ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «АЛГОРИТМ ЗАБЕЗПЕЧЕННЯ ЦІЛІСНОСТІ ТА АВТЕНТИЧНОСТІ ДАНИХ ПРИ ПЕРЕДАЧІ В МЕРЕЖІ НА ОСНОВІ КРИПТОГРАФІЧНОГО ПІДПИСУ»

на здобуття освітнього ступеня магістра

зі спеціальності 126 Інформаційні системи та технології

освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ _____

(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав

здобувач вищої освіти Дмитро ІВАНЧЕНКО

група ІСДМ-61

Керівник: Валентина ДАНИЛЬЧЕНКО

науковий ступінь, вчене звання PhD

Рецензент:

науковий ступінь, вчене звання _____

Ім'я, ПРІЗВИЩЕ

Київ – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти Магістр
Спеціальність підготовки 126 Інформаційні системи та технології
Освітньо-професійної програми Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Каміла СТОРЧАК
«____» _____20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
Іванченку Дмитру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Алгоритм забезпечення цілісності та автентичності даних при передачі в мережі на основі криптографічного підпису
керівник кваліфікаційної роботи Данильченко В.М. PhD
затверджені наказом вищого навчального закладу від «15» жовтня 2024 р. № 320
2. Строк подання кваліфікаційної роботи 27 грудня 2024 р.
3. Вихідні дані до кваліфікаційної роботи

Мета дослідження: розробка автоматизованої системи для безпечного передавання даних у мережі між користувачами, що використовує криптографічний підпис із парою відкритого та закритого ключів.

Об'єктом дослідження є процеси та технології безпечної передачі даних у мережі з використанням криптографічних підписів і блокчейн-платформи Ethereum.

Предметом дослідження є алгоритми шифрування даних.

Науково-технічна література з питань, пов'язаних з наукою про дані.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Розробка моделі поведінки системи передачі даних
2. Уточнення бізнес-процесів завдання забезпечення безпеки при передачі даних.
3. Розробка структури програми для реалізації криптографічного підпису
4. Розроблення загального опису алгоритму розв'язання задачі забезпечення безпеки передачі даних

5. Перелік ілюстрованого матеріалу:
1. Мета, об’єкт та предмет дослідження

2. Завдання роботи та розділи магістерської роботи

3. Сучасні технології обміну інформацією та методи шифрування в інтернеті

4. Розробка безпечної системи передачі даних в інтернеті

5. Розрахунок витрат

6. Висновки та апробація

6. Дата видачі завдання: 15 жовтня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науко-технічної літератури	15.10-23.10.24	виконано
2	Вивчення алгоритмів і технологій	24.10-01.11.24	виконано
3	Розробка теоретичної моделі системи	02.11-07.11.24	виконано
4	Реалізація та тестування системи	08.11-20.11.24	виконано
5	Оформлення роботи: вступ, висновки, реферат	07.12-15.12.24	виконано
6	Розробка демонстраційних матеріалів	16.12-19.12.24	виконано

Здобувач вищої освіти _____ Дмитро ІВАНЧЕНКО

(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник

кваліфікаційної роботи _____ Валентина ДАНИЛЬЧЕНКО

(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 67 стор., 18 рис., 3 табл., 16 джерел.

Мета роботи – розробка автоматизованої системи для безпечного передавання даних у мережі між користувачами, що використовує криптографічний підпис із парою відкритого та закритого ключів.

Об'єкт дослідження – процеси та технології безпечної передачі даних у мережі з використанням криптографічних підписів і блокчейн-платформи Ethereum.

Предмет дослідження – алгоритми шифрування даних.

Короткий зміст роботи: Тема: Розробка алгоритму забезпечення цілісності та автентичності даних при передачі в мережі на основі криптографічного підпису.

Мета цієї роботи полягає у створенні автоматизованої системи для безпечної передачі даних у мережі між користувачами, використовуючи криптографічний підпис на основі пари відкритого та закритого ключів.

В процесі дослідження були проаналізовані сучасні методи шифрування та блокчейн-технології, зокрема криптосистема RSA та платформа Ethereum. Алгоритм RSA було обрано для забезпечення цілісності даних і створення електронного підпису завдяки його високій надійності. Платформа Ethereum, як децентралізована система зі смарт-контрактами, використовується для реалізації системи з підвищеною безпекою та прозорістю.

Результати дослідження показали, що розроблена система дозволяє створювати захищені месенджери, де дані залишаються недоступними для третіх осіб, забезпечуючи доступ з різних пристроїв. Водночас висока вартість транзакцій у блокчейні Ethereum обмежує економічну ефективність такого підходу.

Практична новизна полягає в інтеграції криптографічного підпису та смарт-контрактів для надійного обміну даними, що може бути застосовано в сучасних системах зв'язку та для захисту конфіденційної інформації.

КЛЮЧОВІ СЛОВА: Блокчейн, Ethereum, Захист даних, Смарт-контракти, Цифровий підпис, RSA (криптосистема).

ABSTRACT

Text part of the master's qualification work: 68 pages, 18 pictures, 3 table, 16 sources.

The purpose of the work - development of an automated system for secure data transmission in the network between users, using a cryptographic signature with a pair of public and private keys.

Object of research – processes and technologies for secure data transmission in the network using cryptographic signatures and the Ethereum blockchain platform.

Subject of research – data encryption algorithms.

Summary of the work: Topic: Development of an algorithm for ensuring the integrity and authenticity of data during network transmission based on a cryptographic signature.

The purpose of this work is to create an automated system for secure data transmission in the network between users using a cryptographic signature based on a pair of public and private keys.

In the course of the study, modern encryption methods and blockchain technologies were analyzed, in particular the RSA cryptosystem and the Ethereum platform. The RSA algorithm was chosen to ensure data integrity and create an electronic signature due to its high reliability. The Ethereum platform, as a decentralized system with smart contracts, is used to implement a system with increased security and transparency.

The results of the study showed that the developed system allows creating secure messengers where data remains inaccessible to third parties, providing access from different devices. At the same time, the high cost of transactions in the Ethereum blockchain limits the cost-effectiveness of this approach.

The practical novelty lies in the integration of cryptographic signatures and smart contracts for secure data exchange, which can be used in modern communication systems and to protect confidential information.

KEY WORDS : Blockchain, Ethereum, Data protection, Smart contracts, Digital signature, RSA (cryptosystem).

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

ПОДАННЯ ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ

на здобуття освітнього ступеня магістр

Направляється здобувач Іванченко Д.С. до захисту кваліфікаційної роботи за спеціальністю 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
на тему: «Алгоритм забезпечення цілісності та автентичності даних при передачі в мережі на основі криптографічного підпису».

Кваліфікаційна робота і рецензія додаються.

Директор ННІ ІТ _____
Ім'я ПРІЗВИЩЕ

Висновок керівника кваліфікаційної роботи

Здобувач Іванченко Дмитро Сергійович успішно виконав кваліфікаційну роботу на тему «Алгоритм забезпечення цілісності та автентичності даних при передачі в мережі на основі криптографічного підпису», у процесі виконання роботи здобувач продемонстрував високий рівень теоретичних знань та практичних навичок у використані сучасних технологій. Перелік використаних джерел свідчить про вміння здобувача розбиратись в наукових питаннях та застосовувати їх при дослідженнях. Роботу виконував сумлінно, акуратно та вчасно за планом.

Все це дозволяє оцінити виконану кваліфікаційну роботу здобувача Іванченка Д.С. на оцінку «відмінно» та присвоїти йому кваліфікацію магістр з інформаційних систем та технологій.

Керівник кваліфікаційної роботи _____ Валентина ДАНИЛЬЧЕНКО
(підпис) Ім'я ПРІЗВИЩЕ
«____» _____ 202_ року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач Іванченко Д.С. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою _____ Каміла СТОРЧАК
(підпис) Ім'я ПРІЗВИЩЕ

РЕЦЕНЗІЯ

на кваліфікаційну роботу

на здобуття освітнього ступеня магістра

здобувача вищої освіти Іванченка Дмитра Сергійовича
(прізвище, ім'я, по батькові)
на тему «Алгоритм забезпечення цілісності та автентичності даних при передачі в мережі на основі криптографічного підпису»

Актуальність.

В умовах сучасного розвитку інформаційних технологій, тема роботи є надзвичайно важливою. Критично важливим завданням для багатьох сфер, включно з фінансовими послугами, державним управлінням і приватним листуванням, є безпечне передавання даних у мережі. Інноваційний підхід використання криптографічного підпису дає змогу забезпечити цілісність і автентичність даних, вирішуючи низку проблем, пов'язаних із захистом інформації. Зростаючий обсяг використання блокчейн-технологій для створення децентралізованих систем обміну інформацією ще більше підкреслює актуальність цієї роботи.

Позитивні сторони.

1. У роботі проведено всебічний аналіз сучасних криптографічних алгоритмів та їх практичного використання для захисту даних.

2. Автор демонструє високий рівень теоретичних знань і практичних навичок у реалізації алгоритмів, зокрема розробки системи з криптографічним підписом на платформі Ethereum.

3. Робота має значну практичну цінність, оскільки запропонована система може бути використана для створення децентралізованих месенджерів або інших захищених платформ передачі даних.

Недоліки.

1. Аналіз ефективності запропонованого підходу варто було б зробити більш деталізованим, додавши порівняння з іншими криптографічними методами.

2. Технічна реалізація алгоритмів могла бути описана більш детально. Це допомогло б краще зрозуміти конкретні аспекти та етапи впровадження.

Відзначені зауваження не впливають на загальну позитивну оцінку кваліфікаційної роботи магістерської.

Висновок: *кваліфікаційна робота на здобуття ступеня магістра заслуговує оцінку "відмінно", а здобувач* Іванченко Дмитро *заслуговує присвоєння кваліфікації магістр з інформаційних систем та технологій*

Рецензент:

науковий ступінь, вчене звання _____ _____
(підпис) *Ім'я, ПРІЗВИЩЕ*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DES	(Data Encryption Standard)
AES	(Advanced Encryption Standard)
RSA	(Rivest-Shamir-Adleman)
ECC	(Еліптична крива криптографії)
EVM	(Віртуальна машина Ethereum)
XOR	(виключне АБО)
GHOST	(Greedy Heaviest Observed Subtree)

ВСТУП

Актуальність теми – полягає в тому що, зростання кількості атак і несанкціонованого доступу до конфіденційної інформації підкреслює важливість цієї роботи. Методи віддалених мережевих атак стають дедалі витонченішими, що призводить до того, що наявні алгоритми шифрування не завжди здатні забезпечити достатній рівень захисту даних. Водночас технології блокчейна, що мають високу криптографічну стійкість, продовжують активно розвиватися. Ці тенденції підкреслюють необхідність розроблення та впровадження системи безпечного передавання даних, заснованої на криптографічному підписі з використанням пар ключів - відкритого і закритого.

Мета роботи – розробка автоматизованої системи для безпечного передавання даних у мережі між користувачами, що використовує криптографічний підпис із парою відкритого та закритого ключів.

Завдання дослідження – в процесі дослідження були розглянуті такі завдання:

порівняльний аналіз сучасних систем обміну миттєвими повідомленнями, дослідження та порівняння актуальних алгоритмів і систем шифрування даних, вивчення можливостей блокчейн-платформи Ethereum і смарт-контрактів, розробка децентралізованої системи для передачі та зберігання даних на базі Ethereum.

Об'єкт дослідження – процеси та технології безпечної передачі даних у мережі з використанням криптографічних підписів і блокчейн-платформи Ethereum.

Предмет дослідження – алгоритми шифрування даних.

Методи дослідження – аналіз літератури та інформаційних джерел для вивчення існуючих систем обміну повідомленнями, алгоритмів шифрування та блокчейн-технологій; порівняльний аналіз для оцінки ефективності та безпеки різних методів і платформ; емпіричне моделювання для проєктування і реалізації децентралізованої системи передачі даних на базі Ethereum; експериментальний

метод для тестування розробленої системи та оцінки її продуктивності й надійності; метод структурного проєктування для створення архітектури децентралізованої системи.

Наукова новизна та практична значущість отриманих результатів – Наукова новизна полягає у розробці децентралізованої системи передачі даних, яка поєднує використання криптографічних підписів з відкритими і закритими ключами та можливості платформи Ethereum. Особливістю є інтеграція смарт-контрактів для забезпечення надійності, прозорості та криптографічної стійкості при передачі інформації.

Практична значущість отриманих результатів полягає у створенні автоматизованої системи, яка може бути використана для безпечного обміну конфіденційними даними між користувачами в різних галузях, таких як фінанси, охорона здоров’я або управління документами, забезпечуючи високий рівень захисту та зручність у використанні.

Апробація – результатів магістерської роботи включала їх доповідь і обговорення на другій міжнародній науково-практичній конференції, яка носила назву «Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії»

Структура кваліфікаційної магістерської роботи – магістерська робота складається з наступних структурних елементів:

1. Титульна сторінка:

Відображає назву роботи, автора, наукового керівника, навчальний заклад та рік виконання (1 сторінка).

2. Завдання на кваліфікаційну роботу включає в себе такі пунтки:

- Тему кваліфікаційної роботи
- Строк подання
- Вихідні дані для роботи

- Перелік питань, які потрібно розробити
- Перелік демонстраційного матеріалу
- Дату видачі завдання
- Календарний план

3. Анотація на українській та англійській мовах:

Призначена для експрес-ознайомлення з кваліфікаційною роботою. Включає в себе відомості про обсяг кваліфікаційної роботи, кількість рисунків, таблиць, джерел згідно із списком використаних джерел (3 сторінки).

4. Подання голові екзаменаційної комісії:

Включає інформацію, яка характеризує магістерську роботу та студента, а також містить рекомендації щодо допуску до захисту (1 сторінка).

5. Рецензія на кваліфікаційну роботу:

Офіційний документ, у якому незалежний експерт оцінює якість виконаної роботи. Включає в себе загальну інформацію, актуальність теми, позитивні сторони, недоліки та висновок (1 сторінка).

6. Зміст кваліфікаційної роботи:

Включає перелік усіх розділів, підрозділів, із зазначенням номерів сторінок.

7. Перелік умовних позначень:

У кваліфікаційній магістерській роботі є розділом, у якому перелічуються всі скорочення, абревіатури, спеціальні символи та терміни, що використовуються в тексті роботи.

8. Вступ:

Цей розділ знайомить читача з контекстом дослідження. У ньому визначаються актуальність теми, мета, завдання, об'єкт і предмет дослідження, методи, наукова новизна та практична значущість отриманих результатів. Окрім того, вказується на апробацію роботи та наводиться структура магістерської кваліфікаційної роботи.

9. Перший розділ:

Перший розділ здебільшого має теоретичний характер. У ньому аналізуються ключові категорії та поняття, розвиток теми, розкривається суть проблеми, наводяться різні наукові підходи і власний критичний аналіз. Обґрунтовується вибір методів дослідження. Сучасний стан проблеми ілюструється практичними даними у вигляді таблиць, графіків та діаграм із посиланнями на джерела.

10. Другий розділ:

Другий розділ є аналітичним і містить системний аналіз проблеми з використанням обраних методів дослідження. Результати оформлюються у вигляді таблиць, графіків, діаграм тощо. Цей розділ є основою для формулювання рекомендацій і розробки третього розділу роботи.

11. Третій розділ:

Третій розділ підсумовує результати дослідження, виявлені проблеми та розробляє конкретні пропозиції, рекомендації і заходи для їх вирішення.

12. Всиновки:

Підсумовують основні результати дослідження та роблять узагальнення щодо досягнутих цілей і виконаних завдань. Вони дають чітке розуміння того, що було отримано під час роботи, та можуть включати рекомендації для подальших досліджень чи практичного застосування.

13. Перелік посилань:

Розділ, у якому наведено список усіх джерел інформації, використаних під час написання роботи.

РОЗДІЛ 1. СУЧАСНІ ТЕХНОЛОГІЇ ОБМІНУ ІНФОРМАЦІЄЮ ТА МЕТОДИ ШИФРУВАННЯ В ІНТЕРНЕТІ

1.1 Засоби миттєвого обміну повідомленнями (месенджери)

Месенджери - це програми, мобільні додатки та веб-сервіси, що забезпечують обмін інформацією в реальному часі через Інтернет. Вони стали найважливішими інструментами для комунікації, пропонуючи не лише текстові повідомлення, а й обмін фотографіями, аудіо та відеофайлами, голосовими повідомленнями та документами. Важливість конфіденційності переданих даних вимагає високого ступеня захисту інформації. Щоб захистити дані від несанкціонованого доступу, розробники месенджерів впроваджують різні алгоритми і системи шифрування. До числа найбільш популярних месенджерів належать Viber, WhatsApp, Telegram та інші.

1.2 Порівняння популярних месенджерів

Популярні месенджери дозволяють користувачам зберігати листування для запобігання втрати даних. Однак додатки, такі як Wickr, Signal і Confide, не підтримують цю функцію, що обмежує їхнє використання і робить їх нішевими продуктами. Месенджери, такі як WhatsApp, Viber і Line, використовують хмарні сховища, як-от Apple iCloud і Google Drive, для збереження історії повідомлень, що запобігає втраті даних у разі зміни пристрою. Однак ці резервні копії не використовують кінцеве шифрування, і дані розшифровуються при відновленні на новому пристрої. Це створює загрозу безпеці, оскільки немає гарантії, що всі повідомлення залишаться зашифрованими в хмарі. У результаті особисті дані стають уразливими для хакерів і зловмисників, які можуть отримати доступ через хмарні сховища.

На відміну від цього, месенджери, такі як Wickr, Signal і Confide, не роблять резервних копій, що підвищує рівень безпеки, але тягне за собою обмеження, наприклад, неможливість відновити історію чатів у разі втрати пристрою. Telegram запропонував компроміс, створивши два типи чатів: секретні, які використовують

кінцеве шифрування і не включаються в резервні копії, і хмарні, які також зашифровані, але зберігаються в хмарі Telegram. Користувачі можуть відстежувати, які повідомлення зашифровані та зберігаються в хмарі, а які ні, але відновити кінцево зашифровані повідомлення неможливо. [1]

Однак розвиток блокчейн-технологій відкриває нові можливості для зберігання даних публічно, але безпечно. Месенджер на базі блокчейну може уникнути зберігання інформації на серверах, забезпечуючи високий рівень безпеки та конфіденційності.

1.3 Шифрування даних (криптографія)

Криптографія - це процес перетворення інформації в закодовану форму, доступну тільки користувачеві, який володіє відповідним секретним ключем для розшифрування. Це один із найефективніших методів захисту даних, що включає два основні типи шифрування: симетричне, за якого один і той самий ключ використовують для шифрування і дешифрування, і асиметричне, де застосовують кілька ключів - відкритий (публічний) і закритий (секретний). Основна мета шифрування - забезпечити конфіденційність даних, захищаючи їх під час зберігання і передавання через уразливі канали зв'язку, такі як Інтернет.

Шифрування стало необхідним інструментом в умовах, коли значна частина інформації зберігається на комп'ютерах і передається мережею, що робить її вразливою для різних атак. У минулому використовувався метод DES, розроблений у 70-х роках, але сьогодні цей алгоритм застарів, поступившись місцем складнішим і надійнішим методам захисту. Дані до шифрування називають відкритим текстом, а після обробки вони перетворюються на безглуздий набір символів, який можна розшифрувати тільки за наявності відповідного ключа. [2]

Симетричне шифрування працює швидше, але вимагає обміну ключами між відправником і одержувачем. Асиметрична криптографія, що використовує два різні ключі, уникає цієї необхідності, оскільки публічний ключ доступний усім, а закритий зберігається в секреті. Незважаючи на високий рівень захисту,

шифрування не гарантує абсолютної безпеки. Найпоширенішою атакою є брут-форс - метод перебору ключів, під час якого зловмисник пробує всі можливі комбінації до знаходження відповідного ключа. Надійність захисту залежить від довжини ключа: що він довший, то більше часу і ресурсів знадобиться на його злом. Існують й інші методи атак, як-от криптоаналіз, що шукає вразливості в алгоритмі, і атаки сторонніми каналами, які використовують помилки проєктування системи для вилучення інформації із зашифрованих даних.

Організації змушені використовувати криптографію, особливо коли співробітники працюють із переносними носіями даних або завантажують файли в хмарні сховища. Це створює ризик втрати контролю над даними, що може призвести до їх крадіжки або перехоплення зловмисниками. Шифрування допомагає захистити інформацію від несанкціонованого доступу та шкідливих програм, зберігаючи доступність даних для довірених користувачів. [3]

1.3.1 Симетричне шифрування

Симетричне шифрування - це метод криптографії, де один і той самий ключ використовується для шифрування і дешифрування даних. Цей ключ, пароль або кодова фраза передається між сторонами, які потім можуть використовувати його для захисту та доступу до повідомлень. Серед поширених алгоритмів симетричного шифрування виокремлюють стандарт шифрування даних (DES) з 56-бітними ключами. Потрійний DES, що застосовує алгоритм DES тричі з різними ключами, та стандарт розширеного шифрування (AES), рекомендований Національним інститутом стандартів і технологій США для безпечного зберігання та передавання даних.

Симетричні алгоритми шифрування популярні серед організацій завдяки їхній доступності та ефективності. Вони забезпечують надійний захист даних, залишаючись економічно вигідними. Цей метод шифрування включає вбудовану аутентифікацію, оскільки дані, зашифровані одним ключем, не можуть бути

розшифровані іншим. Крім того, симетричні ключі зазвичай меншого розміру, що знижує затримки під час шифрування і дешифрування даних.

Однак у цього методу є й недоліки. Один із головних - це можливість втрати ключів, що вимагатиме від організацій повторного шифрування даних із новим ключем, що може призвести до додаткових витрат і складнощів. Щоб мінімізувати ризики, організації повинні ефективно управляти ключами протягом їхнього життєвого циклу, включно з аудитом і контролем за безпекою зберігання ключів. Це можна зробити за допомогою системи управління ключами, наприклад, через апаратні модулі безпеки (HSM), які захищають ключі та надають доступ тільки авторизованим користувачам. У деяких випадках симетричний ключ може бути обгорнутий асиметричним шифруванням, але це може призвести до додаткових складнощів і циклічних залежностей ключів.

Для забезпечення безпеки та ефективності управління симетричними ключами організаціям важливо автоматизувати цей процес, особливо якщо кілька сторін обмінюються даними через різні захищені канали. Автоматизація управління ключами допомагає уникнути помилок і спрощує процес захисту даних. [4]

Data Encryption Standard (DES)

Алгоритм DES (Data Encryption Standard) являє собою симетричний криптографічний метод, у якому використовується один і той самий ключ для шифрування і дешифрування даних. Розроблений IBM і затверджений урядом США в 1977 році, DES став одним із найпопулярніших методів захисту даних, широко використовуваним у різних обчислювальних системах для зберігання й передавання інформації, включно з електронним обміном комерційною інформацією та поштовими системами.

Алгоритм DES, який використовує 56-бітний ключ для шифрування даних, працює з блоками по 64 біти. Він складається з 16 етапів і побудований на основі мережі Фейштеля, що розділяє дані на дві частини. У процесі роботи 64-бітний вхідний блок перетворюється на вихідний блок такої ж довжини. DES реалізує

комбінацію перестановок, замін і операції гаммування. Для виконання шифрування дані потрібно представити у вигляді двійкового коду.

Процес шифрування 64-бітного блока даних ділиться на три етапи:

1. Початкова підготовка блоку даних.

2. 16 раундів «основного циклу ».

3. Кінцева обробка блока даних

На першому етапі виконується початкова перестановка 64-бітного блоку даних, що збільшує хаотичність вихідного повідомлення і знижує ймовірність успішного криптоаналізу. Одночасно з цим виконується початкова перестановка 56-бітного ключа, з якого за певним алгоритмом створюють 48-бітні ключі для кожного з 16 раундів. Для цього 56-бітний ключ ділиться на дві частини по 28 біт, які потім зсуваються і переставляються для отримання 48 біт. Цей процес називається «перестановка зі стисненням». Отриманий результат — це 48-бітний набір. У процесі шифрування кожен біт 56-бітного ключа в середньому використовується в 14 з 16 підключів. При цьому розподіл використання окремих бітів може бути нерівномірним.

На другому етапі алгоритму 64-бітний блок ділиться на дві рівні частини по 32 біти, після чого розпочинається основний цикл перетворень на базі мережі Фейштеля. Цей цикл включає 16 ідентичних раундів, у кожному з яких обчислюється проміжне значення довжиною 64 біти, яке передається на наступний раунд.

Спочатку права половина блоку розширюється до 48 бітів за допомогою таблиці перетворення, яка додає 16 додаткових бітів. Це вирівнює розмір правої частини з довжиною ключа, що необхідно для виконання операції XOR. Така операція підсилює взаємозв’язок між бітами даних, ключа та результату.

Потім виконується операція XOR між розширеною до 48 біт правою частиною та ключем. Результат цієї операції подається на вхід блоку підстановки,

де він перетворюється в 32-бітне значення. Підстановка здійснюється з використанням восьми S-блоків, кожен з яких обробляє 6-бітні фрагменти даних і перетворює їх у 4-бітні, використовуючи заздалегідь задані таблиці заміни, розроблені для забезпечення високої криптостійкості. В результаті підстановки формується об'єднене 32-бітне значення.

Далі це 32-бітне значення проходить через перетворення, що не залежить від ключа, яке змінює порядок бітів так, щоб у наступному раунді кожен біт оброблявся іншим блоком перетворення.

Результат перетворення об'єднується через операцію XOR з лівою половиною початкового блоку. Потім ліва та права частини міняються місцями, і починається новий раунд.

Цей цикл повторюється 16 разів, при цьому права частина перетворюється за допомогою функції шифрування та часткового ключа, обчисленого з основного ключа. Після завершення всіх раундів виконується фінальне перетворення, яке є оберненим до початкового, завершує процес шифрування блоку.

В кінці всіх етапів блок даних вважається зашифрованим, і алгоритм переходить до обробки наступного блоку повідомлення. [5]

Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) — симетричний блочний шифр, затверджений урядом США для захисту секретної інформації. Він широко використовуються в програмних і апаратних рішеннях для забезпечення безпеки конфіденційних даних по всьому світу.

Ініціатором розробки AES став Національний інститут стандартів і технологій (NIST), який у 1997 році заявив про необхідність створення нового алгоритму для заміни застарілого стандарту шифрування даних (DES), що втратив надійність через вразливості до атак методом перебору.

Алгоритмом, обраним для AES, став Rijndael,R створений бельгійськими криптографами Вінсентом Рейменом і Йоаном Дайменом. Цей алгоритм вирізнявся високим рівнем безпеки, ефективністю і адаптивністю, що зробило його ідеальним вибором для нового стандарту.

Алгоритм Rijndael є симетричним блочним шифром, що підтримує довжини ключів 128, 192 і 256 біт. Обробка даних відбувається в блоках розміром 128 біт, хоча, крім вимог стандарту AES, можливе використання різних співвідношень розмірів блоків і ключів.

Кількість раундів шифрування визначається розміром ключа та блоку:

- 9 раундів для ключа і блоку по 128 біт,
- 11 раундів для ключа і блоку по 192 біт,
- 13 раундів для ключа і блоку по 256 біт.

Rijndael є шифром із лінійним перетворенням заміни та не використовує мережу Фейштеля. В основі алгоритму лежать три взаємопов'язані оборотні перетворення (шари), що забезпечують високу криптостійкість:

- Лінійне перетворення, що відповідає за дифузію даних.
- Нелінійне перетворення, що ускладнює залежність між вхідними та вихідними даними.
- Перетворення ключа, яке інтегрує зашифрований блок із поточним ключем.

Перед початком першого раунду в алгоритмі Rijndael виконується базова операція додавання ключа для підвищення безпеки процесу. Потім слідують Nr-1 стандартних раундів і фінальний раунд. Перетворення під час шифрування створюють структуру даних, яка зберігається до завершення всіх етапів.

У процесі шифрування дані організовуються в масив з 4 рядків, де кількість стовпців визначається як довжина блоку, поділена на 32. Ключ шифрування також представлений у вигляді масиву з 4 рядків, а його довжина, поділена на 32, визначає

кількість стовпців. Блоки даних можна розглядати як одновимірні масиви, що складаються з векторів розміром 4 байти.

Преобразування в алгоритмі Rijndael виконуються наступним чином:

1. Підстановка байтів (SubBytes): Нелінійна операція, яка обробляє кожен байт форми окремо, використовуючи зворотну таблицю підстановок (S-box). Ця таблиця побудована на двох перетвореннях, що забезпечують криптографічну стійкість.

2. Зсув рядків (ShiftRows): Рядки форми зсуваються на різні значення в залежності від їхнього положення, що сприяє підвищенню дифузії даних. Зсуви залежать від довжини блоку.

3. Змішування стовпців (MixColumns): Кожен стовпець форми обробляється поліноміальними операціями в полі Галуа (2^8). Ця операція множить стовпці на фіксований багаточлен x4 + 1 (по модулю) посилюючи зв’язки між байтами всередині стовпців.

4. Додавання ключа (AddRoundKey): На кожному етапі застосовується операція XOR між поточною формою і унікальним ключем, розрахованим для даного раунду.

5. Розширення ключа (Key Expansion): Початковий ключ шифрування перетворюється в набір ключів для кожного раунду, створюючи унікальні ключі, що ускладнює криптоаналіз.

Алгоритм Rijndael має модульну структуру, що дозволяє вносити зміни для підвищення стійкості до атак, зокрема нових технологій. Це робить його суттєвим покращенням у порівнянні з застарілими шифрами. [6]

1.3.2 Асиметричне шифрування

Асиметрична криптографія, або криптографія з відкритим ключем, використовує два пов'язаних, але різних ключі — відкритий і закритий. Ці ключі є великими числами, пов'язаними математичною залежністю, але не ідентичні.

- Відкритий ключ доступний усім і може вільно розповсюджуватися.
- Закритий ключ зберігається в таємниці і відомий лише власнику.

Будь-який з цих ключів може використовуватися для шифрування даних, а для їх розшифровки знадобиться протилежний ключ з пари. Якщо повідомлення зашифроване відкритим ключем, для його розшифровки використовується закритий ключ, і навпаки.

Багато протоколів, таких як SSH, OpenPGP, S/MIME та SSL/TLS, використовують асиметричну криптографію для забезпечення функцій шифрування та цифрового підпису. Ця технологія також застосовується в веб-браузерах для встановлення захищеного з'єднання через небезпечні мережі, наприклад, в Інтернеті, і для перевірки автентичності цифрових підписів.

Надійність шифрування тісно пов'язана з довжиною ключа: збільшення довжини ключа експоненційно підвищує його стійкість, однак це також уповільнює роботу системи. Зі зростанням обчислювальних потужностей і появою більш ефективних алгоритмів факторизації ймовірність зламу криптографічних систем зростає. Тому необхідно продовжувати збільшувати довжину ключів для підтримання потрібного рівня безпеки.

Для забезпечення конфіденційності, цілісності, автентичності та відмовостійкості під час асиметричного шифрування важливо переконатися, що відкритий ключ справжній, належить заявленому власнику і не був змінений або підроблений зловмисниками. Однак ідеального рішення проблеми автентифікації з використанням відкритих ключів не існує.

Найпоширеніший метод автентифікації — інфраструктура відкритих ключів (PKI). Довірені сертифікаційні центри (ЦС) сертифікують право власності на пари ключів і видають сертифікати, які підтверджують автентичність відкритих ключів. На відміну від PKI, системи шифрування, такі як OpenPGP, засновані на моделі Pretty Good Privacy (PGP), використовують децентралізовану модель

автентифікації, звану веб-службою довіри. У цій моделі користувачі самостійно підтверджують зв'язки між відкритими ключами та їхніми власниками через індивідуальні підписи, що дозволяє будувати довіру в мережі без участі центральних організацій.

Уітфілд Діффі та Мартін Хеллман, дослідники Стэнфордського університету, вперше публічно запропонували концепцію асиметричного шифрування у своїй статті 1977 року під назвою «Нові напрямки в криптографії». Однак до цього в секреті концепцію асиметричного шифрування запропонував Джеймс Елліс, який працював у британській розвідувальної організації GCHQ (Government Communications Headquarters). Алгоритм асиметричного шифрування, описаний Діффі та Хеллманом, використовує спеціальні числа для створення ключів дешифрування, що стало основою для подальших досліджень і розвитку криптографічних методів.

RSA (Rivest-Shamir-Adleman) — один з найширше використовуваних асиметричних алгоритмів, інтегрованих у протокол SSL/TLS для забезпечення безпеки зв’язку в комп'ютерних мережах. Основна безпека RSA ґрунтується на обчислювальній складності факторизації великих цілих чисел, які є добутком двох великих простих чисел. Множення цих простих чисел відбувається швидко, але завдання знаходження оригінальних простих чисел за їхнім добутком є надзвичайно складним, що й забезпечує безпеку криптографії з відкритим ключем.

Час, необхідний для факторизації добутку двох великих простих чисел, вважається настільки довгим, що перевищує можливості більшості нападників, за винятком держав з потужними обчислювальними ресурсами. Зазвичай RSA-ключі мають довжину 1024 або 2048 біт. Однак експерти вважають, що найближчим часом 1024-бітні ключі можуть стати вразливими для атак. Тому уряд і промисловість переходять на мінімальну довжину ключа 2048 біт для забезпечення більшої безпеки. [7]

Еліптична крива криптографії (ECC) стає все більш популярною серед експертів з безпеки як альтернатива RSA для реалізації криптографії з відкритим ключем. ECC використовує теорію еліптичних кривих для створення швидших, компактніших і ефективніших криптографічних ключів. Процес генерації ключів в ECC заснований на властивостях рівнянь еліптичних кривих.

Для зламу ECC потрібно обчислити дискретний логарифм еліптичної кривої, що є значно складнішою задачею, ніж факторизація, яка використовується в RSA. Завдяки цьому ключі ECC можуть бути значно коротшими, ніж ключі RSA, при забезпеченні еквівалентного рівня безпеки. Це призводить до меншого споживання обчислювальних ресурсів та батареї, що робить ECC особливо придатною для мобільних додатків і пристроїв з обмеженими ресурсами порівняно з RSA.

Цифрові підписи, засновані на асиметричній криптографії, забезпечують гарантії щодо походження, ідентифікації та статусу електронних документів, транзакцій або повідомлень, а також підтверджують проінформовану згоду підписанта. Для створення цифрової підписи програмне забезпечення, наприклад, поштовий клієнт, генерує односторонній хеш даних, які потрібно підписати. Цей хеш потім шифрується за допомогою закритого ключа користувача, що повертає унікальне значення, що відповідає хешованим даним. Зашифрований хеш, разом з додатковою інформацією, такою як алгоритм хешування, формує цифрову підпис.

Будь-яка зміна даних, навіть в одному біті, призводить до зміни значення хешу, що дозволяє перевіряти цілісність даних. Для перевірки цілісності використовується відкритий ключ підписанта, за допомогою якого розшифровується хеш. Якщо розшифрований хеш збігається з обчисленим хешем оригінальних даних, це підтверджує, що дані не були змінені після підписання. Якщо хеші не збігаються, це вказує на порушення цілісності даних або на те, що підпис була створена за допомогою закритого ключа, що не відповідає відкритому ключу підписанта, що свідчить про порушення автентифікації.

Цифровий підпис також забезпечує стійкість до відмови, що означає, що підписана сторона не може відмовитися від підписання документа. Якщо підписант намагається заперечувати дійсність свого підпису, це може свідчити про те, що або їхній закритий ключ був скомпрометований, або вони надають неправдиву інформацію. У деяких країнах цифрові підписи мають юридичну силу, аналогічну традиційним підписам на паперових документах, що дозволяє використовувати їх у юридичних та офіційних транзакціях. [8]

Протокол Діффі – Хеллмана

Початкова публікація алгоритму Діффі-Хеллмана з'явилася в 1970-х роках в статті Вітфілда Діффі та Мартіна Хеллмана, де були введені основні концепції криптографії з відкритим ключем. Алгоритм Діффі-Хеллмана не призначений для шифрування даних або створення електронного підпису, а слугує для розподілу ключів. Він дозволяє двом або більше користувачам обмінюватися ключем для симетричного шифрування без необхідності використання посередників. Цей алгоритм став першим в історії криптосистем, який забезпечував захист інформації без передачі секретних ключів по захищених каналах. Схема відкритого розподілу ключів, запропонована Діффі та Хеллманом, справила революцію в області шифрування, вирішивши ключову проблему класичної криптографії — безпечний обмін ключами.

Алгоритм Діффі-Хеллмана базується на складності обчислення дискретних логарифмів. У цьому алгоритмі, як і в інших алгоритмах з відкритим ключем, обчислення проводяться по модулю великого простого числа P. Спочатку вибирається натуральне число A, яке менше P. Щоб зашифрувати значення X, обчислюється:

$$Y = A^x \mathrm{mod} \qquad (1.1)$$

де X — секретне значення, вибране відправником, а Y — результат, який передається отримувачу.

Отримувач, використовуючи свій секретний ключ та відкритий параметр A, може обчислити спільний секрет, не передаючи його безпосередньо.

Якщо відоме значення X, обчислити Y доволі просто. Однак зворотне завдання — знайти X за значенням Y — є складною задачею, відомою як дискретний логарифм. Завдяки складності обчислення дискретного логарифма, значення Y можна безпечно передавати через незахищені канали зв'язку. При достатньо великому значенні модуля P, задача знаходження X стає практично неможливою. Цей принцип лежить в основі алгоритму Диффі-Хеллмана для формування ключа.

Припустимо, два користувача — користувач 1 і користувач 2 — хочуть створити спільний ключ для симетричного шифрування. Для цього їм потрібно вибрати велике просте число P і деяке число A, яке відповідає умові $1 < A < P-1$. При цьому всі числа з діапазону [1, 2, ..., P-1] повинні бути представлені як різні ступені A по модулю P. Ці параметри, відомі всім учасникам системи, можуть бути вибрані відкрито і є спільними для всіх.

Потім користувач 1 обирає число X_1, яке менше P, і генерує його за допомогою випадкових чисел. Це число стане закритим ключем першого користувача, і його необхідно зберігати в секреті. Використовуючи цей закритий ключ, користувач 1 обчислює значення:

$$Y_1 = A^{x_1} \, mod \, P \tag{1.2}$$

який він надсилає другому користувачу. Другий абонент робить те ж саме, генеруючи X_2 і обчислюючи

$$Y_2 = A^{x_2} \, mod \, P \tag{1.3}$$

Цей результат відправляється першому користувачеві.

Після цього у користувачів є наступна інформація:

Таблиця 1.1

Параметри ключів шифрування

	Загальні параметри	Відкритий ключ	Закритий ключ
1-й користувач	P, A	Y_1	X_1
2-й користувач		Y_2	X_2

З чисел Y_1 і Y_2, а також з особистих закритих ключів кожен користувач може згенерувати спільний секретний ключ Z для сеансу симетричного шифрування. Перший користувач:

$$Z = (Y_2)^{x_1} \, mod \, P \tag{1.4}$$

Ніхто, крім першого користувача, не може цього зробити, оскільки число X_1 є секретним. Другий користувач може отримати таке ж число Z, використовуючи свій закритий ключ та відкритий ключ користувача 1:

$$Z = (Y_1)^{x_2} \, mod \, P \tag{1.5}$$

Якщо протокол формування спільного секретного ключа виконано правильно, то значення Z, отримані обома абонентами, збігатимуться. Важливо, що зловмисник, не знаючи секретних чисел X_1 і X_2, не зможе обчислити Z. Навіть якщо він спробує обчислити Z, маючи лише відкриті параметри P, A, Y_1 і Y_2, йому це не вдасться.

Безпека алгоритму Диффі-Хеллмана ґрунтується на складності обчислення дискретних логарифмів, попри те, що обчислення експонент за модулем простого числа відносно просте. Для великих простих чисел, які містять сотні і тисячі біт, задача стає практично невирішуваною через величезні обчислювальні витрати.

В результаті перший і другий користувачі можуть використовувати обчислене значення Z як спільний секретний ключ для шифрування та розшифрування даних.

Таким чином, будь-яка пара абонентів може обчислити секретний ключ, відомий тільки їм. [9]

Алгоритм шифрування RSA

RSA — це криптографічний алгоритм з відкритим ключем, який базується на складності задачі факторизації великих цілих чисел. Він став першим алгоритмом, що використав відкриту криптографію. Назва алгоритму походить від перших літер прізвищ його творців — Рональда Рівеста, Аді Шаміра та Леонарда Адлемана, трьох вчених з Массачусетського технологічного інституту.

Після ознайомлення з опублікованою в 1976 році статтею Уїтфілда Діффі та Мартіна Хеллмана «Нові напрямки в криптографії», в якій були викладені основи криптографії з відкритим ключем, Рональд Рівест, Аді Шамір і Леонард Адлеман почали шукати математичну функцію, здатну реалізувати запропоновану систему. Після розгляду більш ніж 40 можливих варіантів вони розробили алгоритм, оснований на складності знаходження великих простих чисел і труднощах розкладання добутку таких чисел на множники.

Для шифрування в RSA використовується операція піднесення до степеня за модулем N. Для розшифрування потрібно обчислити функцію Ейлера від числа N, що вимагає знання розкладу числа N на прості множники (задача факторизації). У системі RSA відкриті та закриті ключі складаються з пари цілих чисел. Закритий ключ зберігається в секреті, а відкритий ключ надається іншому учаснику або публікується для використання в процесі шифрування. [10]

Система базується на наступних фактах:

- При відомих числах b і d обчислення числа a з порівняння:

$$a \equiv b^d \ mod \ n.$$

(2.1) По складовому модулю n – це проста задача;

- Визначення невідомого числа b за відомими числами d і a з порівняння по складовому модулю n:

$$a \equiv b^d \ mod \ n \tag{2.2}$$

є складним завданням;

- Якщо відомо, що p і q прості числа і n = pq, то обчислити n легко, а знайти розклад n на прості множники важко;
- Якщо відомо розклад n = pq на прості множники, то задача обчислення числа b з рівняння:

$$A \equiv b^d \ mod \ n \tag{2.3}$$

виконувана.

Теоретичною основою криптосистеми RSA є теорема Ейлера: для будь-яких натуральних і взаємно простих чисел n і a справедливе рівняння:

$$a\varphi^{(n)} \equiv 1 \ mod \ n. \tag{2.4}$$

Тут $\varphi^{(n)}$ є функція Ейлера – кількість взаємно простих з n натуральних чисел від 1 до n.

З теорії чисел відомо, що якщо p і q прості числа, а n = pq, то

$$\varphi^{(n)} = p - 1 \ q - 1. \tag{2.5}$$

Крім того, з теореми Ейлера випливає, що якщо певне число e взаємно просте з $\varphi^{(n)}$, то рівняння:

$$de \ \equiv \ 1 \ mod \ \varphi^{(n)} \ , \tag{2.6}$$

або інакше

$$de \ \equiv \ k \ \varphi^{(n)} + 1 \ , \tag{2.7}$$

однозначно дозволяється щодо d. Рішення легко визначається розширеним алгоритмом Евкліда.

Отже, якщо відомо, що:

$$de \equiv 1 \, mod \, \varphi^{(n)} \tag{2.8}$$

а x – передавана інформація, то справедливо рівність:

$$(x^e)^d \bmod n \equiv x^{k\varphi^{(\mathrm{n})}+1} \, mod \, n \equiv x^{\varphi^{(\mathrm{n})}k} \, x \, mod \, n \equiv x \tag{2.9}$$

Фактично, останнє співвідношення є основою для формулювання системи RSA.

Принцип роботи системи RSA

Спочатку відбувається генерація пари ключів – відкритого і закритого.

Генерація здійснюється наступним чином:

1. Вибираються two простих великих чисел p і q, при цьому вони не рівні.

2. Обчислюється модуль числа:

$$N = p * q. \tag{3.1}$$

3. Обчислюється значення функції Ейлера від модуля числа N:

$$\varphi \, N = p - 1 * q - 1 \tag{3.2}$$

4. Вибирається деяке число e – відкрита експонента – яке лежить у інтервалі 1 < q < $\varphi \, N$ і є взаємно простим зі значенням функції $\varphi \, N$

5. Обчислюється число d – секретна експонента. Причому воно є мультиплікативно оберненим до числа e за модулем $\varphi \, N$

$$d * e = 1 \, mod \, \varphi \, N \tag{3.3}$$

В результаті роботи системи RSA користувачі отримують пару ключів: (e, N) — відкритий ключ і (d, N) — закритий ключ.

Коли користувач A та користувач B обмінюються повідомленнями в інтернеті, вони використовують ці ключі для забезпечення конфіденційності переписки. Користувач B заздалегідь генерує пару ключів та передає свій відкритий ключ користувачу A. Отримавши відкритий ключ, користувач A використовує його для

шифрування повідомлення перед відправкою, щоб тільки користувач B, маючи закритий ключ, міг розшифрувати це повідомлення.

Шифрування: Користувач A шифрує повідомлення m за допомогою відкритого ключа другого користувача (e, N) і надсилає його:

$$c = e\ m = m^e\ mod\ (N) \tag{3.4}$$

Розшифрування: Прийнявши зашифроване повідомлення, користувач B розшифрує його, використовуючи закритий ключ (d, N):

$$m = d\ c = c^d\ mod\ (N) \tag{3.5}$$

Рис. 1.1 – Принцип роботи RSA

Приклад шифрування и розшифрування RSA

Потрібно зашифрувати повідомлення «RSA». Позначимо кожну букву їхніми порядковими номерами в англійському алфавіті. R - 18, S - 19, A - 1. Далі дотримуємося алгоритму:

1. Вибираємо прості числа (для простоти обчислень візьмемо невеликі): p = 3, q = 11.

2. Обчислюємо модуль N:

$$N = p * q = 3 * 11 = 33. \quad (4.1)$$

3. Знаходимо функцію Ейлера від модуля числа N:

$$\varphi\, N = 3 - 1 * 11 - 1 = 2 - 10 = 20. \quad (4.2)$$

4. Обираємо відкриту експоненту: e = 7.

5. Обчислюємо відкриту експоненту:

$$d * \ 7 = 1 \ mod\ (20), d = 3. \quad (4.3)$$

Отриманим відкритим ключем (7,33) шифруємо кожну літеру вихідного повідомлення:

$$\begin{aligned} &\mathrm{c1} = 18^7\ mod\ 33 = 6; \\ &\mathrm{c2} = 19^7\ mod\ 33 = 13; \\ &\mathrm{c3} = 1^7\ mod\ 33 = 1. \end{aligned} \quad (4.4)$$

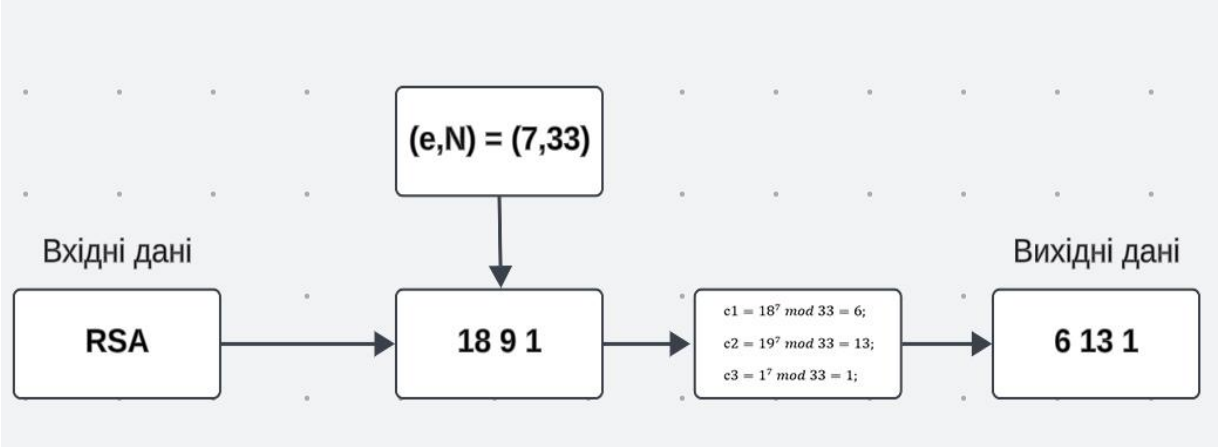


Рис. 1.2 – Приклад шифрування RSA

Щоб розшифрувати отримане повідомлення, використовуємо закритий ключ (3, 33):

$$\begin{aligned} &\mathrm{m1} = 6^3\ mod\ 33 = 18; \\ &\mathrm{m2} = 13^3\ mod\ 33 = 19; \\ &\mathrm{m3} = 1^3\ mod\ 33 = 1. \end{aligned} \quad (4.5)$$

Вийшло вихідне повідомлення. [11]

1.4 Формування вимог до нової технології

Застосування блокчейн-платформи Ethereum значно підвищує безпеку месенджерів. Завдяки сучасним технологіям, підібрати секретний ключ Ethereum майже неможливо, оскільки це вимагає величезних часових і обчислювальних ресурсів.

Децентралізоване зберігання даних на блокчейні запобігає втручанню третіх осіб, що робить цей метод безпечнішим, ніж централізовані системи зберігання, як-от хмарні сервіси та сервери. Цей підхід передбачає єдине шифрування повідомлень з використанням end-to-end алгоритму, усуваючи необхідність поділу чатів на хмарні та зашифровані. Усі дані, включно з повідомленнями, фотографіями, документами та іншими файлами, захищаються одним видом шифрування і залишаються доступними в реальному часі з будь-якого пристрою. Також можливе повне відновлення всієї історії листування і раніше відправлених даних.

Для шифрування використовується алгоритм RSA, який використовує пару ключів: відкритий і закритий. Ця криптосистема має високий ступінь надійності та стійкості до криптоаналізу, що робить її однією з найбезпечніших серед наявних рішень.

РОЗДІЛ 2. РОЗРОБКА БЕЗПЕЧНОЇ СИСТЕМИ ПЕРЕДАЧІ ДАНИХ В ІНТЕРНЕТІ

2.1 Блокчейн технологія Ethereum

Більшість особистих персональних даних, паролів та фінансової інформації зберігаються на серверах та хмарних системах, що належать великим компаніям, таким як Amazon, Facebook, Apple та Google. Такий підхід до зберігання даних має свої переваги. Ці компанії вкладають значні кошти в інформаційну безпеку, створюючи команду експертів для захисту даних, моніторингу та усунення потенційних загроз. Крім того користувачі також звільняються від необхідності самостійно забезпечувати хостинг і піклуватися про доступність своїх даних, оскільки це завдання повністю лягає на інфраструктуру сервісів, що надаються.

Попри зручності централізованого зберігання даних, цей підхід має значний недолік - уразливість. Зловмисники можуть отримати несанкціонований доступ до файлів користувачів, атакуючи або експлуатуючи сторонні сервіси. Це може призвести до витоку, крадіжки або зміни важливої інформації. Творець веб-сервера Apache, Брайан Белендорф, назвав централізоване зберігання даних «первородним гріхом» Інтернету.

На думку таких фахівців, як Белендорф, Інтернет спочатку замислювався як децентралізована структура. Це породило рух зі створення інструментів, таких як блокчейн, для втілення цього ідеалу. Серед таких технологій Ethereum виділяється як одна з найпрогресивніших. Її мета - замінити традиційних посередників в Інтернеті, включно з тими, хто відповідає за зберігання даних, передачу іпотечних кредитів і управління фінансовими інструментами.

Ethereum прагне стати «світовим комп'ютером», який перетворює поточну модель клієнт-сервер на децентралізовану систему. У цій моделі сервери і хмарні сховища замінюються тисячами розподілених вузлів, керованих волонтерами по всьому світу, що робить мережу стійкою і демократичною.

Технологія Ethereum пропонує створити глобальну інфраструктуру, що дасть змогу користувачам з усього світу конкурувати за надання послуг на її основі. У сучасному світі додатки - від банківських сервісів до месенджерів і фітнес-програм - залежать від централізованих організацій, які зберігають дані користувачів, як-от інформація про кредитні картки, історія покупок та інші особисті дані, на своїх серверах.

Ethereum пропонує децентралізовану альтернативу, за якої жодна компанія або організація не зможе контролювати або блокувати додатки, що працюють на блокчейні. Усі зміни здійснюватимуться виключно користувачем, що виключає втручання третіх сторін.

Ця технологія поєднує контроль над збереженням, зміною та видаленням даних зі зручністю їхнього доступу в цифрову епоху. Щоразу, коли вносяться зміни, додається або видаляється інформація, оновлення синхронізуються на кожному вузлі мережі, забезпечуючи консенсус і безпеку даних.

Ethereum являє собою адаптовану і гнучку блокчейн-платформу з відкритим вихідним кодом, яка дає змогу розробляти і використовувати децентралізовані додатки (DApps). Ця система вважається однією з найскладніших і найпередовіших технологій на сьогоднішній день. [12]

Незважаючи на складність механізмів і протоколів, Ethereum можна розділити на три ключові компоненти:

- Блокчейн - децентралізоване сховище даних, де фіксуються всі транзакції;
- Однорангова мережа (peer-to-peer) - інфраструктура для зв'язку і синхронізації між вузлами;
- Віртуальна машина Ethereum (EVM) - універсальне обчислювальне середовище для виконання смарт-контрактів.

2.1.1 Блокчейн

Блокчейн являє собою послідовність блоків, упорядкованих і пов'язаних

таким чином, що зміна будь-якого з них руйнує всі наступні. Кожен блок містить записи про транзакції, хеш попереднього блоку і доказ виконання роботи (proof-of-work) для поточного блоку.

Доказ виконання роботи - це процес знаходження спеціального числа (nonce), яке разом із вмістом блоку генерує певний хеш. Цей хеш має відповідати заздалегідь заданій умові, наприклад, починатися з певної кількості нулів. Що вища складність, то більше обчислювальних ресурсів необхідно для знаходження відповідного хешу.

Оскільки кожен блок містить хеш попереднього блоку, зміна будь-якого блоку вимагає перерахунку й оновлення хешів усіх наступних блоків. Це робить підробку даних вкрай складною, оскільки вимагає перерахунку всього ланцюга. Навіть якщо хтось зможе внести зміни, переконати мережу прийняти підроблений ланцюжок практично неможливо через принципи консенсусу.

Блокчейн Ethereum функціонує як машина станів, де кожна зміна стану відбувається через транзакції. Як і будь-яка машина станів, Ethereum зчитує вхідні дані, обробляє їх і переходить у новий стан на основі цієї інформації.

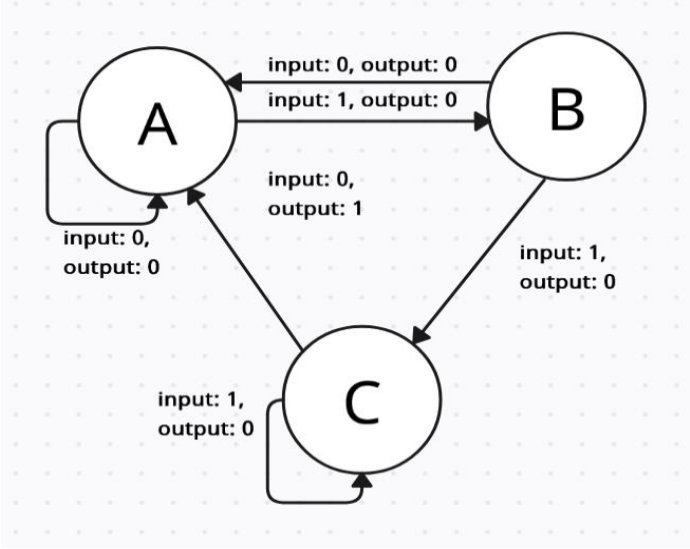


Рис. 2.1 – Парадигма блокчейну Ethereum

У контексті машини станів Ethereum початковою точкою є "стан генезису". Цей стан можна порівняти з чистим аркушем, системою, яка ще не містить записів про будь-які транзакції або дії.

У міру виконання транзакцій у мережі стан генезису перетворюється і переходить у новий стан. Кожен новий стан формується на основі правил протоколу Ethereum і враховує дані всіх попередніх транзакцій.

У будь-який момент поточний стан мережі Ethereum є результатом усіх попередніх змін і транзакцій. Цей стан являє собою «актуальний знімок» усієї інформації в блокчейні, включно з даними про баланси, контракти та інші параметри.

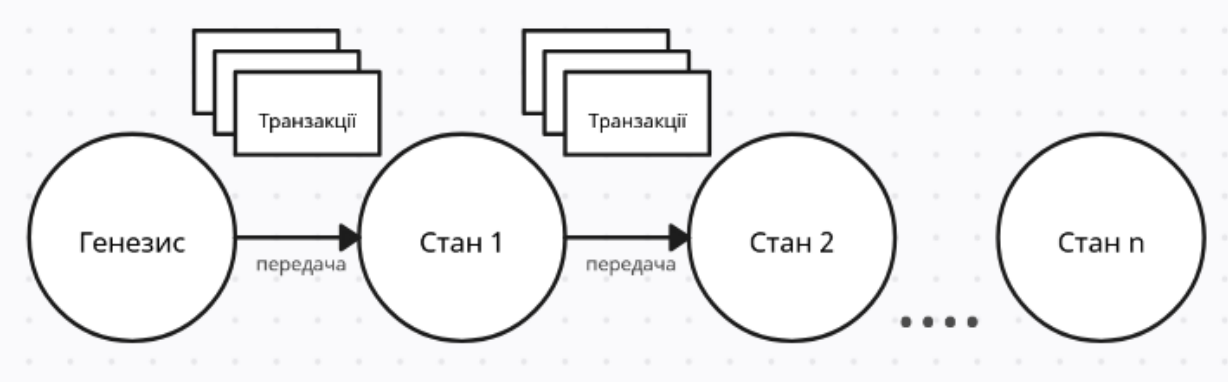


Рис. 2.2 – Стан блокчейну

Стан Ethereum включає безліч транзакцій, які структуруються у вигляді «блоків». Кожен блок являє собою набір транзакцій, оброблених і підтверджених мережею в рамках одного часового інтервалу.

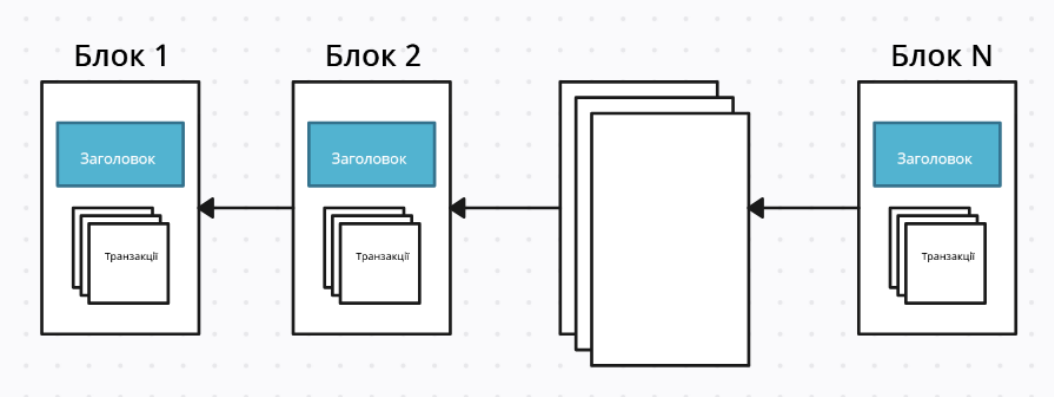


Рис. 2.3 – Блоки транзакцій

Для переходу мережі Ethereum з одного стану в інший, транзакції повинні пройти процес перевірки, відомий як валідація. Цей процес здійснюється через майнінг, під час якого вузли мережі (ноди) використовують обчислювальні ресурси для створення блоку, що містить дійсні транзакції.

Будь-який вузол, який заявляє себе як майнер, може брати участь у створенні та підтвердженні нового блоку. Одночасно безліч нод по всьому світу намагаються створити і затвердити блоки, змагаючись один з одним. Щоб блок було додано в блокчейн, майнер надає математичний доказ, що підтверджує його справжність. Цей доказ слугує гарантом того, що блок відповідає правилам мережі.

Процес перевірки та додавання блоку заснований на механізмі доказу виконання роботи (proof-of-work), що вимагає значних обчислювальних зусиль. Майнер, який першим підтвердив блок, додає його до ланцюжка, тим самим оновлюючи загальний стан мережі. [13]

Оскільки блокчейн являє собою єдиний механізм запису транзакцій та їхнього поточного стану, важливо підтримувати єдину версію ланцюжка, визнану всіма учасниками мережі. Розгалуження (forks) - ситуації, за яких існує кілька конкурентних ланцюжків - створюють плутанину і загрожують цілісності системи. В ідеалі мережа прагне уникати форків, щоб користувачі не опинилися перед вибором, якому ланцюжку довіряти.

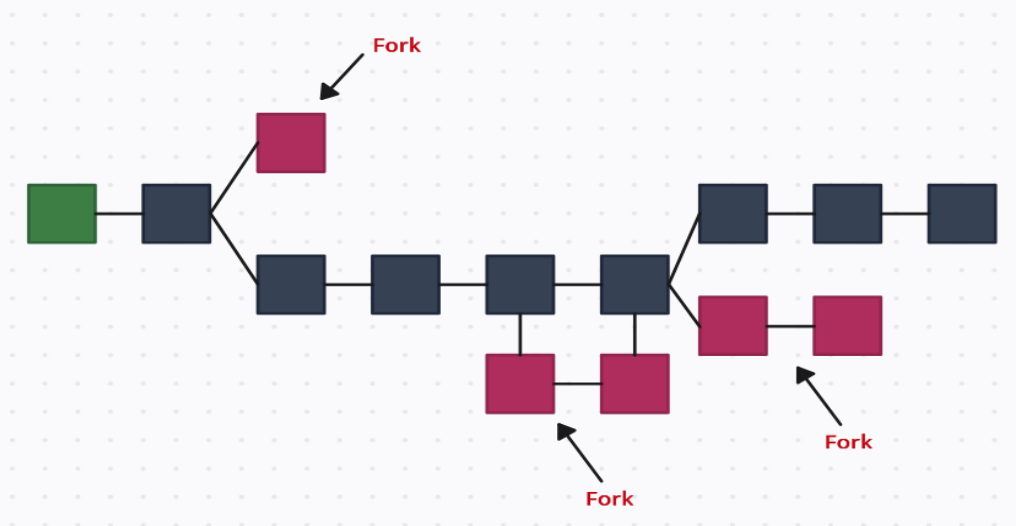


Рис. 2.4 – Розгалуження

Для запобігання появи декількох ланцюжків і визначення найбільш дійсного шляху в Ethereum застосовується механізм під назвою протокол GHOST (Greedy Heaviest Observed Subtree).

Протокол GHOST заснований на виборі ланцюжка, в якому було витрачено найбільшу кількість обчислювальних ресурсів. Це визначається за довжиною ланцюжка, що вимірюється номером останнього блоку. Номер блоку показує загальну кількість блоків у цьому ланцюжку (за винятком генезисного блоку).

Що більший номер останнього блоку, то довший ланцюг і то більше зусиль було докладено майнерами для його створення. Таким чином, вибір ланцюжка з найбільшим номером блоку дає змогу мережі узгодити єдину версію поточного стану блокчейна, уникаючи розгалужень.

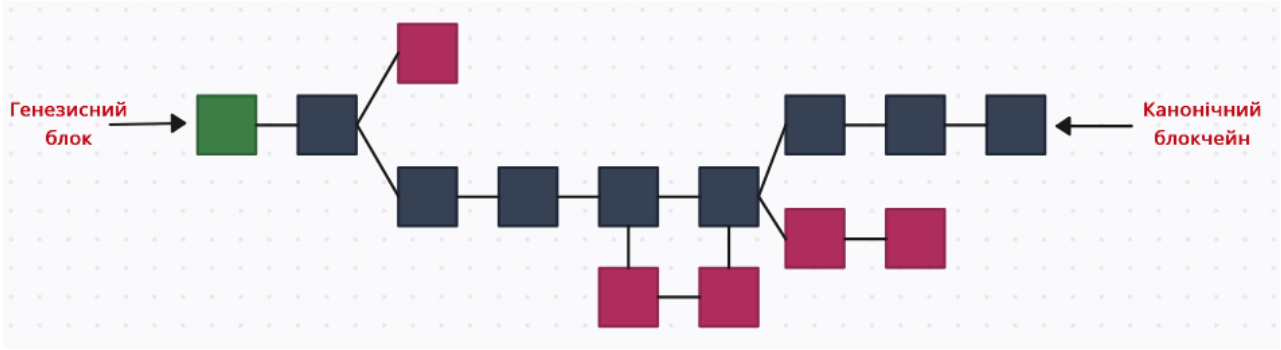


Рис. 2.5 – Канонічний стан

2.1.2 Смарт-контракти

Смарт-контракт - це програмний алгоритм, який регулює виконання зобов'язань сторін під час обміну активами в блокчейн-системі. Це угода, яку автоматично виконує комп'ютер.

Смарт-контракти являють собою закодовані логіки, які керують зміною цифрових активів на основі виконання певних умов. Вони працюють за принципом «якщо умову виконано, то відбудеться дія», де «якщо» - це необхідні вимоги, а «потім» - дія, що виконується в разі їх дотримання. Коли смарт-контракт запущено в блокчейні, його виконання починається автоматично при виконанні всіх заданих умов.

Перевірка і виконання смарт-контрактів забезпечуються безліччю комп'ютерів, підключених до мережі, що гарантує, що такі контракти:

- безпечні,
- відкриті,
- заслуговують на довіру,
- мінімізують можливість помилок.

Однак виконання операцій через мережу блокчейна вимагає участі всіх вузлів, що робить процес енерговитратним і дорогим.

Багато блокчейн-платформ мають обмежений функціонал. Але Ethereum відрізняється завдяки своїй гнучкості. Смарт-контракти в Ethereum дають змогу розробникам створювати і запускати практично будь-які операції, що уможливлює розробку універсальних додатків. [13]

Віртуальна машина Ethereum (EVM) - це програмне забезпечення, повне за Тьюрингом, здатне виконувати будь-який додаток. Унікальність Ethereum полягає в тому, що кожна операція всередині мережі виконується всіма вузлами одночасно. Така система прискорює і спрощує створення смарт-контрактів, роблячи процес продуктивнішим.

Особливістю смарт-контрактів Ethereum є те, що вони отримують унікальну адресу в блокчейні після створення. Код не вставляється в кожен контракт, замість цього запускається спеціальна транзакція, яка закріплює за ним унікальну адресу. Після створення смарт-контракт стає невід'ємною частиною блокчейна і працюватиме доти, доки операція не завершиться успішно.

2.2 Реалізація системи

2.2.1 Створення ключів та їх використання для транзакцій

Ідентифікатор Ethereum являє собою об'єкт, що містить приватний ключ, відповідний відкритий ключ і його адресу. Для створення нового ідентифікатора використовується функція “createIdentity”, яка генерує і повертає цей об'єкт. Ця

функція створює нову пару ключів (приватний і відкритий), а також обчислює адресу, пов'язану з відкритим ключем. Приватний ключ залишається в секреті у користувача, тоді як відкритий ключ і адреса можуть бути використані для проведення транзакцій у мережі Ethereum.

```
const EthCrypto = require('eth-crypto');

const identity = EthCrypto.createIdentity();

console.dir(identity);
/* > {
    address: '0xA1B2C3D4E5F67890A123B4C5D6E7F89A0BCDEF12',
    privateKey: '0x9a1b2c3d4e5f67890ab1c2d3e4f56789a0b1c2d3e4f5b67890abcd1234ef5678',
    publicKey: 'f12a34b56c78d90e12f34a56b78c90d12f34a56b78c90d12e34a56b78c90a1bc...'
} */
```

Рис. 2.6 – Створення нового ідентифікатора

Компоненти ідентифікатора Ethereum включають:

1. Приватний ключ (privateKey) - це секретний ключ, який повинен залишатися конфіденційним. Він використовується для підписання та розшифрування повідомлень, а також для генерації відповідного відкритого ключа.

2. Відкритий ключ (publicKey) - цей ключ стає доступним, коли дані підписуються з використанням приватного ключа. Відкритий ключ зазвичай передається іншим людям, щоб вони могли зашифрувати дані, які можна буде розшифрувати тільки за допомогою відповідного приватного ключа. Відкритий ключ може бути представлений у двох форматах: стислому і нестислому. У випадку з Ethereum створюється нестислий ключ, який починається з 0x04, а стислі ключі починаються з 0x02 або 0x03. Коли ключ представлений, початок 0x04 додається до ключа внутрішньо, у процесі криптографічних операцій.

3. Адреса - це результат застосування функції keccak-256 хешування до останніх 20 байтів відкритого ключа. Адреса використовується для представлення ідентифікатора в блокчейні. Адресу не можна використовувати для обчислення відкритого ключа. Існує два типи подання адреси: звичайна адреса в рядковому

регістрі (просто 20 байтів хеша) і формат із контрольною сумою, де використовуються великі літери для мінімізації помилок під час введення адреси вручну. [14]

Транзакція Ethereum здебільшого являє собою json-об'єкт із певними значеннями.

```
const rawTransaction = {
    from: '0xB1C2D3E4F5A67890A123B4C5D6E7F89A0BCDEF12', // адрес відправника
    to: '0x7F4A052378EA5E0149F9E98FAD47A4B8AC5D1211', // адрес отримувача
    value: 500000000000000000, // кількість wei, яку ми хочемо надіслати (=0.5 ефіру)
    nonce: 1, // інкрементне tx-число, додаємо +1 для кожної транзакції
    gasPrice: 10000000000, // ціна газу для транзакції
    gasLimit: 21000 // стандартний gasLimit для звичайних транзакцій
};
```

Рис. 2.7 – Створення транзакції

Перш ніж транзакцію може бути надіслано на вузол, її потрібно підписати за допомогою privateKey і перевести в шістнадцятковий рядок.

```
const serializedTx = EthCrypto.signTransaction(
  rawTransaction,
  identity.privateKey
);
console.log(serializedTx);
// > '0x8f8462a5b39eb8c4b3543b1d9a0d3dcd6b9a9833b05d6a7f4c2e0c35bbf3c2a9'
```

Рис. 2.8 – Підпис з privateKey

Тепер рядок транзакції може бути відправлений у блокчейн. Для цілей тестування створюється локальний тестовий ланцюжок і перевіряється там. Щоб створити локальну тестову мережу, використовується ganache-cli і підключається до екземпляра web3, щоб взаємодіяти з ним.

```
const Web3 = require('web3');
const ganache = require('ganache-cli');

// створення екземпляра web3
const web3 = new Web3();

// створення провайдера ganache
const ganacheProvider = ganache.provider({
  // встановлення балансу ідентифікатора на 20 ефірів
  accounts: [{
    secretKey: '0x1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef',
    balance: web3.utils.toWei('20', 'ether')
  }]
});

// встановити ganache на web3 як провайдер
web3.setProvider(ganacheProvider);
```

Рис. 2.9 – Створення локальної тестової мережі

sendSignedTransaction викликається, щоб надіслати підписану транзакцію в тестовий ланцюжок. Ganache негайно виконає транзакцію, і квитанція повернеться назад. Щоб переконатися, що транзакція спрацювала, перевіримо баланс адреси одержувачів.

```
const receipt = await web3.eth.sendSignedTransaction(serializedTx);
const balance = await web3.eth.getBalance('0x7F4A052378EASE0149F9E98FAD47A4B8AC5S1211');
console.log(balance); // > '5000000000000000000'
```

Рис. 2.10 – Надсилання транзакції

2.2.2 Підпис і перевірка даних за допомогою Solidity

Для того щоб підписати дані за допомогою JavaScript і перевірити підпис у смарт-контракті Solidity, спочатку необхідно створити два ідентифікатори: один для творця та інший для одержувача. Потім слід розгорнути локальну тестову мережу, в якій надається баланс 10 ефірів для творця і один ефір для одержувача, щоб забезпечити достатній газ для надсилання транзакцій.

Перед тим як почати взаємодіяти з локальним блокчейном, слід скомпілювати код Solidity у байт-код з використанням JavaScript-версії компілятора solc:

```
const path = require('path');
const SolidityCli = require('solidity-cli');
const contractPath = path.join(__dirname, '../../contracts/DonationBag.sol');
const compiled = await SolidityCli.compileFile(contractPath);
const compiledDonationBag = compiled[':DonationBag'];

const createCode = EthCrypto.txDataByCompiled(
    JSON.parse(compiledDonationBag.interface), // ABI
    compiledDonationBag.bytecode, // байткод
    [creatorIdentity.address] // аргумент конструктора
);

console.dir(compiledDonationBag);
```

Рис. 2.11 – Компіляція в байткод

Тепер, коли є байт-код контракту, ми можемо надіслати транзакцію для створення нового екземпляра в локальному тестовому ланцюжку.

```
// відправка в локальний ланцюг
const receipt = await web3.eth.sendSignedTransaction(serializedTx);
const contractAddress = receipt.contractAddress;

console.log(contractAddress);
```

Рис. 2.12 – Надсилання транзакції

Тепер, коли контракт розгорнуто в блокчейні, для перевірки його коректності можна викликати відповідну функцію. Перш ніж підписувати пожертвування, необхідно відправити деяке значення в контракт.

Під час підписання повідомлення ми не підписуємо безпосередньо адресу одержувача, а замість цього підписуємо хеш, який складається з узагальнених даних:

- Префікс (Prefix): Для захисту від випадкового підписання ненадійної транзакції в Ethereum ми додаємо унікальні дані, щоб гарантувати, що творця не обдурять.
- Адреса контракту (contractAddress)**: Якщо у творця є кілька екземплярів контракту, підписані дані можуть бути відтворені в різних екземплярах. Щоб уникнути такої атаки, ми додаємо адресу контракту в хеш.
- Адреса одержувача (receiverAddress): Підписуючи цю адресу, творець підтверджує, що саме ця адреса має отримати пожертву.

Тепер одержувач може надіслати підпис у контракт, щоб запросити пожертвування. Якщо все виконано правильно, баланс одержувача має збільшитися. [15]

```
const receiverBalance = await web3.eth.getBalance(recieverIdentity.address);
console.dir(receiverBalance);
// '2500000000000000000'
```

Рис. 2.13 – Перевірка балансу

2.2.3 Шифрування та підпис повідомлень

За допомогою Ethereum-ключів можна не тільки взаємодіяти з блокчейном, а й безпечно обмінюватися повідомленнями через ненадійні канали. Ідентифікатори Ethereum будуть використовуватися для надсилання повідомлень.

Для початку створюємо два ідентифікатори - користувача A і користувача B. Користувач A хоче надіслати повідомлення «Hello, B». Перш ніж надіслати його, потрібно гарантувати, що тільки користувач B зможе прочитати повідомлення, а також переконатися, що воно справді надійшло від користувача A. Для цього спочатку підписуємо повідомлення за допомогою приватного ключа користувача A, а потім зашифровуємо його, використовуючи публічний ключ користувача B. Це забезпечує безпеку та автентичність повідомлення.

```
const signature = EthCrypto.sign(
    A.privateKey,
    EthCrypto.hash.keccak256(secretMessage)
);

const payload = {
    message: secretMessage,
    signature
};

const encrypted = await EthCrypto.encryptWithPublicKey(
    B.publicKey,
    privateKey,
    JSON.stringify(payload)
);

const encryptedString = EthCrypto.cipher.stringify(encrypted);
```

Рис. 2.14 – Шифрування та підпис повідомлення

Коли одержувач приймає повідомлення, він починає з розшифрування його своїм секретним ключем, а потім перевіряє підпис.

```
const encryptedObject = EthCrypto.cipher.parse(encryptedString);

const decrypted = await EthCrypto.decryptWithPrivateKey(
    B.privateKey,
    encryptedObject
);

const decryptedPayload = JSON.parse(decrypted);

const senderAddress = EthCrypto.recover(
    decryptedPayload.signature,
    EthCrypto.hash.keccak256(payload.message)
);

console.log(
    'Повідомлення отримано від ' +
    senderAddress +
    ': ' +
    decryptedPayload.message
);
```

Рис. 2.15 – Розшифровка

Тепер, коли повідомлення підписане і зашифроване, користувач А може відправити його користувачеві В через ненадійний канал. Користувач В, отримавши зашифроване повідомлення, розшифровує його за допомогою свого приватного ключа і перевіряє підпис за допомогою публічного ключа користувача А. Таким чином, користувач В може бути впевнений, що повідомлення справді надійшло від користувача А і не було змінено на шляху.

РОЗДІЛ 3. РОЗРАХУНОК ВИТРАТ

3.1 Gas в мережі Ethereum

Gas - це внутрішня одиниця виміру, яка використовується в мережі Ethereum для оцінювання та оплати обчислювальних ресурсів, необхідних для виконання операцій і контрактів. Вона служить для визначення обсягу обчислювальної потужності, необхідної для обробки конкретної транзакції або операції. Оплата за обчислення стягується незалежно від результату транзакції. Навіть якщо транзакцію відхилено, вузли мережі зобов'язані виконати обчислення, за які отримують винагороду, забезпечуючи оплату їхньої роботи незалежно від успішності транзакції.

Ціна Gas визначається виходячи з архітектури Ethereum, заснованої на Тьюринг-повній віртуальній машині (EVM). Основна мета - запобігти нескінченним циклам, здатним перевантажити мережу. Наприклад, 1 Gas еквівалентний виконанню однієї команди або рядка коду. Якщо у користувача недостатньо ефіру для покриття вартості транзакції, вона визнається недійсною. Ця система знижує ймовірність атак, спрямованих на відмову в обслуговуванні, підвищує ефективність коду і змушує зловмисників оплачувати ресурси, які вони використовують, включно з пропускною спроможністю, обчислювальною потужністю та обсягом сховища даних. [16]

Що більше обчислювальних операцій потрібно, то вищі витрати Gas. Так, для простої операції, такої як переказ 1 Ether від одного користувача до іншого, відправнику потрібно буде сплатити суму трохи більшу за 1 Ether, включно з комісією за Gas. Однак якщо користувач планує укласти контракт, що залежить від майбутньої вартості Ether, обсяг коду, який необхідно виконати, збільшується. Це призводить до більшого споживання ресурсів мережі і, відповідно, до збільшення витрат на Gas.

Деякі операції обходяться дорожче, оскільки вимагають значних обчислювальних потужностей або збільшують обсяг даних, що зберігаються в блокчейні.

3.2 Апроксимація

Розрахувати ліміт Gas'а можна за такою формулою:

$$gasLimit = G_{transaction} + G_{txdatanonzero} * dataByteLength. \quad (5.1)$$

Gtransaction - це стандартна плата за кожну транзакцію, що дорівнює 21000 Gas.

Gtxdatanonzero - плата за кожен ненульовий байт даних або коду для транзакції. Вартість 68 Gas.

dataByteLength - розмір даних у байтах.

Для оцінки ефективності зберігання даних у блокчейні Ethereum необхідно розв'язати задачу визначення оптимального значення вартості Gas, яке дасть змогу доставити повідомлення в задані часові рамки. Завдання полягає в обчисленні такої ціни Gas, за якої час підтвердження транзакції не перевищує задане значення N секунд.

Для цього можна використовувати графік, що показує залежність часу підтвердження транзакції від вартості Gas. Досліджуючи цей графік, можна визначити мінімальну ціну Gas, яка відповідає заданому обмеженню за часом. Цей підхід дає змогу збалансувати витрати на комісію з необхідністю своєчасного опрацювання транзакції, що особливо важливо для ефективного використання ресурсів блокчейна та забезпечення економічної доцільності зберігання даних.

Таблиця 3.1

Ціна Gas'a і час підтвердження

Ціна Gas’a	2	3	4	5	6	7	8	9	10
Час підтвердження (в секундах)	961,8	869,1	473,1	465,1	132,4	8,5	8	2	1,1

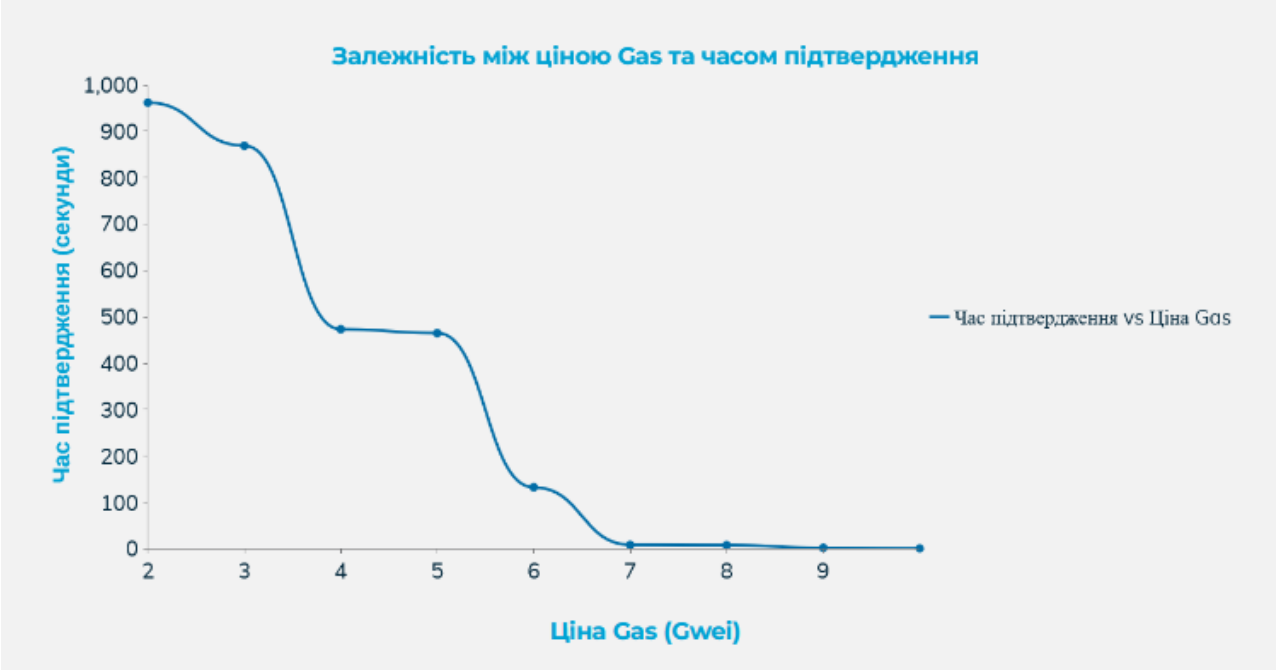


Рис. 3.1 - Час підтвердження за ціною Gas'a

Згідно з графіком, щоб повідомлення доставлялося за 400 секунд, вартість Gas має дорівнювати 5,2. Наша мета - мінімізувати витрати на обмін повідомленнями. Для цього ми застосуємо кубічну регресію, яка допоможе визначити оптимальні витрати.

Використовуємо метод найменших квадратів для знаходження кубічної функції, за допомогою якої будемо апроксимувати залежність часу підтвердження транзакції від вартості Gas.

Рівняння регресії:

$$y = ax^3 + bx^3 + cx + d. \quad (5.2)$$

Коефіцієнти a, b, c і d знайдемо з розв'язку системи:

$$\begin{aligned} a \quad & x_i^3 + b \quad x_i^2 + c \quad x_i + nd = y_i, \\ a \quad & x_i^4 + b \quad x_i^3 + c \quad x_i^2 + d \quad x_i = y_i, \\ a \quad & x_i^5 + b \quad x_i^4 + c \quad x_i^3 + d \quad x_i^2 = x_i^2 y_i, \\ a \quad & x_i^6 + b \quad x_i^5 + c \quad x_i^4 + d \quad x_i^3 = x_i^3 y_i. \end{aligned} \quad (5.3)$$

Опустимо обчислення, шукана функція має вигляд:

$$y = 2{,}3364x^3 - 22{,}021x^2 - 146{,}16x + 1356{,}1. \quad (5.4)$$

Зобразимо на графіку функцію і простежимо за вартістю Gas'a в межах 400 секунд:

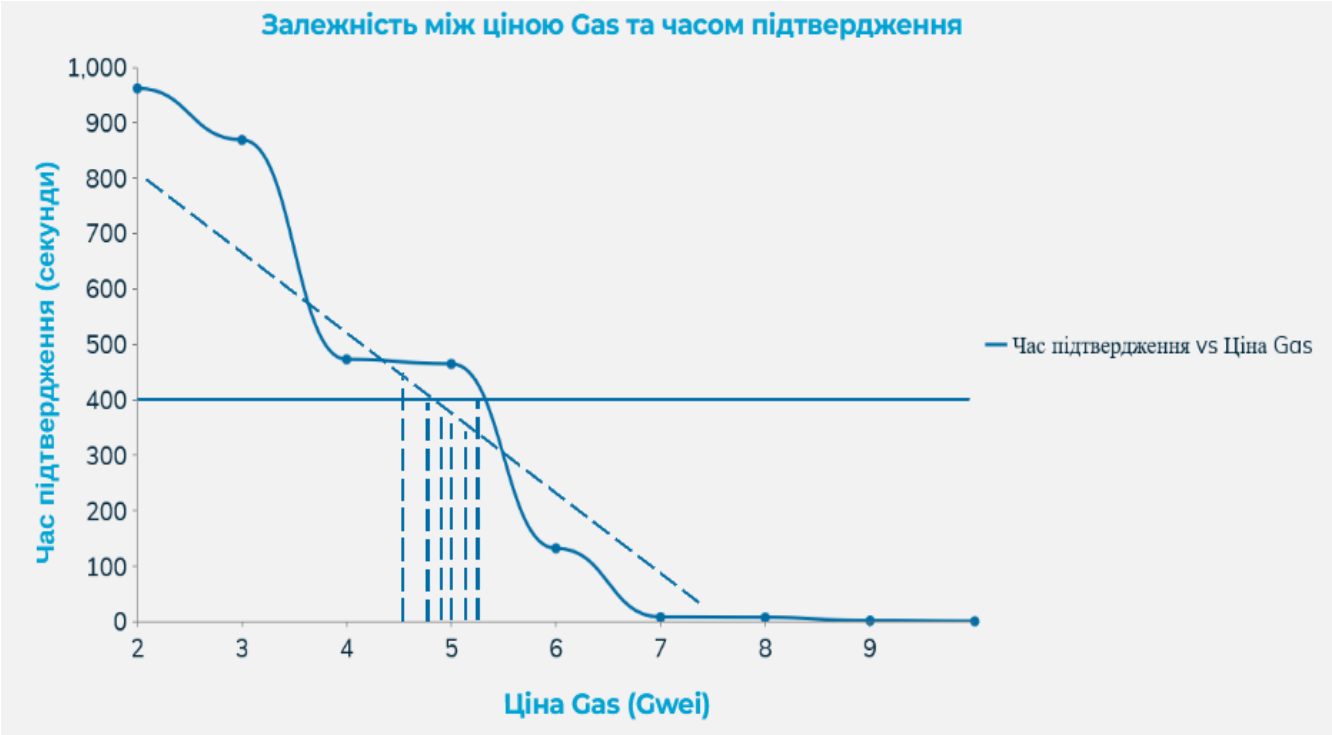


Рис. 3.2 – Апроксимація

З графіка видно, що для надсилання повідомлення можна затратити меншу кількість Gas'a - 5,1 замість 5,2. Отже, спостерігається економія.

Спробуємо ще знизити вартість транзакцій.

Таблиця 3.2

Ціна Gas'a після апроксимації

Ціна Gas’a	5,1	5	4,9	4,6
Час підтвердження (в секундах)	350	360	380	450

Слід зазначити, що за вартості Gas у 4,6 час підтвердження перевищує встановлений ліміт у 400 секунд.

Таким чином, незважаючи на всі переваги запропонованої системи, для надсилання одного й того самого повідомлення в інших системах обміну повідомленнями не буде потрібно додаткових витрат. У нашому випадку, хоча й доведеться заплатити за повідомлення, це гарантує максимальну безпеку передавання та зберігання даних, а доступ до них можна отримати з будь-якого пристрою, використовуючи єдиний обліковий запис.

ВИСНОВОКИ

У рамках виконання магістерської роботи було досліджено різні алгоритми шифрування та їхні теоретичні аспекти. Особливу увагу було приділено криптосистемі RSA, включно з її математичними основами та практичним застосуванням, що супроводжується демонстрацією невеликого прикладу. Цей алгоритм було обрано як основу для реалізації системи безпечного передавання даних між користувачами в мережі.

RSA є одним із найнадійніших алгоритмів шифрування, що ґрунтуються на використанні пари ключів - відкритого і закритого, а також для створення електронного підпису. Ця криптосистема широко використовується в сучасних продуктах і протоколах, які потребують високого рівня безпеки, що робить її ключовим елементом забезпечення захищеного обміну даними в інтернеті.

Крім того, було детально розглянуто платформу Ethereum, призначену для створення децентралізованих застосунків, включно з її смарт-контрактами та принципами роботи. Ця блокчейн-технологія забезпечує високий рівень безпеки під час зберігання даних, оскільки злом секретного ключа Ethereum потребує величезних обчислювальних ресурсів і часу, що вимірюється століттями. Це дає змогу використовувати платформу для розроблення месенджера, що виключає необхідність поділу чатів на секретні та хмарні. Така система гарантує захист листування від втручання третіх осіб або зловмисників, а також надає доступ до повідомлень і файлів з будь-якого пристрою в реальному часі. Це особливо зручно в разі втрати або зміни пристрою, а також у разі використання одного облікового запису на кількох платформах.

Водночас тестування показало, що зберігання даних у блокчейні Ethereum, незважаючи на його надійність і безпеку, є економічно невигідним через високу вартість транзакцій.

ПЕРЕЛІК ПОСИЛАНЬ

1. Why Isn’t Telegram End-to-End Encrypted by Default? URL: https://telegra.ph/Why-Isnt-Telegram-End-to-End-Encrypted-by-Default-08-14 (дата звернення: 25.09.2024).

2. Що таке криптографія? Основні поняття. *ДРУЗІ*. URL: https://druzy.com.ua/sho-take-kriptografiia-osnovni-poniattia/ (дата звернення: 25.09.2024).

3. Криптографія та шифрування. *Школа трейдингу CRYPTOLOGY.KEY | Навчання трейдингу*. URL: https://cryptology.school/blog/kriptografiia-ta-sifruvannia (дата звернення: 25.09.2024).

4. Symmetric Encryption: What, Why, and How | Venafi. *World's #1 Machine Identity Management Platform*. URL: https://venafi.com/blog/what-symmetric-encryption/ (дата звернення: 26.09.2024).

5. Contributors to Wikimedia projects. Data Encryption Standard - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Data_Encryption_Standard (дата звернення: 28.09.2024).

6. Contributors to Wikimedia projects. Advanced Encryption Standard - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard (дата звернення: 29.09.2024).

7. Шифрування: типи і алгоритми. Що це, чим відрізняються і де використовуються? • Hostpro Wiki. *Hostpro Wiki*. URL: https://hostpro.ua/wiki/ua/security/encryption-types-algorithms/ (дата звернення: 01.10.2024).

8. Brush K., Cobb M. What is Asymmetric Cryptography? Definition from SearchSecurity. *Search Security*. URL:

https://www.techtarget.com/searchsecurity/definition/asymmetric-cryptography (дата звернення: 02.10.2024).

9. Contributors to Wikimedia projects. Diffie–Hellman key exchange - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Diffie–Hellman_key_exchange (дата звернення: 03.10.2024).

10. What is RSA Cryptography? Complete Guide to this Encryption Algorithm. *Blockonomi*. URL: https://blockonomi.com/rsa-cryptography/ (дата звернення: 04.10.2024).

11. Simplilearn. RSA Algorithm: Secure Your Data with Public-Key Encryption. *Simplilearn.com*. URL: https://www.simplilearn.com/tutorials/cryptography-tutorial/rsa-algorithm (дата звернення: 05.10.2024).

12. What Is Ethereum?. *CoinDesk*. URL: https://hotfix.coindesk.com/learn/what-is-ethereum (дата звернення: 06.10.2024).

13. Ethereum Whitepaper | ethereum.org. *ethereum.org*. URL: https://ethereum.org/en/whitepaper/ (дата звернення: 07.10.2024).

14. GitHub - ethereum/ethereum-org-website: Ethereum.org is a primary online resource for the Ethereum community. *GitHub*. URL: https://github.com/ethereum/ethereum-org-website (дата звернення: 07.10.2024).

15. Chainani V. ♦ Build Your First Smart Contract using Solidity. *DEV Community*. URL: https://dev.to/envoy_/build-your-first-smart-contract-using-solidity-111p (дата звернення: 08.10.2024).

16. *Dr. Gavin Wood · GitHub Pages*. URL: https://ethereum.github.io/yellowpaper/paper.pdf (дата звернення: 05.11.2024).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних систем та технологій

Магістерська робота

АЛГОРИТМ ЗАБЕЗПЕЧЕННЯ ЦІЛІСНОСТІ ТА АВТЕНТИЧНОСТІ ДАНИХ ПРИ ПЕРЕДАЧІ В МЕРЕЖІ НА ОСНОВІ КРИПТОГРАФІЧНОГО ПІДПИСУ

Виконав: студент групи ІСДМ-61, Іванченко Дмитро Сергійович

Керівник: Данильченко В.М.

Київ - 2024

2

Мета, завдання, об'єкт та предмет дослідження, наукова новизна

Мета: розробка автоматизованої системи для безпечного передавання даних у мережі між користувачами, що використовує криптографічний підпис із парою відкритого та закритого ключів

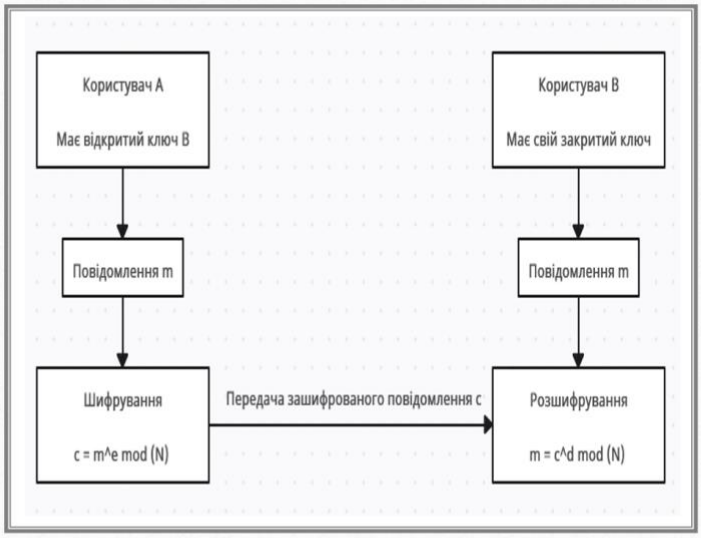
Завдання: аналіз сучасних систем обміну повідомленнями, порівняння алгоритмів шифрування, дослідження можливостей Ethereum і смарт-контрактів, розробка децентралізованої системи на базі Ethereum

Об'єкт дослідження: безпечна передача даних у мережі

Предмет дослідження: алгоритми шифрування даних

Наукова новизна та практична значущість: розроблена децентралізована система передачі даних поєднує криптографічний підпис і можливості Ethereum. Інтеграція смарт-контрактів забезпечує прозорість і надійність. **Практична значущість:** система може використовуватися для захисту даних у фінансах, медицині та управлінні документами

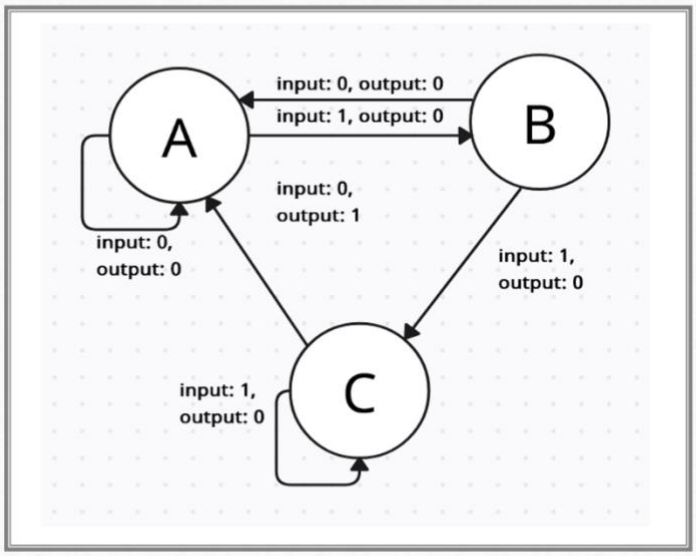
3



РОЗДІЛ 1. АЛГОРИТМ ШИФРУВАННЯ RSA

ПРИНЦИП РОБОТИ RSA

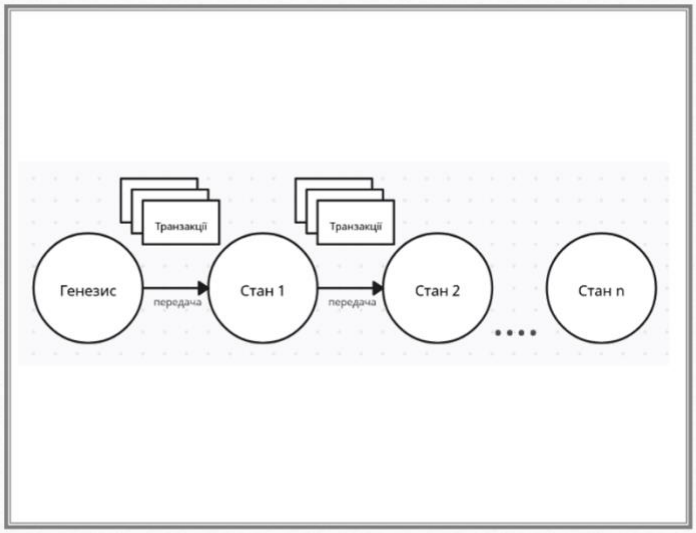
4



РОЗДІЛ 2. РОЗРОБКА БЕЗПЕЧНОЇ СИСТЕМИ ПЕРЕДАЧІ ДАНИХ В ІНТЕРНЕТІ. БЛОКЧЕЙН

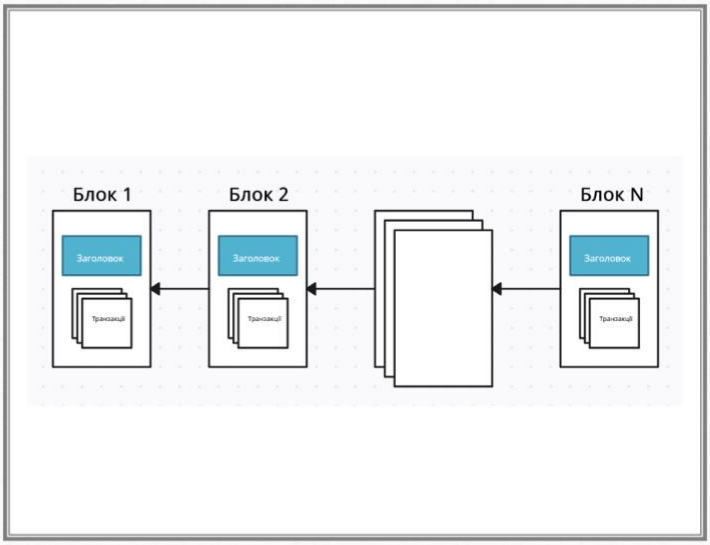
ПАРДИГМА БЛОКЧЕЙНУ ETHEREUM

5



РОЗДІЛ 2. РОЗРОБКА БЕЗПЕЧНОЇ СИСТЕМИ ПЕРЕДАЧІ ДАНИХ В ІНТЕРНЕТІ. БЛОКЧЕЙН

СТАН БЛОКЧЕЙНУ



РОЗДІЛ 2. РОЗРОБКА БЕЗПЕЧНОЇ СИСТЕМИ ПЕРЕДАЧІ ДАНИХ В ІНТЕРНЕТІ. БЛОКЧЕЙН

БЛОКИ ТРАНЗАКЦІЙ

РОЗДІЛ 2. РЕАЛІЗАЦІЯ СИСТЕМИ. СТВОРЕННЯ КЛЮЧІВ ТА ЇХ ВИКОРИСТАННЯ ДЛЯ ТРАНЗАКЦІЙ

```
const EthCrypto = require('eth-crypto');

const identity = EthCrypto.createIdentity();

console.dir(identity);
/* > {
    address: '0xA1B2C3D4E5F67890A123B4C5D6E7F89A0BCDEF12',
    privateKey: '0x9a1b2c3d4e5f67890ab1c2d3e4f56789a0b1c2d3e4f5b67890abcd1234ef5678',
    publicKey: 'f12a34b56c78d90e12f34a56b78c90d12f34a56b78c90d12e34a56b78c90a1bc...'
} */
```

СТВОРЕННЯ НОВОГО ІДЕНТИФІКАТОРА

РОЗДІЛ 2. РЕАЛІЗАЦІЯ СИСТЕМИ. СТВОРЕННЯ КЛЮЧІВ ТА ЇХ ВИКОРИСТАННЯ ДЛЯ ТРАНЗАКЦІЙ

```
const rawTransaction = {
    from: '0xB1C2D3E4F5A67890A123B4C5D6E7F89A0BCDEF12', // адрес відправника
    to: '0x7F4A052378EA5E0149F9E98FAD47A4B8AC5D1211', // адрес отримувача
    value: 500000000000000000, // кількість wei, яку ми хочемо надіслати (=0.5 ефіру)
    nonce: 1, // інкрементне tx-число, додаємо +1 для кожної транзакції
    gasPrice: 10000000000, // ціна газу для транзакції
    gasLimit: 21000 // стандартний gasLimit для звичайних транзакцій
};
```

СТВОРЕННЯ ТРАНЗАКЦІЇ

РОЗДІЛ 2. РЕАЛІЗАЦІЯ СИСТЕМИ. СТВОРЕННЯ КЛЮЧІВ ТА ЇХ ВИКОРИСТАННЯ ДЛЯ ТРАНЗАКЦІЙ

```
const serializedTx = EthCrypto.signTransaction(
  rawTransaction,
  identity.privateKey
);
console.log(serializedTx);
// > '0x8f8462a5b39eb8c4b3543b1d9a0d3dcd6b9a9833b05d6a7f4c2e0c35bbf3c2a9'
```

ПІДПИС З PRIVATEKEY

РОЗДІЛ 2. РЕАЛІЗАЦІЯ СИСТЕМИ. СТВОРЕННЯ КЛЮЧІВ ТА ЇХ ВИКОРИСТАННЯ ДЛЯ ТРАНЗАКЦІЙ

```
const Web3 = require('web3');
const ganache = require('ganache-cli');

// створення екземпляра web3
const web3 = new Web3();

// створення провайдера ganache
const ganacheProvider = ganache.provider({
  // встановлення балансу ідентифікатора на 20 ефірів
  accounts: [{
    secretKey: '0x1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef',
    balance: web3.utils.toWei('20', 'ether')
  }]
});

// встановити ganache на web3 як провайдер
web3.setProvider(ganacheProvider);
```

СТВОРЕННЯ ЛОКАЛЬНОЇ ТЕСТОВОЇ МЕРЕЖІ

РОЗДІЛ 2. РЕАЛІЗАЦІЯ СИСТЕМИ. СТВОРЕННЯ КЛЮЧІВ ТА ЇХ ВИКОРИСТАННЯ ДЛЯ ТРАНЗАКЦІЙ

```
const receipt = await web3.eth.sendSignedTransaction(serializedTx);
const balance = await web3.eth.getBalance('0x7F4A052378EA5E0149F9E98FAD47A4B8AC5S1211');
console.log(balance); // > '500000000000000000'
```

НАДСИЛАННЯ ТРАНЗАКЦІЇ

РОЗДІЛ 2. РЕАЛІЗАЦІЯ СИСТЕМИ. СТВОРЕННЯ КЛЮЧІВ ТА ЇХ ВИКОРИСТАННЯ ДЛЯ ТРАНЗАКЦІЙ

```
const path = require('path');
const SolidityCli = require('solidity-cli');
const contractPath = path.join(__dirname, '../../contracts/DonationBag.sol');
const compiled = await SolidityCli.compileFile(contractPath);
const compiledDonationBag = compiled[':DonationBag'];

const createCode = EthCrypto.txDataByCompiled(
    JSON.parse(compiledDonationBag.interface), // ABI
    compiledDonationBag.bytecode, // байткод
    [creatorIdentity.address] // аргумент конструктора
);

console.dir(compiledDonationBag);
```

КОМПІЛЯЦІЯ В БАЙТКОД

РОЗДІЛ 2. РЕАЛІЗАЦІЯ СИСТЕМИ. СТВОРЕННЯ КЛЮЧІВ ТА ЇХ ВИКОРИСТАННЯ ДЛЯ ТРАНЗАКЦІЙ

```
// відправка в локальний ланцюг
const receipt = await web3.eth.sendSignedTransaction(serializedTx);
const contractAddress = receipt.contractAddress;

console.log(contractAddress);
```

НАДСИЛАННЯ ТРАНЗАКЦІЇ

РОЗДІЛ 2. РЕАЛІЗАЦІЯ СИСТЕМИ. СТВОРЕННЯ КЛЮЧІВ ТА ЇХ ВИКОРИСТАННЯ ДЛЯ ТРАНЗАКЦІЙ

```
const receiverBalance = await web3.eth.getBalance(recieverIdentity.address);
console.dir(receiverBalance);
// '2500000000000000000'
```

ПЕРЕВІРКА БАЛАНСУ

РОЗДІЛ 2. РЕАЛІЗАЦІЯ СИСТЕМИ. СТВОРЕННЯ КЛЮЧІВ ТА ЇХ ВИКОРИСТАННЯ ДЛЯ ТРАНЗАКЦІЙ

```
const signature = EthCrypto.sign(
    A.privateKey,
    EthCrypto.hash.keccak256(secretMessage)
);

const payload = {
    message: secretMessage,
    signature
};

const encrypted = await EthCrypto.encryptWithPublicKey(
    B.publicKey,
    privateKey,
    JSON.stringify(payload)
);

const encryptedString = EthCrypto.cipher.stringify(encrypted);
```

ШИФРУВАННЯ ТА ПІДПИС ПОВІДОМЛЕННЯ

РОЗДІЛ 3. РОЗРАХУНОК ВИТРАТ. АПРОКОМІСІЯ

ЧАС ПІДТВЕРДЖЕННЯ ЗА ЦІНОЮ GAS'A

РОЗДІЛ 3. РОЗРАХУНОК ВИТРАТ. АПРОКОМІСІЯ

АПРОКОМІСІЯ

ВИСНОВКИ

- У рамках виконання магістерської роботи було досліджено різні алгоритми шифрування та їхні теоретичні аспекти. Особливу увагу було приділено криптосистемі RSA. Яка забезпечує надійний захист даних, використовуючи пару ключів (відкритий та закритий).
- Було детально розглянуто платформу Ethereum, призначену для створення децентралізованих застосунків, включно з її смарт-контрактами та принципами роботи.
- Також був зроблений висновок що висока вартість транзакцій робить зберігання даних у блокчейні Ethereum економічно невигідним.