракти і технологія блокчейн

### 1.2.1 Біткоїн

#### 1.2.1.1 Передумови виникнення

Переважна більшість систем електронної готівки створених протягом XX і на початку XXI ст. покладались на існування центральної довіреної особи, яка зберігала інформацію про рахунки всіх користувачів та про всі зареєстровані в системі транзакції. Одна із причин переважання систем такого виду – відсутність інших засобів вирішення проблеми подвійної витрати, яка буде згадуватись далі. Проте наявність центральної довіреної особи, що контролює всю інформацію системи робить її вразливою, тому активізувались дослідження, присвячені розробці децентралізованих систем електронних грошей.

Перша така система під назвою *біткоїн* (*BitCoin*), була запропонована у 2008 році розробником під псевдонімом Сатоші Накамото (Satoshi Nakamoto), який опублікував роботу із заголовком «BitCoin: A Peer-to-Peer Electronic Cash System» [6]. Мережа біткоїн запрацювала у 2009 році і з тих пір знайшла чимало прихильників по всьому світу.

Окремої уваги заслуговує, закладена в основу біткоїна, технологія *блокчейн* (*blockchain*). Вона дозволила вийти за межі децентралізованих систем електронної готівки і дала життя системам децентралізованого зберігання та обробки інформації, мова про які йтиме далі у даному розділі.

### 1.2.1.2 Основні складові та принципи роботи

**Адресація та гаманці.** Одним із основних будівельних блоків криптовалюти біткоїн є електронний цифровий підпис (далі ЕЦП). Розробниками системи було прийнято рішення використовувати ЕЦП на основі еліптичних кривих (ECDSA) із кривою $secp256k1$. Довжина секретного ключа такого алгоритму складає 256 біт, а відповідного йому відкритого – 512 біт [7].

Для того аби почати працювати із системою користувач повинен згенерувати собі пару із відкритого та секретного ключа $(pk, sk)$, що виконується на обчислювальному пристрої користувача і не потребує підключення до мережі. Згенеровані таким чином ключові пари додаються до спеціального файлу, який називається *гаманцем*. Гаманець може містити довільну кількість ключових пар і зберігається у файловій системі в зашифрованому вигляді.

Після цього, відкритий ключ $pk$ використовується для генерації *адреси*, асоційованої із даною ключовою парою. Адреса має довжину 160 біт і обчислюється як:

$$A = RIPEMD160\big(SHA256(pk)\big),$$

де $pk$ – відкритий ключ користувача, $RIPEMD160$ та $SHA256$ – геш-функції із довжиною виходу 256 біт та 160 біт відповідно, $A$ – відповідна адреса.

Секретний ключ $sk$, в свою чергу, використовується для авторизації транзакцій, виконуваних від імені користувача із адресою $A$. Загалом, користувач, як особа, може мати довільну кількість ключових пар та відповідних їм адрес, але для простоти викладення надалі в даному тексті під словом «користувач» ми будемо розуміти одну певну біткоїн адресу, власником якої він є, якщо не сказано інше.

**Транзакції.** Транзакції є найбільш важливою частиною системи біткоїн. Решта системи проектувалась так, аби забезпечити створення транзакцій, їх розповсюдження по мережі, перевірку і, нарешті, їх додавання до глобального реєстру – блокчейна.

*Транзакцією* називається структура даних, яка містить інформацію про переведення певної суми біткоїнів від джерел коштів, які називаються *входами* (*input*), до місць їх призначення, які називаються *виходами* (*output*) (рисунок 1.4).

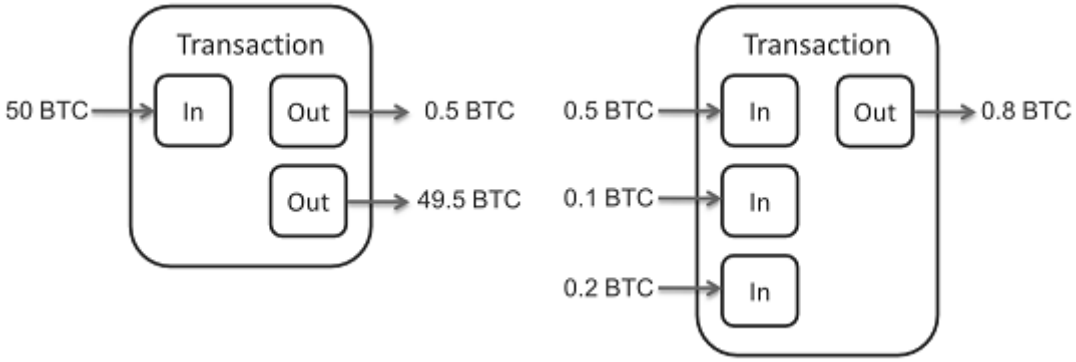


**Рисунок 1.4** – Біткоїн транзакція

Фундаментальним поняттям, закладеним в основу біткоїн транзакцій є невикористаний *вихід транзакції* (*unspent transaction output*) або НВТ (UTXO). НВТ є неподільною грошовою величиною, яка належить конкретному користувачу, записана до блокчейна і визнається платіжною одиницею іншими учасниками мережі. Поточний капітал користувача формується сукупністю його НВТ, отриманих ним в результаті успішно виконаних транзакцій. Таким чином, баланс користувача не є числом, що зберігається у блокчейні. Баланс є сумою всіх НВТ, що йому належать.

Входи транзакції, по суті, є НВТ, що належать користувачу, який виконує дану транзакцію. Кожен НВТ може бути використаний у якості входу тільки один раз, оскільки після цього, власне, він перестає бути «невикористаним».

Кожен вихід транзакції складається із суми біткоїнів та адреси користувача, у власність якому ця сума передається. Кожен вихід транзакції створює НВТ для користувача, чия адреса у даному виході зазначена.

Частіше всього, до виходів додається один спеціальний вихід, що називається *комісією* (*transaction fee*), сутність якого буде розкрито у розділі про майнінг.

Сума входів та сума виходів транзакції обов'язково має співпадати, оскільки кожен включений до переліку входів НВТ має бути повністю використаний даною транзакцією. Оскільки досить рідко сума наявних у користувача НВТ буде точно дорівнювати сумі платежу, який він хоче виконати, більшість транзакцій мають такий вихід як *решта* (*change*), адреса якого встановлюється рівною адресі користувача, що виконує транзакцію і дозволяє повернути невикористані кошти назад до його власності. Таким чином, невикористані кошти формують новий НВТ і знову стають доступними для використання у подальших транзакціях.

Для підтвердження того, що транзакція була дійсно створена користувачем, що є власником усіх використаних в ній НВТ, до транзакції додається цифровий підпис ключем $sk$, що відомий тільки даному користувачу.

Підсумовуючи, у дещо спрощеному вигляді, можна сказати, що кожна транзакція містить наступні складові:

– один чи більше входів

– один чи більше виходів

– цифровий підпис користувача

**Виконання транзакцій.** Транзакція, враховуючи включену до неї інформацію, можна вважати «наміром» користувача передати право власності на свої кошти деяким іншим користувачам. Аби цей «намір», підтверджений цифровим підписом користувача, набрав чинності його потрібно передати іншим користувачам мережі. Проте отримувачі платежу повинні мати певні

гарантії того, що платник не спробував використати свої НВТ двічі, відправивши різні за складом транзакції різним групам користувачів мережі.

Класичним рішенням даної проблеми є введення деякої *довіреної центральної сутності*, яка містить інформацію про всі виконані транзакції і може перевіряти їх на факт здійснення спроб подвійної витрати коштів. Першу транзакцію, яка використала певний НВТ у якості входу ми будемо визнавати дійсною, а всі інші будемо відхиляти. Проблема даного підходу в тому, що доля всієї системи залежить від компанії, яка виконує дану роль. Наша мета – це побудова децентралізованої альтернативи.

У децентралізованому середовищі єдиний спосіб запобігання спроб подвійного використання це знати про всі виконані у системі транзакції. Аби певну транзакцію можна було обрати за першу серед усіх, що містять один і той же НВТ, необхідно мати інструмент, який дозволить усім учасникам розподіленої системи досягати консенсусу і виробляти спільну для всіх них історію транзакцій.

**Блокчейн.** Засобом, який дозволив вирішити проблему досягнення консенсусу з приводу спільної історії транзакцій став блокчейн.

Блокчейн (рисунок 1.5) – це структура даних, що являє собою впорядкований зв’язний список *блоків*, кожен із яких містить посилання на попередній блок, який називається *батьківським* (*parent block*). По відношенню до батьківського блока, даний блок називається *дочірнім* (*child*). Кожен блок складається із *заголовка* (*header*) і списку включених до нього транзакцій [7].

Кожен блок у ланцюгу однозначно ідентифікується геш-значенням, обчисленим за допомогою геш-функції $SHA256$. Посилання на попередній у ланцюгу блок реалізується включенням геш-значення заголовка попереднього блока до заголовка даного блока.

Очевидно, що якби кожен користувач додавав блоки довільним чином, то із часом блокчейн втратив би структуру ланцюга і обрати серед множини гілок ту, якій можна довіряти було б неможливо. Надійність у такій системи

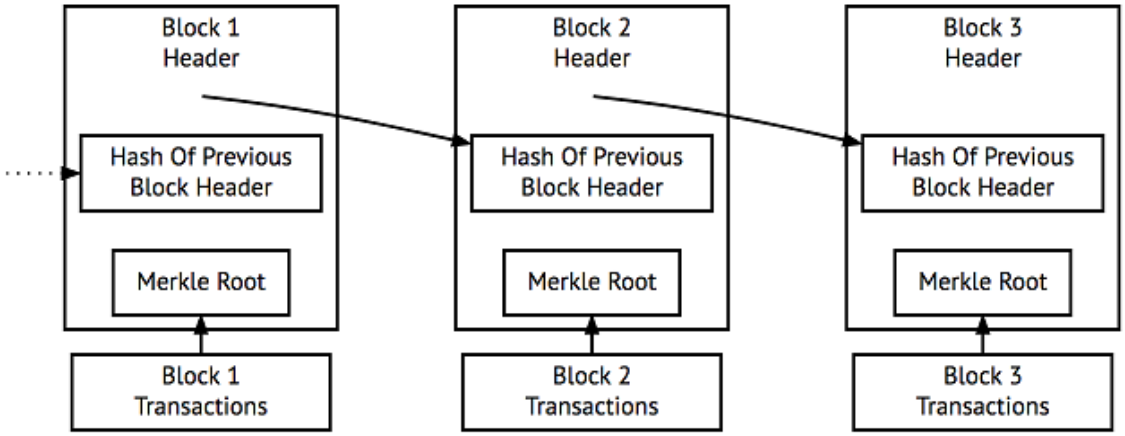


**Рисунок 1.5** – Спрощена структура біткоїн блокчейна

заснована на тому, що новий блок неможливо сформувати швидше (в середньому), ніж за певний проміжок часу, а саме за 10 хвилин у системі біткоїн. Перевірка коректності блоку, при цьому, повинна відбуватись швидко.

Механізм, який забезпечує дотримання даної умови у біткоїні було ґрунтується на виконанні великої кількості обчислень і отримав назву *Proof-of-Work* (*PoW*). Для його підтримки до заголовка блоку було додано спеціальне поле $nonce$, яке може містити довільне 4-байтове значення. Ідея полягає в наступному. Нехай задано деякий параметр мережі $t$, який називається *ціль* (*target*), $0 < t < 2^{256}$. Тоді, розглядаючи геш-значення блоку як число $h$, $0 \leq h < 2^{256}$, задача полягає в тому, аби підібрати таке значення поля $nonce$, що буде виконана умова $h < t$.

Генерація нових блоків називається *майнінг* (*mining*) і виконується користувачами мережі, які, відповідно, називаються *майнерами* (*miner*). Загальна процедура складається із таких кроків.

1) Обрати серед незафіксованих транзакцій ті, які помістяться в один блок.

2) Заповнити ті поля заголовка блоку, значення яких визначаються однозначно (всі, окрім $nonce$).

3) Встановити $nonce$ в деяке значення, наприклад 0.

4) Виконувати в циклі:

а) Включити поточне значення $nonce$ у блок.

б) Порахувати значення $h = SHA256(block\_header)$.

в) Якщо $h < t$, то вважаємо блок сформованим, інакше збільшуємо значення $nonce$ на 1 і повторюємо цикл.

5) Розіслати сформований блок іншим вузлам мережі.

Існує ймовірність того, що відразу декілька майнерів одночасно сформують коректний блок і розповсюдять його по мережі, тому у ланцюга на деякий час може з'явитись розгалуження. Тоді кожен майнер може обрати будь-який із двох блоків і намагатись згенерувати новий блок на його основі. Врешті-решт, одна із гілок стане довшою (та, яка була підтримана більшістю майнерів) і буде визнана основною. Всі транзакції зафіксовані в іншій гілці в цей момент переходять назад в пул незафіксованих транзакцій і повинні чекати затвердження знову.

За свою роботу майнери отримують винагороду у вигляді деякої наперед визначеної кількості біткоїнів, які переходять до них у власність у вигляді НВТ за кожен успішно згенерований блок. Додатково, до винагороди майнера включається сума усіх комісії, оголошених авторами транзакцій, що були включені до згенерованого блоку. Це означає, що серед усіх транзакцій, які очікують на включення до блокчейна, більша ймовірність бути обраними і включеними у тих, які оголосили більшу комісію.

Що стосується значення цілі $t$, то воно постійно адаптується у відповідності до поточних обчислювальних ресурсів мережі таким чином, аби середній час генерації блоку підтримувався на рівні 10 хвилин на блок.

Отже, ми бачимо, що, в запропонованій структурі блокчейна, геш попереднього блока впливає на значення поточного блока, а отже і на геш-значення його заголовка. Це означає, що зміна значення будь-якого поля у будь-якому блоці у ланцюгу призведе до зміни всіх блоків, які були додані до блокчейна після нього. В свою чергу, це означає, що для того, аби інші

учасники мережі визнали модифіковану версію ланцюга дійсною нам потрібно для кожного зміненого блока вирішити задачу PoW, що займає в середньому 10 хвилин на блок. Окрім того, довжина нашого ланцюга повинна стати більшою ніж довжина основного ланцюга аби решта учасників мережі прийняла його за основний. Звідси слідує, що для перепису історії блокчейна, потенційний зловмисник повинен зосередити у себе більше ніж 50% усіх обчислювальних ресурсів мережі – це є єдиною потенційною, проте, поки не досяжною на практиці, вразливістю методу PoW.

**Дерева Меркля.** Одна із оптимізацій, використовуваних у структурі блокчейна біткоїн, пов'язана із використанням *Дерев Меркля* або *Двійкових Дерев Гешів*. Деревом Меркля називається структура даних, яка являє собою двійкове дерево, вузлами якого є геш-значення. Вони використовуються у біткоїні для узагальнення всіх транзакцій блоку, виробляючи загальний «відбиток» всього набору транзакцій і надаючи можливість ефективної перевірки входження певної транзакції у даних блок.

Дерево Меркля будується знизу вгору. Припустимо, що у нас є чотири транзакції $A$, $B$, $C$ і $D$. *Листками* (*leaves*) дерева стають їх геш-значення $H_A$, $H_B$, $H_C$ та $H_D$, де, наприклад, $H_A = SHA256\big(SHA256(A)\big)$.

Вузли кожного наступного рівня обчислюються як подвійне значення геш-функції від конкатенації значень дочірніх вузлів, наприклад $H_{AB} = SHA256\big(SHA256(H_A \parallel H_B)\big)$ – батьківський вузол для вузлів $H_A$ та $H_B$.

Нарешті *коренем дерева Меркля* (*Merkle root*) для описаного прикладу стає значення $H_{ABCD} = SHA256\big(SHA256(H_{AB} \parallel H_{CD})\big)$, яке й включається до заголовка блоку (рисунок 1.6).

Якщо на певному рівні для вузла не має пари, яка може бути використана, для обчислення геш-значення наступного рівня, то значення цього вузла переноситься на наступний рівень без змін.

Використання Дерева Меркля для узагальнення інформації про транзакції, які включаються до певного блоку має кілька суттєвих переваг.

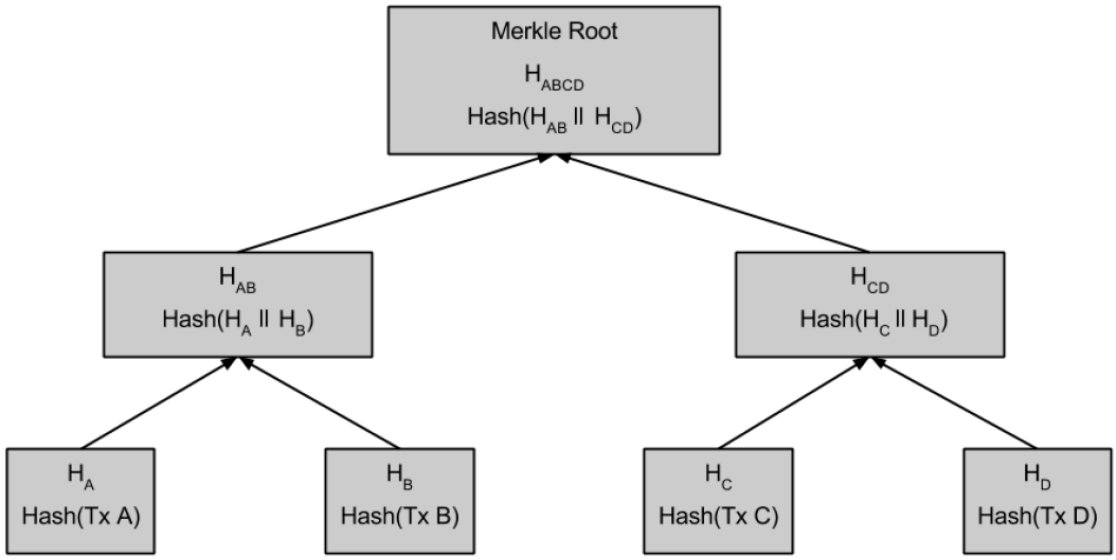


**Рисунок 1.6** – Дерево Меркля, що включає 4 транзакції

– **Перевірка входження транзакції у блок.** Продовжуючи наведений приклад, нехай нам потрібно перевірити чи входить транзакція $C$ у блок. Оскільки у нас є попередньо пораховані геш-значення усіх проміжних вузлів, то нам необхідно повторно обчислити тільки значення $H_{CD}$ та $H_{ABCD}$ і порівняти останнє із значенням, що зберігається у заголовку блоку. Узагальнюючи, для перевірки чи входить певна транзакція у блок, який містить інформацію про $N$ транзакцій необхідно виконати $O(\log_2(N))$ операцій обчислення значення геш-функції $SHA256$.

– **Зберігання інформації про транзакції.** Зберігання повної історії всіх проведених транзакцій із часом призводить до використання надмірної кількості дискового простору. Як тільки всі НВТ, створені певною транзакцією $T$, перетворюються на входи інших транзакцій і всі вони надійно фіксуються у блокчейні, зберігання транзакції $T$ більше не потребується. Якби заголовок блоку зберігав усі транзакції явно, то, очевидно, стирання інформації про одну із них призводило б до пошкодження блоку, що не є припустимим. Проте, узагальнення транзакцій у Дереві Меркля дозволяє поступово звільняти простір, що зайнятий тими гілками, які відносяться до застарілих транзакцій. Поступово можна дійти до того, що необхідним буде зберігання тільки кореня дерева, яке використовує всього 32 байти дискового

простору, незалежно від кількості транзакцій, використаних для його формування.

### 1.2.2 Платформа Ethereum

#### 1.2.2.1 Зародження ідеї

Незадовго після появи криптовалюти біткоїн стало зрозуміло, що потенціал закладеної в її основу технології блокчейн є куди більшим, ніж просто зберігання історії фінансових транзакцій. Так засновник періодичного видання Bitcoin Magazine Віталік Бутерін помітив, що внутрішній стан блоків блокчейна можна розширити для зберігання в ньому довільної інформації, а прості правила обробки грошових коштів можна замінити довільним програмним кодом, який буде публікуватись в блокчейн розробниками. У 2013 році він запропонував платформу *Ethereum,* яка об'єднувала всі запропоновані ідеї, а вже у 2015 році відбувся її запуск.

#### 1.2.2.2 Принципи смарт-контрактів

*Смарт-контракт* (*smart contract*) – це комп'ютерний протокол, призначений для того, щоб у цифровій формі сприяти, перевіряти та приводити в дію виконання *контрактів*. Під контрактом, в даному випадку, розуміється набір обіцянок, даних сторонами одна одній. Смарт контракти повинні забезпечувати прозоре та незворотне виконання зазначених у них домовленостей.

Дана ідея та визначення були висловлені ще у 1996 році Ніком Сабо – американським вченим у сфері криптографії та комп'ютерних наук [8]. Довгий час не існувало достатніх засобів для втілення даної ідеї у життя, проте з виникненням технології блокчейн ситуація змінилась.

Деякі принципи смарт-контрактів були втілені ще у біткоїні – першій криптовалюті, побудованій на його основі. До кожної транзакції можна додати сценарій, написаний спеціальною мовою програмування, умови, зазначені в якому, мають були виконаними, аби отримувач транзакції зміг отримати доступ до переданих йому коштів. Проте використовувана для їх написання мова програмування Script не є повною по Т'юрингу, тому має досить обмежені можливості і рідко застосовується на практиці.

Кілька років потому, реалізувати ідею смарт-контрактів у повній мірі нарешті вдалось розробникам системи децентралізованого зберігання та обробки інформації Ethereum, в основу якої, знову ж таки, було покладено технологію блокчейн.

#### 1.2.2.3 Реалізація смарт-контрактів в Ethereum

**Будова смарт-контракта.** Смарт-контракти описані Ніком Сабо мали дещо абстрактний вигляд, тому будь-яка їх практична реалізація вимагала деякого переосмислення і конкретизації даної концепції. У Ethereum смарт-контрактом називається комп'ютерна програма яка виконується в контексті *віртуальної машини Ethereum* (*Ethereum Virtual Machine* – EVM) [9]. Слово «контракт» не має жодного юридичного значення в даному контексті.

Розробка смарт-контрактів виконується із використанням спеціально розроблених мов програмування. Найбільш популярною на даний момент є об'єктно-орієнтована мова програмування Solidity [10]. Дана мова за своїм синтаксисом дуже схожа на іншу мову програмування JavaScript, яка є однієї

із основних мов програмування у сфері веб-розробки. Така схожість суттєво понижує вхідний бар'єр для розробників, що зацікавились розробкою смарт-контрактів, оскільки дозволяє використовувати уже наявні навички програмування.

У термінах об'єктно-орієнтованого програмування, смарт-контракт є об'єктом, що складається із двох умовних частин:

– стану (state) – даних, які зберігає даний об'єкт

– методів – блоків коду, які використовуються для оновлення стану

Використання високорівневої мови програмування полегшує розробку, проте для того аби виконуватись в EVM, смарт-контракт повинен бути перетворений у зрозумілу для неї форму, тобто бути скомпільованим у низькорівневий байт-код. Після компіляції смарт-контракт необхідно опублікувати у блокчейні Ethereum. Публікація здійснюється виконанням транзакції за спеціальною адресою. Після цього, смарт-контракт отримує власну адресу, яка може використовуватись іншими користувачами для взаємодії із ним.

Надалі під *викликом смарт-контракту* будемо розуміти виклик певного методу даного смарт-контракту. Код смарт-контракту виконується тільки, якщо він був викликаний певною транзакцією. Важливо розуміти, що виконання смарт-контракту повинно бути ініційованим *зовнішнім користувачем* (*externally owned account* – EOA). Смарт-контракт може бути викликаний іншим смарт-контрактом, який в свою чергу був викликаний іншим смарт-контрактом, проте на початку цього ланцюга викликів повинен бути виклик, виконаний зовнішнім користувачем. Смарт-контракти не можуть виконуватись у фоновому режимі або за певним розкладом.

Транзакції [11] (рисунок 1.6.) є *атомарними* (*atomic*), незалежно від того скільки смарт-контрактів було в ній задіяно або що робили смарт-контракти під час викликів. Всі зміни виконані в ході транзакції стають видимими іншим користувачам тільки, якщо транзакція завершилась успішно.

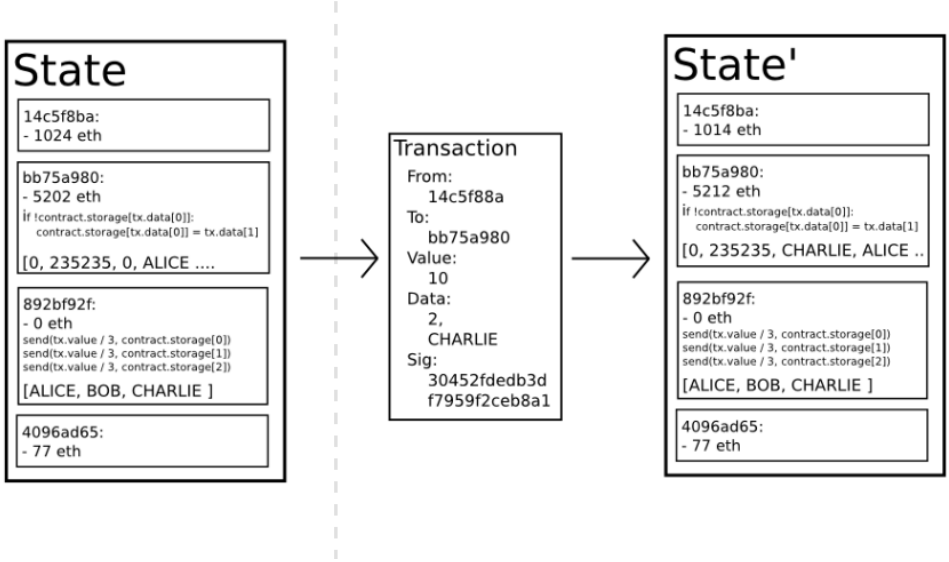


**Рисунок 1.6** – Схематичне зображення транзакції Ethereum

Смарт-контракти є *незмінюваними* (*immutable*). Це означає, що як тільки код смарт-контракту було опубліковано в блокчейні, його більше неможливо змінити. Проте смарт-контракт можливо «видалити» із блокчейна, видаляючи його код і внутрішній стан. Варто розуміти, що це не призведе до видалення історії транзакцій пов’язаних із даним смарт-контрактом, оскільки вони зберігаються у блокчейні, який, в свою чергу, є незмінним. Проте, це дозволить звільнити дисковий простір, відведений для зберігання даних, що відносяться до даного смарт-контракту і зробить його недоступним для виклику у подальших транзакціях.

**EVM.** EVM – це частина протоколу, яка, власне, займається підтримкою внутрішнього стану смарт-контрактів і виконанням обчислень. EVM розуміє спеціальну форму машинного коду, що називається *EVM байт-код*, специфікацію якого можна знайти у офіційній документації Ethereum.

EVM можна розглядати як глобальний децентралізований комп’ютер. В дійсності, EVM працює локально на комп’ютері кожного користувача мережі Ethereum, обробляючи дані, які знаходяться на цьому ж комп’ютері або приходять із мережі. Проте, завдяки використанню блокчейна для зберігання стану EVM ми можемо бути впевненими в тому, що всі вузли, рано чи пізно, матимуть однакову копію всіх даних.

В зв’язку із тим, що EVM є глобальним комп’ютером, увесь код смарт-контрактів повинен бути детермінованим, аби його виконання завжди давало однаковий результат, незалежно від того на якому комп’ютері було виконано обчислення. Для забезпечення даної властивості можливості смарт-контрактів обмежуються маніпуляціями внутрішнім станом і переданими аргументами та взаємодією з іншими смарт-контрактами. Будь-яка взаємодія із зовнішнім оточенням, таким як мережа або файлова система, у смарт-контрактах заборонена. З тих же причин мова Solidity не має вбудованих засобів генерації псевдовипадкових чисел.

**Гас.** Виконання коду потребує використання обчислювальних ресурсів користувачів мережі. Оскільки код смарт-контрактів виконується на комп’ютерах всіх користувачів мережі, то виникає загроза проведення DoS-атак, шляхом публікації смарт-контрактів, методи якого містять великий об’єм обчислень, і викликом цих методів. В зв’язку із цим, розробникам Ethereum необхідно було передбачити засоби захисту від подібний дій користувачів.

Ідея захисту полягає у введені ціни, яку користувач повинен сплатити за виклик того чи іншого методу смарт-контракту. Об’єм обчислень, який необхідно провести для виконання методу смарт-контракту оцінюється у абстрактних одиницях, які називаються *гасом* (*gas*). Гасом не можна володіти чи витрачати його, він існує тільки всередині EVM для оцінки кількості роботи. Оплата стягується у основній валюті мережі яка називається *ефіром* (*ether*). Вона служить як додаткова комісія за проведення транзакції і зараховується на рахунок майнера, якому вдалось сформувати блок, що включає в себе дану транзакцію. Окрема одиниця була введена з метою розділення вартості обчислень, яка залежить тільки від коду смарт-контракту, і ціни ефіру, яка формується ринком і не є сталою.

Оцінка гасу може бути неточною, оскільки вона може залежати від параметрів виклику того чи іншого методу. Тому насправді оплата відбувається наступним чином. Перед відправкою транзакції, користувач зобов’язаний вказати значення двох параметрів:

– *лімітом газу* (*gas limit*) – максимальна кількість гасу, яку він згоден витратити на виконання даної транзакції

– *вартості газу* (*gas price*) – ціна одиниці гасу, яку він згоден сплатити

Ефір у кількості $gas\ limit \times gas\ price$ знімається з рахунку користувача перед початком виконання транзакції, аби запобігти ситуації «банкрутства» користувача під час виконання транзакції. З цієї причини, користувачі не можуть вказати ліміт гасу, який перевищує їхній баланс.

Якщо в ході виконання транзакції виявилось, що кількість гасу необхідна для її завершення виявилась більшою, ніж ліміт, вказаний користувачем, то транзакція скасовується, а витрачений гас не повертається назад на рахунок користувача, оскільки майнери уже провели певну кількість обчислень до того моменту і заслуговують на отримання компенсації.

Якщо вказана кількість виявилась більшою на дійсно необхідну, то різниця повертається назад на рахунок користувача, а майнерам компенсується тільки кількість дійсно виконаної роботи.

Оскільки автор транзакції сам встановлює ціну одиниці гасу, яку він згоден сплатити на користь майнера, він може впливати на шанси своєї транзакції бути включеною до наступного блоку, оскільки майнери віддають перевагу транзакціям із більшою комісією.

#### 1.2.2.4 Альтернативні протоколи досягнення консенсусу

Із ростом популярності криптовалют все більше людей та обладнання долучалось до процесу майнінгу з метою отримання якнайбільшого прибутку у якомога менший проміжок часу. За останні декілька років сумарне споживання електроенергії усіма обчислювальними пристроями, що використовуються для майнінгу стало надто великим. Ще у 2012 році сумарна потужність біткоїн мережі перевищила найпотужніший у світі

суперкомп'ютер. В зв'язку з цим постало досить серйозне питання пошуку альтернативи, яка виконувала б роль аналогічну до PoW, але не призводила до марнування такої кількості електроенергії.

Першим альтернативним протоколом досягнення консенсусу став *Proof-of-Stake* (*PoS*), який був запропонований у 2011 році і вперше реалізований у криптовалюті PPCoin у 2012 році [12].

Загальна ідея полягає в тому, що обмежений ресурс, яким можна голосувати, можливо знайти не тільки у зовнішньому світі (як електроенергія у випадку із PoW), а й всередині самої системи – власне валюта даної системи.

Для формування блоку у системі на основі PoS майнеру все ще необхідно підібрати таке геш-значення $h$ заголовка блоку, яке буде менше ніж встановлена ціль $t$. Різниця із протоколом PoW полягає в тому, що:

1) До заголовка блоку замість значення $nonce$ майнер повинен включити один зі своїх НВТ.

2) Ціль $t$ встановлюється не глобально для всієї мережі із урахуванням її загальної обчислювальної потужності, а залежить від НВТ, який майнер обрав для формування блоку.

Більш формально, нехай майнер обрав НВТ $utxo_A$. Тоді ціль $t$ обчислюється наступним чином:

$$t = d_0 \times coins(utxo_A) \times timeweight(utxo_A),$$

де:

– $d_0$ – глобальна константа, яка регулюється так, аби середній час генерації блоку був на рівні 10 хвилин

– $coins(utxo_A)$ – кількість ефіру НВТ $utxo_A$

– $timeweight(utxo_A)$ – час, який пройшов с моменту включення у блокчейн попереднього блоку, згенерованого із використанням НВТ $utxo_A$

Очевидно, що більша ціль (і, відповідно, більша ймовірність згенерувати блок) буде для НВТ, який містить більше монет і який останній раз використовувався для формування іншого блоку досить давно. Таким чином,

відразу після успішної генерації блоку, НВТ «блокується» на деякий час і не може бути використаним для формування інших блоків. Так досягається обмеженість даного ресурсу.

Таким чином, кількість спроб на формування нового блоку кожного майнера обмежена кількістю наявних у нього НВТ, що, зазвичай, не є дуже великим числом. Оскільки перебір відбувається на обмеженій кількості варіантів, то він майже не залежить від потужності обчислювального обладнання і не використовує такої кількості електроенергії як PoW.

Проте і PoS має деякі суттєві недоліки, які перешкоджають його масовому провадженню до існуючих блокчейн систем [13]. Справа в тому, що із PoS стає можливо і навіть вигідно майнити одночасно кілька гілок блокчейна. Із PoW це стає неможливо, оскільки якщо ви витрачаєте ресурси на майнінг одної гілки, то на іншу ресурсів не лишається, тому ваш середній дохід від майнінгу залежить тільки від ваших ресурсів. У PoS майнінг не потребує проведення суттєвої кількості обчислень, тому можливо майже «безкоштовно» майнити одночасно декілька гілок блокчейну, підвищуючи тим самим свої шанси на отримання винагороди.

#### 1.2.2.5 Застосування блокчейну та смарт-контрактів

За недовгий час свого існування Ethereum і, закладена в його основу, технологія блокчейн знайшли чимало прихильників, що призвело до їх широкого розповсюдження і пошуку нових способів їх застосування. На основі блокчейну було створено кілька десятків альтернативних криптовалют, хоча варто зауважити, що біткоїн досі є найбільш популярною із них і не збирається у найближчому майбутньому поступатись цим місцем. Крім того, відомі спроби побудови систем розподіленого зберігання даних та

електронного голосування на основі блокчейна, які потенційно здатні замінити існуючі централізовані аналоги.

Можливість використання контрактів, контроль за виконанням яких здійснюється автоматично і не покладається на третю особу створює нові можливості для бізнесу. Оскільки всі умови вказані у смарт-контракті, збереженому у блокчейні, то ви спокійно можете вести справи із незнайомими людьми і не хвилюватись з приводу їх виконання.

У мережі Ethereum вже опубліковано сотні смарт-контрактів, що забезпечують роботу такої ж кількості застосунків на їх основі. Можливість писати довільний код, причому на мові програмування, яка дуже схожа на більшість найпопулярніших на даний момент мов, означає, що розробка нових ідей застосування смарт-контрактів обмежена тільки уявою розробників.

Досить показовим є і той факт, що останнім часом все більше урядових установ у багатьох країнах світу повідомляє про свої плани перенесення існуючих систем зберігання і обробки інформації на блокчейн. Особливо приємним є те, що серед таких країн Україна є одним із найбільш активних учасників.

## Висновки до розділу 1

Розробка і подальша експлуатація систем електронних платежів мають суттєвий вплив як на розвиток бізнесу так і на життя кожної окремої людини. З точки зору бізнесу, відбувається автоматизація процесів, проведення яких раніше потребувало суттєвих витрат часу та коштів. Це дозволяє бізнесу зменшити накладні витрати, сфокусуватись на вирішенні основних задач і розвиватись більш швидкими темпами. З точки зору звичайних людей, СЕП також встигли стати невід’ємною складовою життя, оскільки дозволяють

надзвичайно спростити та пришвидшити операції покупки і продажу товарів та послуг, а також переведення коштів між рахунками.

У даній роботі розглядаються анонімні СЕП, що працюють в режимі реального часу. СЕП Чаума є одним із найбільш відомих представників даного типу систем, одним із найсуттєвіших недоліків якої є неможливість передачі монет між користувачами. Дана робота присвячена вивченню та вдосконаленню методів вирішення даної проблеми.

«Блокчейн» є однією із найбільш багатообіцяючих технологій останнього десятиріччя, що підтверджується широким колом її прихильників та досить розвиненою екосистемою. Одним із можливих застосувань даної технології є побудова протоколів електронних платежів на її основі, що стає можливим завдяки можливостям, які вона пропонує та властивостям які забезпечує.

У даній роботі технологію смарт-контрактів Ethereum планується використати для вдосконалення системи електронних платежів Чаума таким чином, аби забезпечити можливість безпечної та анонімної передачі монет між користувачами.

# 2 ВИРІШЕННЯ ЗАДАЧІ ЗАБЕЗПЕЧЕННЯ МОЖЛИВОСТІ БЕЗПЕЧНОЇ ПЕРЕДАЧІ МОНЕТ МІЖ КОРИСТУВАЧАМИ ЗА ДОПОМОГОЮ ТЕХНОЛОГІЇ БЛОКЧЕЙН

Даний розділ містить детальний опис усіх процедур та алгоритмів, що лежать в основі запропонованого рішення проблеми передачі монет системи електронних платежів Чаума між користувачами. Крім того, наводиться аналіз властивостей отриманого рішення та можливі напрямки подальших досліджень.

## 2.1 Загальна схема запропонованого рішення

Пропонований метод вирішення даної задачі базується на ідеї використання стану смарт-контракту для зберігання інформації про поточного власника монети і його методів для безпечної передачі монет між користувачами. Життєвий цикл монети складається із наступних етапів:

– Генерація монети та реєстрація монети у смарт-контракті

– Передача права власності на монету між користувачами

– Зарахування монети на банківський рахунок та її виведення із обігу

Перший етап відбувається поза блокчейном, в той час як усі інші нерозривно із ним пов’язані. Далі у даному розділі наводиться детальний опис усіх етапів життєвого циклу монети і описується будова смарт-контракту, що лежить в основі запропонованої системи. Вихідний код прототипу даного смарт-контракту наводиться у Додатку А.1.

**Генерація та реєстрація монети.** Генерація монети проводиться у спосіб аналогічний класичній системі електронних платежів Чаума. Абонент Банку генерує монету та отримує у Банку цифровий підпис її серійного номера за допомогою протоколу сліпого підпису.

Після генерації монети, абонент використовує свій обліковий запис мережі Ethereum для внесення у смарт-контракт запису, який містить саму монету та адресу її власника. Внесення інформації про створену монету до смарт-контракту є кроком, обов'язковим для проведення будь-яких подальших операцій із даною монетою. Оскільки інформація про користувача, який виконує поточну транзакцію, доступна в контексті виклику кожного методу смарт-контракту, то передавати адресу початкового власника монети в якості параметра не потрібно (рисунок 2.1).

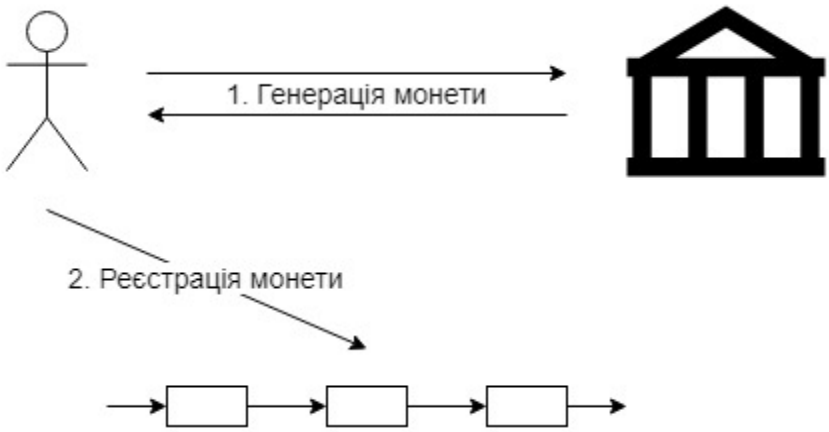


**Рисунок 2.1** – Генерація та реєстрація монети

Нехай абонент $A$ хоче отримати монету $C$ і зареєструвати її у смарт-контракті. Для цього абонент $A$, Банк $B$ і смарт-контракт $SC$ виконують такі кроки.

1) $A$ випадковим чином генерує серійний номер $0 \leq SN < n$ і обирає номінал монети $D$.

2) $A$ обирає випадкове число $r$, $1 < r < n$, $(r, n) = 1$ і обчислює $SN' = SN\, r^{e_D} \, mod \, n_D$.

3) $A$ надсилає на підпис банку осліплений серійний номер $SN'$ і обраний номінал $D$.

4) Банк $B$ генерує підпис $S' = SN'^{d_D} \, mod \, n_D$ і знімає із рахунку абонента $A$ відповідну суму коштів.

5) $A$ обчислює $S = S'r^{-1}\ mod\ n = SN^{d_D}\ mod\ n_D$, отримуючи таким чином дійсний цифровий підпис банку на серійному номері.

6) $A$ формує монету $C = \langle SN, S, D\rangle$ та формує транзакцію, в якій монета додається до смарт-контракту $SC$.

7) $SC$ перевіряє, що така монета не була зареєстрована раніше а також перевіряє чи дійсно $SN = S^{e_D}\ mod\ n_D$.

8) Смарт-контракт $SC$ вносить монету $C$ до реєстру.

Схематично, дану процедуру можна зобразити наступним чином:

$$(1)\ A \rightarrow B : SN', D$$

$$(2)\ A \leftarrow B : S'$$

$$(3)\ A \rightarrow SC : C$$

**Передача монети іншому власнику.** Для того, аби передати монету іншому власнику, поточний власник монети повинен викликати відповідний метод смарт-контракту, передавши в якості параметрів серійний номер монети та адресу її нового власника. Даний метод виконує перевірку того, що дана монета дійсно існує (тобто була зареєстрована у смарт-контракті), а користувач, який викликав метод, дійсно є її власником. Якщо всі перевірки завершились успіхом, то у смарт-контракт записується адреса нового власника даної монети.

Дана процедура є найпростішою у даній системі і схематично може бути зображена наступним чином:

$$(1)\ A \rightarrow SC : SN, address_B$$

де $A$ – абонент, який виконує переказ, $SC$ – смарт-контракт, $SN$ – серійний номер монети, $B$ – абонент, який є новим власником монети, $address_B$ – адреса абонента $B$ у мережі Ethereum.

**Зарахування монети на рахунок в Банку.** Дана процедура проводиться у випадку, якщо поточний власник монети вирішує «використати» одну із своїх монет, зарахувавши на свій рахунок у Банку кошти, що відповідають її номіналу. На відміну від класичної системи

електронних платежів Чаума, у пропонованій нами системі ця процедура є більш складною та вимагає від Банку проведення таких перевірок.

– Перевірка того, що монета з таким серійним номером була зареєстрована у смарт-контракті. Якщо монета не була зареєстрована, то стає неможливою перевірка того, що її початковий власник не спробував розрахуватись нею двічі, тому Банк відмовляється обслуговувати такі монети. Разом із тим всі проміжні власники повинні слідкувати за тим, аби інформація про монету, якою проводиться оплата перш за все додавалась в блокчейн і тільки після цього надавати відповідні послуги абоненту, що проводить оплату.

– Перевірка коректності цифрового підпису монети ключем Банку, який відповідає її номіналу. Ця перевірка є аналогічною перевірці у класичній системі.

– Перевірка того, що монета ще не була використана.

– Перевірка того, що користувач, який хоче зарахувати монету на свій рахунок, дійсно є її власником.

Із проведенням останньої перевірки пов’язані деякі труднощі. Справа в тому, що облікові записи користувачів у мережі Ethereum жодним чином не прив’язані до їх особистості або до їх Банківських рахунків. Запис у смарт-контракті може підтвердити належність монети певному користувачу мережі Ethereum, проте ніяк не доводить її належність певному абоненту Банку. Це означає, що необхідно проведення певної процедури, в ході якої абонент Банку надає достатні докази того, що він є власником даної адреси і, відповідно, пов’язаних із даною адресою монет.

Нехай є абонент $A$, який хоче зарахувати на свій рахунок у Банку $B$ монету $C$, що йому належить. Із урахуванням необхідності проведення описаної вище перевірки, для цього пропонується виконати процедуру, що складається із таких кроків (рисунок 2.2).

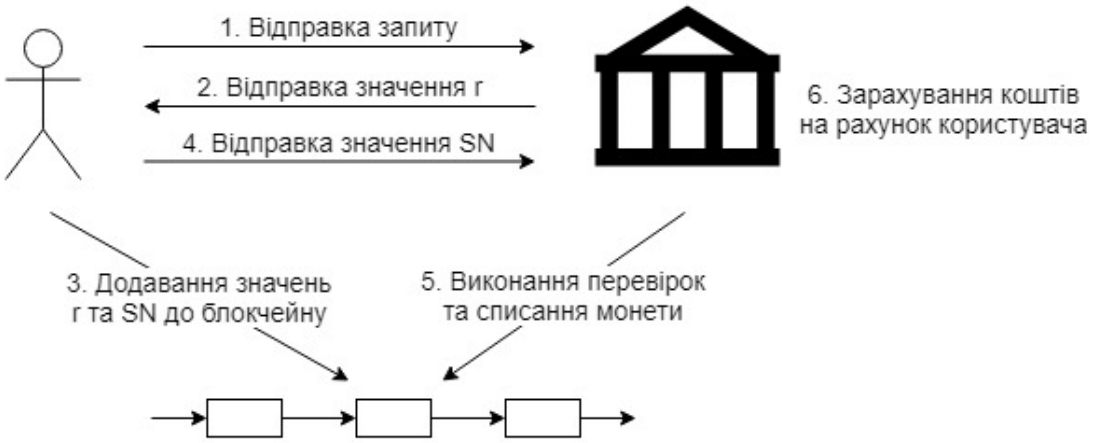


**Рисунок 2.2** – Зарахування монети на рахунок абонента в банку

1) Абонент $A$ відправляє до Банку $B$ запит, в якому ініціює проведення процедури зарахування.

2) Банк $B$, у відповідь, генерує велике випадкове число $r$ та повідомляє його абоненту $A$.

3) Абонент $A$, аби довести Банку $B$, що він дійсно є власником даної монети, додає до смарт-контракту $SC$ запис, що містить отримане число $r$ та серійний номер $SN$ монети $C$. До цього ж запису смарт-контрактом автоматично додається адреса користувача, який його створив.

4) Абонент $A$ повідомляє Банку $B$ значення $SN$.

5) У рамках нової транзакції Банк викликає метод смарт-контракту $SC$, який перевіряє чи існує запис із серійним номером $SN$ та числом $r$, а також, що адреса користувача, який вніс цей запис співпадає із адресою поточного власника монети. Додатково перевіряється чи не була дана монета використана раніше. Якщо всі перевірки пройшли успішно, то смарт-контракт $SC$ помічає монету $C$ як використану для попередження спроб її подальшої передачі іншим користувачам або спроб її повторного зарахування на банківський рахунок її останнього власника.

6) Якщо транзакція виконана на попередньому кроці завершилась успішно, Банк переказує на рахунок абонента кошти у розмірі рівному номіналу $D$ монети $C$.

Схематично, дана процедура може бути зображена наступним чином:

$$(1)\ A \rightarrow B : "initial"$$

$$(2)\ A \leftarrow B : r$$

$$(3)\ A \rightarrow SC : r$$

$$(4)\ A \rightarrow B : SN$$

$$(5)\ B \rightarrow SC : SN, r$$

## 2.2 Обмін монет на ефір

Виконання будь-яких операцій із смарт-контрактом потребує витрати певної кількості ефіру, який повинен бути наявним на рахунку користувача. Як ми знаємо, ефір можна отримати або за допомогою майнінгу або переказавши його із деякого іншого рахунку. Враховуючи, що майнінг потребує достатньої кількості обчислювальних ресурсів та часу, очевидно, що в контексті нашої системи, він не є доцільним методом отримання ефіру, а отже необхідно альтернативне рішення даної проблеми.

Пропоноване рішення базується на простій ідеї: ми можемо обмінювати отримані у Банку монети на ефір у спеціальних *обмінних пунктах*. При проведенні звичайного платежу, переказ монет, шляхом виклику відповідного методу смарт-контракту, виконує поточний власник монет, тобто покупець. У випадку із обміном монет на ефір така схема не може працювати, оскільки на Ethereum рахунку покупця ще не має ефіру, який би він міг використати для проведення платежу. Тому для виконання операції обміну пропонується процедура, яка за участі абонента $A$, Банку $B$, обмінного пункту $E$ та смарт-контракту $SC$ складається із таких кроків (рисунок 2.3).

1) Абонент $A$ підписує монети у Банку $B$ із використанням сліпого цифрового підпису.

2) Абонент $A$ пересилає монети в обмінний пункт $E$, без їх внесення до смарт-контракту $SC$.

3) Отримавши монети, обмінний пункт $E$ викликає спеціальну операцію смарт-контракту $SC$, яка:

а) перевіряє, що отримані монети нікому не належать (тобто вони ще не були зареєстровані у смарт-контракті)

б) записує власником монет обмінний пункт (тобто користувача, який викликав дану операцію)

в) перераховує із рахунку обмінного пункту ефір, кількість якого відповідає сумі номіналів отриманих монет, на рахунок абонента $A$, переданий у якості аргументу даної операції.

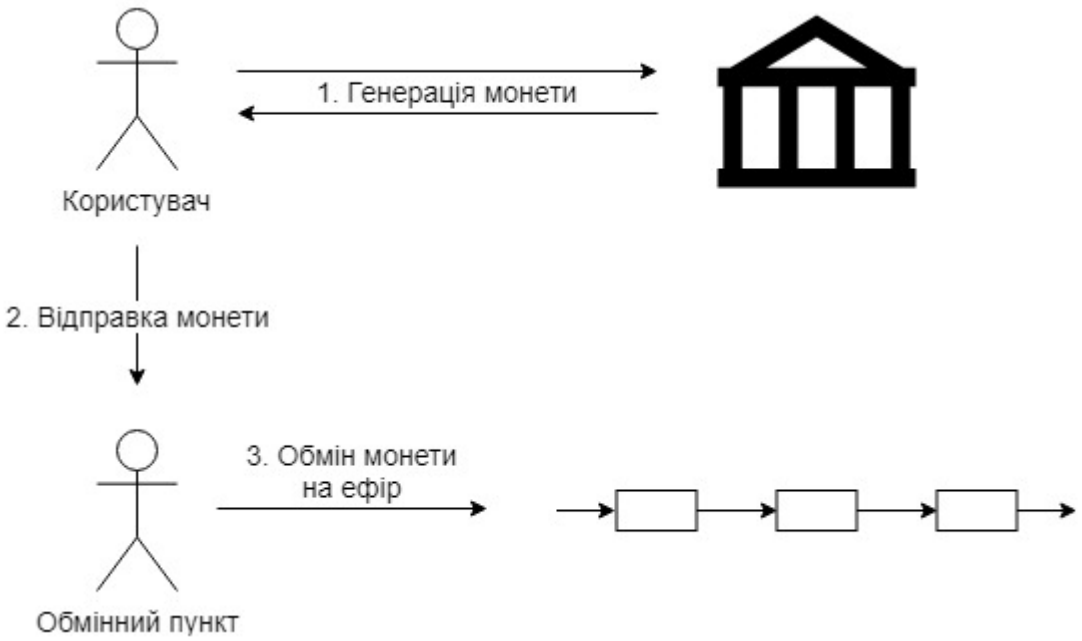


**Рисунок 2.3** – Обмін монети на ефір у обмінному пункті

Обмінний пункт може підтримуватись як самим банком, так і незалежними користувачами мережі Ethereum, являючись, таким чином, певним видом платної послуги, оплата за яку проводиться монетами нашої системи.

Проте даний підхід має інший суттєвий недолік. Оскільки передача монет від користувача до обмінного пункту відбувається поза блокчейном,

стає можливим шахрайство, коли обмінний пункт отримує монети і вносить їх на свій рахунок, не переказуючи при цьому ефір на рахунок користувача.

Для вирішення цієї проблеми пропонується модифікувати запропоновану систему електронних платежів таким чином, аби прив'язувати кожну монету до облікового запису мережі Ethereum, що належить її початковому власнику, в процесі її створення. Задля досягнення поставленої мети, для генерації серійного номера монети обирається не випадкове число, а відкритий ключ деякої системи електронного цифрового підпису. Відповідний йому секретний ключ використовується для формування електронного цифрового підпису адреси обраного облікового запису для подальшого його включення до складу монети. Оновлена процедура генерації монети складається із таких кроків.

1) Абонент $A$ випадковим чином генерує пару ключів $K_{coin} = (pk_{coin}, sk_{coin})$ обраної системи ЕЦП та обирає бажаний номінал монети $D$.

2) За допомогою протоколу сліпого цифрового підпису абонент отримує значення $S_{pk} = Sign\big(pk_{coin}, (n_D, d_D)\big)$, де $(n_D, d_D)$ – це секретний ключ Банку $B$, що відповідає номіналу монет $D$.

3) Абонент $A$ обирає свій обліковий запис у мережі Ethereum, до якого він хоче прив'язати нову монету – позначимо його також $A$.

4) Абонент $A$ обчислює $S_A = Sign(A, sk_{coin})$.

5) Абонент $A$ формує монету $C = \langle pk_{coin}, S_{pk}, A, S_A, D\rangle$.

Процедури внесення монети в реєстр та обміну монети модифікуються таким чином, аби перевіряти цифровий підпис $S_A$ адреси, до якої монета прив'язується або на який нараховується відповідна кількість ефіру. Запропонована модифікація монети та процесу її формування дає нам такі переваги.

1) Шахрайство в процесі виконання обміну монети стає неможливим, оскільки обмінний пункт не зможе зарахувати монету на будь-яку іншу адресу,

окрім тої, яка вказана у самій монеті та підписана секретним ключем $sk_{coin}$ її початкового власника.

2) Виклик методу смарт-контракту, який зараховує монету на адресу певного користувача можливо ініціювати навіть від імені деякого іншого користувача мережі, оскільки підмінити адресу власника на будь-яку іншу він все-одно не зможе, не знаючи секретного ключа $sk_{coin}$.

Крім того, наявність асоційованої із монетою ключової пари деякої системи ЕЦП дає нам можливість усунути інший недолік описаної процедури обміну монети. Справа у тому, що, незважаючи та те, що обмінний пункт більше не може зарахувати монету на інший рахунок і змушений перерахувати за неї певну кількість ефіру на рахунок її власника, він все ще має можливість перерахувати кількість ефіру, що значно менше домовленої. Для вирішення даної проблеми пропонується проводити процедуру обміну в наступний спосіб (для простоти припустимо, що відбувається обмін тільки однієї монети).

1) Абонент $A$ генерує ключову пару $(pk_{coin}, sk_{coin})$ і формує монету $C = \langle pk_{coin}, S_{pk}, A, S_A, D \rangle$ описаним вище способом.

2) Абонент $A$ домовляється із обмінним пунктом $E$ про кількість ефіру $N$, яка буде обміняна на монету $C$.

3) Абонент $A$ формує запит на обмін ефіру $R = (C, N, t, S_R)$, де $C$ – монета, $N$ – домовлена кількість ефіру, $t$ – термін дії запиту, $S_R = Sign(C \parallel N \parallel t, sk_{coin})$ – цифровий підпис параметрів запиту секретним ключем абонента.

4) Абонент $A$ надсилає запит R в обмінний пункт $E$.

5) Обмінний пункт $E$ перевіряє правильність сформованого запиту, зокрема, що зазначене в ньому число $N$ дійсно дорівнює домовленому. Якщо всі параметри зазначено правильно, то переходимо до 6-го кроку.

6) Обмінний пункт викликає у смарт-контракту $SC$ операцію обміну, передаючи в якості параметрів отриманий від абонента запит.

7) Смарт-контракт $SC$ виконує такі кроки.

а) Перевіряє, що цифрові підписи $S_{pk}$ та $S_A$, які входять до складу монети $C$ дійсні.

б) Перевіряє, що монета $C$ не зареєстрована у смарт-контракті.

в) Перевіряє, що цифровий підпис $S_R$ запиту $R$ дійсний.

г) Перевіряє, що термін дії $t$ запиту $R$ менший за поточний час. Під поточним часом розуміється час генерації блоку, в який додається транзакція, в рамках якої виконується обмін.

д) Якщо всі попередні перевірки пройшли успішно, перераховує на рахунок абонента ефір у кількості N одиниць та записує обмінний пункт власником монети $C$.

Схематично, обмін монети на ефір виглядає наступним чином:

$$(1)\ A \rightarrow B:\ pk'_{coin}, D$$

$$(2)\ A \leftarrow B:\ S'_{pk}$$

$$(3)\ A \leftrightarrow E:\ N$$

$$(4)\ A \rightarrow E:\ R$$

$$(5)\ E \rightarrow SC:\ R$$

Сума обміну $N$ може обиратись сторонами обміну згідно деякого глобального курсу. Оскільки курс може змінюватись з часом, нам необхідно обмежити термін придатності запиту $R$, аби запобігти спробам обмінного пункту затримати виконання операції обміну до часу, поки курс зміниться так, що вартість домовленої кількості ефіру стане значно меншою, ніж номінал монети. Саме з цією метою до запиту $R$ додається значення $t$, яке перевіряється смарт-контрактом в процесі виконання обміну.

Аби додати можливість виконання обміну не однієї, а відразу декількох монет, достатньо передавати їх всі у складі запиту $R$, а підпис $S_R$ формувати секретним ключем, що відповідає будь-якій із монет. Для зручності можна вважати, що підпис формується монетою, яка у запиті $R$ згадується першою.

## 2.3 Аналіз властивостей запропонованого рішення

**Анонімність.** Покажемо, що всі операції, проведені в нашій системі зберігають анонімність користувачів.

**Твердження 2.1.** *Банк не має можливості встановити зв'язок між монетою і користувачем, якому вона була видана.*

**Доведення.** Оскільки монета підписується Банком із використанням протоколу сліпого цифрового підпису, то Банк не має можливості дізнатись її серійний номер під час генерації підпису. □

**Твердження 2.2.** *Додавання монети в блокчейн та виконання розрахунків є анонімними операціями.*

**Доведення.** Реєстрація монети у смарт-контракті зв'язує монету із адресою у мережі Ethereum користувача, який є її початковим власником. Для передачі монети у власність іншому користувачу необхідно, також, вказати його адресу у мережі Ethereum. Жодна інша інформація про користувачів для проведення даних операцій не потребується. □

Це означає, що блокчейн дозволяє встановити зв'язок між монетою та адресами користувачів мережі Ethereum, які коли-небудь володіли даною монетою. Проте, система не передбачає наявності будь-якого зв'язку між обліковими записами користувачів Банку та їх адресами у мережі Ethereum, а отже встановити справжню особу поточного або будь-якого попереднього власника монети не можливо. Це, в свою чергу, означає, що проведення платежу в запропонованій системі є анонімною операцією.

***Зауваження.*** Рахунок в Банку зобов'язані мати тільки перший та останній власники монети. Проміжні власники рахунку можуть не мати взагалі.

Єдиний абонент мережі Ethereum, чия особа встановлюється банком, це останній власник монети, який виконує зарахування коштів на власний рахунок. Підхід, який може бути використано для досягнення повної

анонімності дій будь-якого абонента, полягає у створенні нового облікового запису (і, відповідно, нової адреси) у мережі Ethereum для отримання кожного нового платежу від інших абонентів. Зауважимо, що для багатьох абонентів такий підхід може бути надлишковим і використання однієї і тієї ж адреси для всіх операції може їх цілком задовольняти. Тому система дає змогу обирати той рівень анонімності, який абонент вважає доцільним для себе.

**Твердження 2.3.** *Операція обміну монет на ефір не порушує анонімності користувачів.*

**Доведення.** Аналогічно до процедури реєстрації монети у смарт-контракті, під час обміну на ефір встановлюється зв'язок тільки між монетою та адресою користувача у мережі Ethereum, який її згенерував. Дана операція не передбачає передачі в обмінний пункт жодної додаткової інформації про початкового власника монети, отже вона є анонімною. □

На основі доведених тверджень ми можемо сформулювати наступну теорему.

**Теорема 2.1.** *Запропонована система електронних платежів є анонімною.*

**Доведення.** Із твердження 2.1 слідує, що Банк не може встановити особу користувача під час генерації і підписування монети, а із тверджень 2.2 і 2.3 слідує, що анонімними є всі операції, які проводяться за участі смарт-контракту. Особа користувача розкривається лише під час зарахування монети на його рахунок у Банку, що відбувається тільки за його бажання. У всіх інших випадках, описана система є анонімною і не дозволяє встановити особу жодного користувача. □

**Можливість подвійної витрати монет.** Важливою особливістю моделі обчислень EVM є те, що всі транзакцію виконуються послідовно одна за одною, тому повністю виключається можливість створення умов перегонів (race conditions) між двома транзакціями, які взаємодіють із одним і тим же

смарт-контрактом, здатних пошкодити цілісність і узгодженість інформації, що зберігається у смарт-контракті.

Враховуючи послідовне виконання транзакцій, легко бачити, що у запропонованому рішенні можливість подвійного використання монети повністю виключається, оскільки будь-які зміни власника або стану монети, виконані однією транзакцією, відразу стануть видимими усім наступним.

**Безпека.** Покажемо, що структура описаного смарт-контракту та загальні властивості рішень на базі блокчейну Ethereum забезпечують безпеку усіх операцій нашої системи.

**Твердження 2.4.** *Монету* $C = \langle pk_{coin}, S_{pk}, A, S_A, D \rangle$ *не може собі привласнити ніхто, окрім користувача із адресою* $A$ *у мережі Ethereum.*

**Доведення.** Монета може бути внесена до смарт-контракту двома шляхами: за допомогою операції реєстрації або за допомогою операції обміну на ефір.

У випадку реєстрації, смарт-контракт перевіряє, що підпис $S_A$ є коректним і вносить її у реєстр тільки, якщо перевірка пройшла успішно. Оскільки тільки справжній власник монети знає секретний ключ $sk_{coin}$, жоден інший користувач не може сформувати коректний підпис $S_A$, і, таким чином, привласнити монету собі.

У випадку обміну монету на ефір, обмінний пункт повинен передати у смарт-контракту, отриманий від користувача, запит $R = (C, N, t, S_R)$. Смарт-контракт виконає операцію обміну тільки, якщо підпис $S_R$ є коректним, а термін дії запиту ще не закінчився. Знову ж, оскільки тільки користувач, який генерував монету, знає секретний ключ $sk_{coin}$, обмінний пункт не може підробити запит $R$ і зарахувати на рахунок користувача меншу кількість ефіру, ніж було домовлено.

Таким чином, тільки справжній власник монети може зареєструвати її у смарт-контракті або обміняти на ефір у обмінному пункті. Жоден інший користувач привласнити монету собі не може. □

**Твердження 2.5.** *Передати монету у власність іншому користувачу тільки її поточний користувач.*

**Доведення.** Під час проведення операції передачі монети іншому користувачу, смарт-контракт перевіряє, що транзакція виконується користувачем, який є поточним власником даної монети. Більш того, інформацію про користувача, який виконує транзакцію, смарт-контракт отримує із метаданих транзакції, а не як аргумент виклику, тому підмінити цю інформацію неможливо принципово.

Оскільки кожна транзакція в Ethereum підписується цифровим підписом користувача, який її виконує, гарантується, що ніхто не може видати себе за власника деякої монети і виконати з нею якісь дії від його імені.

Отже, передача монети іншому користувачу може бути виконана тільки користувачем, який є її поточним власником. □

**Твердження 2.6.** З*арахування монети на рахунок у Банку може бути виконано тільки користувачем, який є власником даної монети.*

**Доведення.** На останньому кроці запропонованого протоколу зарахування монети на рахунок користувача в Банку викликається операція смарт-контракту, яка перевіряє що користувач, який володіє даною монетою і користувач, який вніс запис із отриманим від банку числом $r$ співпадають. Навіть, якщо власник монети не володіє рахунком на який виконується зарахування даної монети, необхідна його згода на проведення цієї операції. Відповідно, зарахування монети на рахунок в Банку без відома її поточного власника неможливе.

Оскільки число $r$, яке необхідно внести до смарт-контракту, є досить великим і генерується випадковим чином, воно є відомим тільки Банку і користувачу, який звернувся до Банку із запитом на зарахування монети. Таким чином, число $r$ встановлює відповідність між обліковим записом користувача в Банку (якому число було повідомлено) і обліковим записом

користувача у мережі Ethereum (який вніс число до смарт-контракту), а отже і між монетами, які належать даному користувачу.

Отже, виконання зарахування монети на рахунок у Банку без відома і згоди її поточного власника є неможливим. □

**Твердження 2.7.** *Обмін монети на ефір не може бути виконаний частково, на користь тільки однієї із сторін обміну.*

**Доведення.** Зарахування монет на рахунок обмінного пункту та перерахунок відповідної кількості ефіру їх власнику виконуються в рамках однієї блокчейн транзакції, отже дана операція є атомарною і не може виконатись частково, тобто не може виконатись тільки перерахунок ефіру на рахунок користувача або тільки зарахування монет на рахунок обмінного пункту. □

**Твердження 2.8.** *Переписування історії блокчейну з метою привласнення монет є обчислювально важкою задачею.*

**Доведення.** Завдяки протоколам досягнення консенсусу, які використовуються для побудови блокчейна Ethereum, гарантується, що переписування його історії з метою подвійного використання або привласнення монет є обчислювально важкою задачею, яку неможливо вирішити на практиці. □

Доведені твердження дозволяють нам сформулювати наступну теорему.

**Теорема 2.2.** *Запропонована система електронних платежів забезпечує високий рівень безпеки.*

**Доведення.** Із твердження 2.4 слідує, що будь-які операції із монетою, яка ще не була внесена до смарт-контракту, можуть бути виконані тільки за ініціативою її справжнього власника.

Із тверджень 2.5 і 2.6 слідує, що виконання будь-яких операцій над монетою, яка зареєстрована в блокчейні можливе тільки за відома і згоди її поточного власника.

Твердження 2.7 показує, що обмін монет на ефір є безпечною операцією, оскільки всі обчислення виконуються в рамках єдиної блокчейн транзакції, а отже, операція є неподільною і не може бути виконана частково, на користь тільки одного із учасників.

Нарешті, із твердження 2.8 слідує, що перепис історії транзакцій із метою привласнення чи подвійної витрати монет є обчислювально тяжкою, а отже неможливою на практиці, задачею.

Підсумовуючи усі доведені твердження, ми бачимо, що запропонована система електронних платежів дійсно забезпечує високий рівень безпеки для всіх учасників. □

## 2.4 Недоліки запропонованого рішення

Зважаючи на будову та принципи роботи запропонованої системи електронних платежів можна виділити такі її недоліки.

– Необхідність встановлення зв’язку між особистістю користувача і його адресою у мережі Ethereum під час виконання операції зарахування монети на рахунок в Банку. Хоч, у випадку необхідності досягнення повної анонімності, і можливо генерувати нову адресу для проведення кожного наступного розрахунку, це створює певні незручності у користуванні, тому дана проблема заслуговує окремого розгляду і пошуку шляхів її вирішення.

– Виконання будь-яких операцій із монетами потребує виконання блокчейн транзакцій, за проведення яких користувач повинен сплатити певну кількість ефіру на користь майнера. Хоча у даній роботі і була запропонована схема обміну монет на ефір, необхідно проведення подальших досліджень з метою пошуку способів спрощення системи з точки зору кінцевих користувачів та підвищення зручності її користування.

– Запропонована система має обмежену кількість доступних номіналів монет, тому проведення розрахунків на довільні суми може вимагати від користувача генерації певної кількості нових монет за участі Банку. Це може призвести до того, що Банк буде залучено до проведення досить великої кількості платежів, що, знову ж, спричинить підвищення вимог до його обчислювальних ресурсів. Для уникнення даної проблеми можливо вдосконалити запропоновану систему, додавши до неї операцію розміну монет, яка буде проводитись повністю на блокчейні і не вимагатиме участі Банку.

– Взаєморозрахунки у запропонованій системі можливі тільки між користувачами одного й того ж самого банку. Оскільки на практиці нерідко виникає необхідність у проведенні розрахунків між клієнтами різних банків, є доцільним проведення додаткових досліджень з метою вдосконалення даної системи для забезпечення такої можливості.

## 2.5 Аспекти практичного застосування

Запропоновані у даній роботі алгоритми та математичну модель можна використовувати для розробки повноцінної системи електронних платежів, яка працює в межах одного банку. Основними компонентами такої системи є:

– блокчейн мережі Ethereum

– банківське програмне забезпечення

– клієнтський застосунок.

Банківське програмне забезпечення, окрім традиційних складових, міститиме компонент, головною задачею якого буде взаємодія із смарт-контрактом, опублікованим у блокчейні мережі Ethereum. Даний компонент буде залучений до проведення операцій генерації монети та

зарахування монети на рахунок абонента банку, у відповідності до процедур, описаних у даній роботі.

Клієнтський застосунок матиме графічний інтерфейс і буде встановлюватись на обчислювальні пристрої користувачів (персональні комп'ютери і мобільні пристрої). Застосунок буде відображати користувачу всю наявну інформацію стосовно стану його рахунку і дозволятиме проводити операції генерації монет, обміну монет на ефір, виконання розрахунків і зарахування монет на банківський рахунок користувача. Клієнтський застосунок буде взаємодіяти із відповідним компонентом банківського програмного забезпечення та проводитиме транзакції у мережі Ethereum.

Маючи достатню кількість монет, користувачі здатні проводити більшість платежів без обміну даними із банківським програмним забезпеченням, а взаємодіючи тільки із мережею Ethereum. Це означає суттєве зменшення навантаження на обчислювальні ресурси банку, зменшення витрат на їх підтримку і, як результат, можливе зменшення вартості послуг банку. В цьому полягає основна перевага даної СЕП у порівнянні із класичною СЕП Чаума.

## Висновки до розділу 2

У даному розділі було запропоновано модифікацію системи електронних платежів Чаума, яка дозволяє:

– передавати монети між абонентами системи

– зараховувати, представлені монетою, кошти на банківський рахунок користувача

– обмінювати монети на ефір, необхідний для проведення транзакцій у мережі Ethereum.

Було доведено, що запропонована система забезпечує повну анонімність усіх користувачів, безпеку всіх виконаних операцій та неможливість повторного використання монет.

Було показано, що система містить певні недоліки, які заслуговують проведення додаткових досліджень з метою її подальшого вдосконалення.

Було показано, як дану систему можна реалізувати та використовувати на практиці.

Система побудована на основі технології смарт-контрактів Ethereum, що відрізняє її від усіх існуючих рішень даної задачі.

# ВИСНОВКИ

У даній роботі було проведено огляд систем електронних платежів, їх класифікацію, особливості та принципи побудови деяких із них. Найбільш детально було розглянуто систему електронних платежів Чаума, її переваги та недоліки. Було встановлено, що одним із найбільш суттєвих недоліків даної системи є неможливість передавачі монет між користувачами без залучення банку.

Також у даній роботі було розглянуто принципи роботи та основні властивості технології блокчейн і технологій, в основу яких вона покладена: криптовалюти біткоїн і платформи смарт-контрактів Ethereum. Було з'ясовано, що, завдяки своїм властивостям і можливостям, платформа Ethereum може бути використана для усунення недоліків системи електронних платежів Чаума та її вдосконалення.

В результаті, було запропоновано модифікацію системи електронних платежів Чаума, в якій є можливою передача монет між користувачами системи, без залучення банку до проведення даної операції. Було проаналізовано властивості побудованої системи і доведено, що всі операції в ній є безпечними та анонімними, а подвійне використання монет є неможливим. Було показано, як описана система може бути реалізована на практиці і які переваги вона має перед класичною системою електронних платежів Чаума.

Запропонована система містить певні недоліки, які не порушують її основних властивостей, проте ускладнюють її використання кінцевими користувачами. Подальші дослідження можуть бути спрямовані на усунення недоліків та розширення функціональних можливостей описаної системи.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Дылянов Д. В. Защита информации в банковском деле и электронном бизнесе / Д. В. Дылянов, М. А. Деднев, М. А. Иванов. – Москва: КУДИЦ-ОБРАЗ, 2004. – 512 с.

2. Запечников С. В. Криптографические протоколы и их применение в финансовой и коммерческой деятельности / С. В. Запечников. – Москва: Горячая линия - Телеком, 2007. – 320 с.

3. David Chaum. Blind signatures for untraceable payments // Advances in Cryptology Proceedings of Crypto. - 1983. - С. 199–203.

4. Anonymous Transferable E-Cash [Електронний ресурс] / F.Baldimtsi, M. Chase, G. Fuchsbauer, M. Kohlweiss. – 2015. – Режим доступу до ресурсу: https://www.microsoft.com/en-us/research/wp-content/uploads/2015/03/PKC-BCFK15.pdf.

5. Tewari H. Fully Anonymous Transferable Ecash [Електронний ресурс] / H. Tewari, A. Hughes. – 2016. – Режим доступу до ресурсу: https://eprint.iacr.org/2016/107.pdf .

6. Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System [Електронний ресурс] / Satoshi Nakamoto. – 2008. – Режим доступу до ресурсу: https://bitcoin.org/bitcoin.pdf.

7. Antonopoulos A. Mastering Bitcoin / Andreas M. Antonopoulos., 2014. – 282 с.

8. Szabo N. Smart Contracts: Building Blocks for Digital Markets [Електронний ресурс] / Nick Szabo. – 1996. – Режим доступу до ресурсу: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html.

9. Ethereum White Paper [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ethereum/wiki/wiki/White-Paper.

10. Solidity — Solidity 0.4.23 documentation [Електронний ресурс] - Режим доступу до ресурсу: https://solidity.readthedocs.io/en/v0.4.23/

11. Wood G. Mastering Ethereum [Електронний ресурс] / G. Wood, A. Antonopoulos – Режим доступу до ресурсу: https://github.com/ethereumbook/ethereumbook.

12. King S. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake [Електронний ресурс] / S. King, S. Nadal. – 2012. – Режим доступу до ресурсу: https://peercoin.net/assets/paper/peercoin-paper.pdf.

13. «Алгоритмы консенсуса»: Подтверждение доли и доказательство работы [Електронний ресурс] - Режим доступу до ресурсу: https://habr.com/company/bitfury/blog/327468/

# ДОДАТОК А
# ТЕКСТИ ПРОГРАМ

## А.1 Смарт-контракт, що лежить в основі побудованої системи

```
pragma solidity ^0.4.0;

contract CoinExchange {

    struct Coin {
        // Address of the first owner of this coin
        address initialOwner;
        // Initial owner address signature
        bytes ownerSig;
        // Serial number signature
        bytes serialNumSig;

        // Whether this coin exist at all
        bool exist;
        // Whether this coin has already been withdrawn by a user
        bool withdrawn;
        // Ethereum address of the current owner of the coin
        address owner;

        // Challenge number received by the user
        // from bank during coin withdrawal.
        uint256 challengeNumber;
        // Address of a user which was challenged,
        // since the coin's owner may change in between
        address challengedAddress;
    }

    // Should be replaced with real address of Bank account
    address private bankAddress = 0xDEADBEEF;

    // Maps coin's serial number to the structure
    // which stores all related information
    mapping(bytes => Coin) private coins;

    function register(bytes serialNumber, bytes ownerSig, bytes serialNumSig)
public {
        Coin storage coin = coins[serialNumber];

        require(!coin.exist);
        require(checkSignature(toBytes(msg.sender), ownerSig, serialNumber));

        coin.exist = true;
        coin.owner = msg.sender;
        coin.initialOwner = msg.sender;
        coin.ownerSig = ownerSig;
        coin.serialNumSig = serialNumSig;
    }

    function transfer(bytes serialNumber, address receiver) public {
        Coin storage coin = coins[serialNumber];

        require(coin.exist);
        require(coin.owner == msg.sender);
        require(!coin.withdrawn);

        coin.owner = receiver;
    }
```

```
    function exchange(bytes serialNumber, address originalOwner, bytes ownerSig,
bytes serialNumSig, uint32 value) public {
        Coin storage coin = coins[serialNumber];

        require(!coin.exist);
        require(checkSignature(toBytes(originalOwner), ownerSig, serialNumber));

        // send ether to coin's owner
        originalOwner.transfer(value);

        // assign coin to ourselves
        coin.exist = true;
        coin.owner = msg.sender;
        coin.initialOwner = originalOwner;
        coin.ownerSig = ownerSig;
        coin.serialNumSig = serialNumSig;
    }

    function challenge(bytes serialNumber, uint256 challengeNum) public {
        Coin storage coin = coins[serialNumber];

        require(coin.exist);
        require(!coin.withdrawn);
        require(coin.owner == msg.sender);

        coin.challengeNumber = challengeNum;
        coin.challengedAddress = msg.sender;
    }

    function withdraw(bytes serialNumber, uint256 challengeNum) public {
        require(bankAddress == msg.sender);

        Coin storage coin = coins[serialNumber];
        require(coin.exist);
        require(!coin.withdrawn);
        require(coin.owner == coin.challengedAddress);
        require(coin.challengeNumber == challengeNum);

        coin.withdrawn = true;
    }

    function getCoinState(bytes serialNum) public constant returns (string) {
        Coin storage coin = coins[serialNum];
        if (!coin.exist return 'invalid';
        if (coin.withdrawn) return 'withdrawn';

        return 'valid';
    }

    function getOwner(bytes serialNumber) public constant returns (address) {
        Coin storage coin = coins[serialNumber];

        require(coin.exist);
        require(!coin.withdrawn);

        return coin.owner;
    }

    function toBytes(address x) private pure returns (bytes b) {
        b = new bytes(20);
        for (uint i = 0; i < 20; i++)
            b[i] = byte(uint8(uint(x) / (2 ** (8 * (19 - i)))));
    }

    function checkSignature(bytes data, bytes signature, bytes key) private pure
returns (bool) {
        // Needs to be replaced with actual implementation
        return true;
    }
}
```