Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Програмної інженерії_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

_____другий (магістерський)_____
(рівень вищої освіти)

Дослідження та оцінка ефективності алгоритмів шифрування, що використовуються в технології Блокчейн

(тема)

Виконав: студент 2 курсу, групи ІПЗм-18-1__
спеціальності _121- Інженерія програмного забезпечення_____
(код і повна назва спеціальності)
освітньо-професійної програми _Інженерія програмного забезпечення_____
(повна назва освітньої програми)
_____Терещенко Г.Ю._____
(прізвище, ініціали)

Керівник _____к.т.н Назаров А.С._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук

Кафедра програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121– Інженерія програмного забезпечення
(код і повна назва)

Освітньо-наукова програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

Студентові Терещенко Глібу Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та оцінка ефективності алгоритмів шифрування, що використовуються в технології Блокчейн

затверджена наказом по університету від «____»_____20___р № ____

2. Термін подання студентом роботи до екзаменаційної комісії «12» травня 2020 р.

3. Вихідні дані до роботи *Алгоритми шифрування, алгоритми захисту даних та пояснювальна записка. Використовувати ОС Windows, середовище об'єктно-орієнтованого проектування.*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз проблемної галузі і постановка задачі, аналіз алгоритмів шифрування в технології блокчейн, опис об'єктних моделей, методи та алгоритми, архітектура програмної системи, опис розробленої програмної системи, результати тестування програмної системи*

5. Консультанти розділів роботи

Найменування розділу	Консультант (посаду, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц. Назаров А.С.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз предметної галузі	25 березня 2020 р.	
2.	Огляд існуючих методів	31 березня 2020 р.	
3.	Методи криптоаналізу	15 квітня 2020 р.	
4.	Підготовка пояснювальної записки	20 квітня 2020 р.	
5.	Спецчастина	28 квітня 2020 р.	
6.	Підготовка презентації та доповіді	03 травня 2020 р.	
7.	Попередній захист	07 травня 2020 р.	
8.	Нормоконтроль, рецензування	08 травня 2020 р.	
9.	Занесення диплома в електронний архів	10 травня 2020 р.	
10.	Допуск до захисту в зав. кафедри	12 травня 2020 р.	

Дата видачі завдання _«_____»_ _____20____р.

Студент _____
(підпис)

Керівник роботи _____ __доц. Назаров А.С.__
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ ABSTRACT

Атестаційна робота магістра містить: 135 с., 20 рис., 7 табл., 3 додататки, 40 джерел.

БЛОКЧЕЙН, АЛГОРИТМИ ШИФРУВАННЯ, RSA, BLOWFISH, КРИПТОСТІЙКІСТЬ, СМАРТ КОНТРАКТИ.

Об’єктом дослідження є оцінка ефективності алгоритмів шифрування, які можуть бути використані у технології блокчейн.

Метою роботи є вирішення науково-практичної задачі аналізу та оцінки ефективності алгоритмів шифрування при створенні смарт контрактів, що застосовуються у технології блокчейн, для попереджування неочікуваних наслідків від наявних вразливих місць та всіляких загроз.

Методи розробки базуються на методах математичного моделювання.

У результаті роботи здійснено вибір алгоритмів шифрування та програмна реалізація системи.

BLOCKCHAIN, ENCRYPTION ALGORITHMS, RSA, BLOWFISH, CRYPTIC RESISTANCE, SMART CONTRACTS.

In terms of efficiency, the algorithm is more widely used, which can be used in blockchain technology.

The method of work consists in the use of scientific and practical tasks and requires the efficiency of the encryption algorithm in creating smart counterparties used in the technological unit, to preview uncertain cases of existing disaggregated places and all threats.

Mailing methods are based on mathematical modeling methods.

As a result of the work the choice of encryption algorithms and the program of realization of systems was offered.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМІНІВ

Блок – Це Частина Ланцюжку Блоків, Що Містить Та Підтверджує Багато Транзакцій, Що Очікують підтвердження

ЕЦП – Електронний Цифровий Підпис, Реквізит Електронного Документа, Призначений Для Захисту Даного Електронного Документа Від Підробки

ЗКЗІ – Засоби Криптографічного Захисту Інформації

Підпис – Математичний Механізм, Що Дозволяє Підтвердити Право Власності

Приватний ключ – Секретна Послідовність Певних Даних, Що Дає Вам Право Витрачати Цифрову Валюту З Конкретного Гаманця За Допомогою Криптографічного Підпису

Blockchain – Ланцюжок З Формованих Блоків Транзакцій, Який Побудований За Певними Правилами

P2P – Термін Peer-To-Peer Означає Системи, Що Працюють Як Організована Спільнота, Надаючи Можливість Кожному Учаснику Безпосередньо Взаємодіяти З Іншими

3DES – Стандарт Потрійного Шифрування Даних

AES – Розширений Стандарт Шифрування

CBC – Режим Ланцюга Блоків Ланцюга

CTR – Режим Лічильника

CFB – Режим Зворотного Зв'язку

ECB – Режим Електронної Кодової Книги

PKC – Криптографія Відкритого Ключа

ВСТУП

Технологія блокчейн (Blockchain) з'явилася недавно, але вже набрала популярність і має потенційно велику значимість для проведення різних операцій завдяки своїй захищеності і надійності.

Технологія зберігання інформації блокчейн стала досить актуальна в останні роки. Неможливо заперечувати її значимість в суспільстві, в якому збільшується потреба у підтвердженні правдивості і охорони збереженої інформації. Численні фахівці переконані в тому, що ця технологія здатна застосовуватися в багатьох галузях. Технології, засновані на застосуванні блокчейна можуть зробити переворот в концепції урядового управління, економічних послуг та індустрії. Надаються величезні можливості, проте основне питання полягає в їх реалізації.

Блокчейн – це розподілена база даних, в якій можна зберігати будь-які дані або транзакції. У блокчейні зберігається інформація усієї мережі, що складається з персональних комп'ютерів, і в результаті виходить не тільки децентралізований, а й розподілений простір. Отже, ні компанія, ні людина, ні будь-яка інша довірена сторона не є власником цієї мережі. Всі люди можуть користуватися системою і тим самим підтримувати її функціонування, тому одній людині дуже складно зламати або повністю знищити мережу. Користувачі мережі використовують свої персональні комп'ютери для зберігання пучків даних інших користувачів, які називаються блоками («blocks») в хронологічній ланцюга («chain»), звідси і назва «блокчейн» дослівно «ланцюг з блоків». Блокчейн-революцію прийнято розділяти на 3 категорії: Блокчейн 1.0 (криптовалюта біткойн), 2.0 (смарт контракти) і 3.0 (алгоритм шифрування). Блокчейн має можливість розв'язувати питання безпеки, високої доступності та швидкості виконання транзакцій [33].

Актуальність даної роботи зв'язана з тим, що методи та технології віддалених мережевих атак регулярно удосконалюються, і існуючі алгоритми і системи шифрування не завжди повністю захищають конфіденційну інформацію. Але

розвиток технологій блокчейна, які мають високу криптографічну стійкість, так само не стоять на місці. Ці обставини роблять розробку і впровадження системи безпечної передачі даних за допомогою криптографічного підпису з відкритим і закритим ключами дуже актуальними.

Сучасність і наукова новизна даної роботи полягає в тому, що в поточний момент технологія Блокчейн є порівняно новою і більшість наукових досліджень випущено нещодавно (2014-2019 рр.), отже модель безпечного децентралізованого сховища, заснованого на даній технології практично не вивчена [36].

Проблемам вивчення ефективності алгоритмів шифрування присвячено чимало досліджень. Серед останніх робіт у цій галузі слід відмітити праці А. Єлізарова [5], дослідження Ю. Мінгальова [15], С. Кос [34], М. Лапіна [12], О. Г. Качко [6] та інших. Однак всі перелічені роботи розглядають алгоритми шифрування з огляду на криптостійкість та швидкодію у межах конкретних замкнених систем кожного з алгоритмів, а не у межах технології блокчейну, або смарт контрактах. Тому вважаємо актуальним питання дослідження та оцінки ефективності алгоритмів шифрування, що використовуються в технології блокчейн.

Метою роботи є вирішення науково-практичної задачі аналізу та оцінки ефективності алгоритмів шифрування при створенні смарт контрактів, що застосовуються у технології блокчейн, для попереджування неочікуваних наслідків від наявних вразливих місць та всіляких загроз.

Для досягення поставленної мети необхідно вирішити наступні завдання:

– проаналізувати і порівняти існуючі алгоритми і системи шифрування даних з точки зору предмета дослідження;

– дослідити смарт-контракти, що застосовуються у технології блокчейн;

– розробити власний блокчейн;

– визначити як працює блокчейн з алгоритмами шифрування та запропонувати ефективніший алгоритм, який дозволить підвищити надійність і рівень безпеки для блокчейну.

Об’єктом дослідження є оцінка ефективності алгоритмів шифрування, що використовуються в технології блокчейн.

Предметом дослідження є алгоритми шифрування даних, які можуть бути використані у смарт контрактах.

Методи досліджень. Теоретичною базою виконаних досліджень є фундаментальні положення теорії алгоритмів та теорії статистичного аналізу, методи математичного моделювання, порівняльний аналіз предмету досліджень, що застосовються і можуть бути запропоновані для майбутнього використання.

Апробація результатів дослідження. На тему дослідження було опубліковано вісім статей та тез доповідей [16, 17, 23, 24, 27, 29, 38, 39], зміст яких було викладено у періодичних фахових виданнях та на міжнародних науково-практичних конференціях, отримано свідоцтво про реєстрацію авторьского права на твір [14, 25]:

– міжнародна Науково-технічна конференція «Інформаційні системи та технології» (ІСТ – 2017) (Коблево-Харків, 11-16 вересня, 2017 р.);

– konferencji medzynarodowej Naukowo-praktycznej «Rzwnój i praktyka. Inżynieria i technologia» (Zakopane (PL), 29.12.2017 р.);

– 14-th International Scientific Conference «Intellectual Systems for Decision Making and Problems of Computational Intelligence» (ISDMCI’2018’) (Kherson, 2018 р.);

– ІІ міжнародна Науково-практична конференція «Теорія і практика актуальних наукових досліджень» (м. Одеса, 28-29 квітня 2018 р.);

– 7-ма міжнародна науково-технічна конференція «Інформаційні системи та технології» (ІСТ – 2018) (Коблево - Харків, 2018 р.);

– міжнародна Науково-практична конференція «Інтелектуальні системи та інформаційні технологі» (ISIT-2019) (Одеса, 19 − 24 серпня 2019 р.);

– 23-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті» (Харків, 2019 р.)

АНАЛІЗ СТАНУ РОЗВ'ЯЗАННЯ ПРОБЛЕМИ ТА ОБҐРУНТУВАННЯ ЦІЛЕЙ ДОСЛІДЖЕННЯ

1.1 Опис проблемної галузі

Смарт-контракти – головний рушій технології блокчейн, адже тільки з системою, як блокчейн, в якій не можна підробити, змінити та видалити дані, смарт контракти можуть прирівнюватися до офіційних документів. Наразі, у блокчейні використовується лише один алгоритм шифрування у смарт контрактах, бо вважається, що для його взлому необхідно більше декількох сот років на сучасному етапі. Але, ніхто не змушую нас використовувати лише один алгоритм, що як використовувати різні алгоритми, або суміш алгоритмів шифрування, яка зможуть дати навіть більшу криптостійкість та швидкість виконання алгоритму.

У ході роботи буде створено власний блокчейн та створений імітація онлайн гаманця, в якому усі транзакції будуть відбуватися у блокчейні. Це зроблено для того, щоб у момент транзакції створити смарт контракт та зашифрувати його з допомогою різних алгоритмів шифрування та зрозуміти, який є найбільш швидким, надійним та вигідним для використання.

Можна бути впевненими, що блокчейн технологія має величенний потенціал. Ринки прогнозів та платформи для керування активами, децентралізовані біржі – оце все лише частка з зустрічающихся привабливих додатків, які вивчаються розробниками блокчейну. Але всупереч загальновідомим перевагам, блокчейн має декілька визначальних технічних бар’єрів. Це робить його непрактичним для застосування. Перелік головних з них:

- обмежена масштабованість;
- обмежена конфіденційність;
- відсутності у формальній перевіці смарт-контрактів;
- обмеження по зберіганню даних;
- нестійкість механізмів консенсусу;

– незадовільнє оснащення;

– відсутність засобів управління та стандартів;

– квантова обчислювальна небезпека, тощо.

У цьому розділі буде розглянута проблема приватності в блокчейні більш детально. Це дасть змогу надалі сформувати спиок випадків, коли через недоліки в реалізації відносно захисту даних клієнтів мережі дана технологія не може бути використана.

Беручи до уваги, що в блокчейні транзакції прямо не прив’язані до користувачів, вони, з першого погляду, можуть здаися достатньо приватними. Будь-хто таємно може створити новий електронний гаманець та здійснювати транзакції з його допомогою. Однак доцільно присвітити даному процесу та його реалізації дещо більше уваги, щоб мати змогу ширше розуміння про нього. З однієї сторони, безумовно правильно, що великй плюс блокчейн технології – анонімність. Тут транзакції записуються і зберігаються в публічній книжці обліку, а окрім того вони зв’язані з адресою облікового запису, який формується виключно з цифр і букв. За умови, що до цієї адреси не додана жодна ідентичність, то визначення відправника транзакції здається неможливим.

Однак подібна дефініція повної безпеки може бути досить оманлива. Користувач, насправді, зберігає свою приватність доки його псевдонім не зв’язан з його ідентичністю, але як тільки хтонебудь викаже такий зв’язок – то таємниця викривається. Одна ілюстрація такого явища була оприлюднена, коли правоохоронні органи визначили, що під час одного з розслідувань вони змогли ідентифікувати реальних користувачів біткойну. Таким чином вони були «де-анонімізовани», а загальна передумова транзакційної невидимості була порушена. Веб-трекери та «cookie» – це файли на веб-сайтах припускають вихід інформації про транзакцію в мережі інтернет, де абихто, у тому числі уряд та правоохоронні органи, а також і зловмисників, з готовністю користуються цими даними [40].

1.2 Наявні дослідження в області технології блокчейн

Криптостійкі алгоритми, прийняті в якості національних або світових стандартів, є загальнодоступними. Їх криптостойкость базується на нерозв'язних за прийнятний час математичних задачах. Але реалізація криптоалгоритмів з урахуванням високої швидкодії, відсутності помилок і гарантованого виконання вимог математичних перетворень – непросте завдання, яким займаються кваліфіковані розробники [22].

В разі, якщо електронний підпис використовується в критичних додатках (наприклад, для виконання юридично значимих дій), реалізація криптоалгоритмів в обов'язковому порядку проходить процес сертифікації на відповідність вимогам безопасності.Дополнітельно, засоби криптографічного захисту інформації (ЗКЗІ) можуть мати саме різне уявлення: від програмних бібліотек до високопродуктивних спеціалізованих залозок (Hardware Security Module, HSM).

Саме через складність реалізації і регулювання даного виду продукції існує ринок рішень з криптографічного захисту інформації, на якому грають різні гравці. З метою сумісності різних реалізацій, а також спрощення їх вбудовування в прикладне програмне забезпечення, були розроблені кілька стандартів, які стосуються різних аспектів роботи з ЗКЗІ і безпосередньо електронним підписом [2].

PKCS#11 (Public-Key Cryptography Standard#11) – програмний інтерфейс для роботи з апаратно реалізованими ЗКЗІ: смарт-карти, HSM'и, криптографічні токени. Іноді PKCS#11 використовується для доступу до програмно реалізованим криптографічним бібліотекам.

PKCS#11 являє собою досить великий документ, опублікований RSA Laboratories, який описує набір функцій, механізмів, алгоритмів та їх параметрів для роботи з криптографічними пристроями або бібліотеками. В даному документі

чітко виписані правила, відповідно до яких буде працювати прикладне ПЗ при виклику криптографічних функцій.

Цей стандарт підтримується в багатьох open source-проектах, що використовують криптографію. Для прикладу, Mozilla Firefox дозволяє зберігати сертифікати і закриті ключі для аутентифікації через SSL/TLS на токенах, працюючи з ними по PKCS#11.

Якщо прикладне програмне забезпечення «вміє» працювати з PKCS#11, то виробнику ЗКЗІ необхідно реалізувати програмну бібліотеку, яка назовні буде виставляти інтерфейси, описані в стандарті, а всередині реалізовувати необхідну ЗКЗІ логіку: робота безпосередньо з криптографічним пристроєм або реалізація необхідних криптоалгоритмів програмно. В цьому випадку прикладне програмне забезпечення повинне «підчепити» необхідну бібліотеку і виконувати необхідні дії відповідно до рекомендацій стандарту. Це забезпечує замінюваність різних пристроїв і їх бібліотек для прикладного програмного забезпечення та прозоре використання ЗКЗІ в різних системах.

Єдиним істотним мінусом PKCS#11 є відсутність рекомендацій по роботі з сертифікатами відкритого ключа, що передбачає або використання пропрієтарних розширень стандарту, або використання інших засобів для роботи з сертифікатами.

Операційна система Microsoft Windows, незалежно від версії, досить складна система, яка крім усього іншого займається питаннями забезпечення безпеки користувача і корпоративних даних. У цих завданнях традиційно широко використовується криптографія.

Компанія Microsoft для уніфікації доступу до криптографічних функцій розробила пропріетарний API: Microsoft Crypto API. Широке поширення набула версія Crypto API 2.0. Парадигма Microsoft Crypto API базується на використанні так званих криптопровайдерів, або, по-українські, постачальників криптографії.

Специфікація Crypto API описує набір функцій, що повинні бути надані бібліотекою криптопровайдера операційній системі, способи інтеграції з нею і специфікації викликів. Таким чином, виробник ЗКЗІ, що виконує правила Crypto

API, має можливість інтеграції свого рішення в операційну систему Microsoft Windows, а прикладне ПЗ отримує доступ криптографічним функцій за допомогою уніфікованого інтерфейсу [8]. Додатковим плюсом є те, що компанія Microsoft реалізувала над Crypto API досить велика кількість функцій, що відповідають за виконання прикладних завдань: робота з сертифікатами, формування різних видів підпису, робота з ключовою інформацією та ін. Також Crypto API, як неважко здогадатися, тісно інтегрований з операційною системою і її внутрішніми механізмами.

Але розплатою за зручність стає платформозалежність і необхідність тісної інтеграції рішення в операційну систему.

У Windows Vista Microsoft представила нову версію криптографічного програмного інтерфейсу – Microsoft CNG (Cryptography API: Next Generation). Даний інтерфейс базується вже не на постачальниках криптографії, а на постачальниках алгоритмів і постачальників сховищ ключової інформації. В силу того, що поширеність Windows XP все ще досить велика, CNG в прикладних системах використовується вкрай мало.

Наявність стандартизованих інтерфейсів ніколи не було перешкодою для існування пропрієтарних бібліотек зі своїми інтерфейсами. Прикладом цього може служити бібліотека OpenSSL, робота з якою здійснюється за власними правилами. Стандарт PKCS#11 передбачає можливість використання пропрієтарних розширень. Фактично, це додані виробником функції, не описані в стандарті. Зазвичай вони служать для виконання специфічних операцій з пристроями або реалізують необхідні, на думку виробника, функції.

Описані вище інтерфейси для доступу до криптографічних функцій дозволяють звертатися за виконанням математичних перетворень до різних, як добре було помічено Microsoft, «постачальникам криптографії».

Але алгоритмів вироблення і перевірки електронного підпису існує безліч. У кожного з цих алгоритмів є набір параметрів, які повинні бути узгоджені при виробленні та перевірці. Плюс для перевірки підпису по-хорошому потрібен

сертифікат. Всі ці параметри необхідно або узгодити в прикладної системі, або передавати разом з підписом [3].

PKCS#7 (Public-Key Cryptography Standard#7), або CMS (Cryptographic Message Syntax) – стандарт, що публікується і підтримуваний все тієї ж RSA Laboratories, описуваний синтаксис криптографічних повідомлень, також публікується в якості RFC з номером 2315.

Синтаксис CMS описує способи формування криптографічних повідомлень, в результаті чого повідомлення стає повністю самодостатнім для його відкриття і виконання всіх необхідних операцій.

З цією метою в PKCS#7 розміщується інформація про вихідний повідомленні (опціонально), алгоритмах хешування і підписи, параметрах криптоалгоритмів, часу підпису, сертифікат ключа електронного підпису, ланцюжок сертифікації і т.ін.

Більшість атрибутів PKCS#7 є опціональними, але їх обов'язковість може визначатися прикладної системою.

Окремо слід відзначити, що PKCS#7 дозволяє ставити кілька підписів під одним документом, зберігаючи всю необхідну інформацію в повідомленні.

Формування та перевірка PKCS#7 реалізовані в криптографічних «надбудовах» в Microsoft Crypto API, мова про які йшла вище. Але сам формат досить добре спеціалізований і його підтримка може бути реалізована нативно.

W3C розробила і опублікувала рекомендації щодо складання підписаних повідомлень в форматі XML.

Фактично XML-DSig вирішує ті ж питання, що і PKCS#7. Основна відмінність в тому, що в PKCS#7 дані зберігаються в структурах, сформованих відповідно до розмітки ANS.1 (фактично, бінарні дані), а в XML-DSig дані зберігаються в текстовому форматі відповідно до правил документа «XML Signature Syntax and Processing».

Описані вище формати електронного підпису необхідні при взаємодії різнорідних систем, коли інформація про відправника підписаного повідомлення мінімальна.

При взаємодії в рамках однієї інформаційної системи, коли інформація про відправника, використовуваних криптоалгоритмах і їх параметрах відома одержувачу, більш раціонально використання «сирого» електронного підпису.

У цьому випадку фактично передається тільки саме значення електронного підпису разом з документом. Інформація для перевірки береться одержувачем з централізованої бази даних [37].

Це дозволяє значно спростити процедури вироблення і перевірки підпису, так як відсутні кроки формування та розбору повідомлень відповідно до будь-якого формату.

1.3 Постановка задачі оцінки алгоритмів шифрування

При проектуванні шифрів першорядним є питання забезпечення їх стійкості. Ця проблема одночасно є і найбільш складною. Оцінка стійкості є одним з найбільш тривалих, трудомістких і різнопланових етапів. При цьому важливим є, творчий підхід, оскільки в даний час немає закінченої теорії блокових шифрів, яка б дозволила виробити повноцінну методику оцінювання стійкості. Однак вже запропоновані деякі загальні вимоги до якості шифруючих перетворень. Якщо шифр задовольняє таким вимогам, то говорять про доказовість стійкості (до відомих методів криптоаналізу) або про досягнення гарантованих властивостей шифруючих перетворень.

Етап.1. Вивчення галузі використання, тобто здійснюється вибір типу криптосистеми і формулювання вимог до її основними параметрами. Блокові шифри в даний час знайшли більш широке застосування. Вони забезпечують

високу стійкість в режимі незалежного шифрування окремих блоків, дозволяючи здійснювати довільний доступ до зашифрованих масивів даних.

Етап.2. Вибір довжини секретного ключа. В даний час для універсального випадку рекомендується довжина ключа не менше 128 біт, але в окремих випадках може застосовуватися довжина ключа 64 і навіть 56 біт, якщо інформація не представляє значної цінності.

Етап.3. Вибір розкладу використання ключа. Порядок використання криптографічної системи включає системи установки і управління ключами. Система установки ключів визначає алгоритми і процедури генерації, розподілу передачі і перевірки ключів. Система управління ключами – визначає порядок використання, зміни, зберігання та архівування, резервного копіювання та відновлення, заміни або вилучення з обігу скомпрометованих, а також знищення старих ключів.

Етап.4. Вибір базових криптографічних примітивів і розробка кріптосхеми, тобто необхідні знання про основні підходи до побудови шифрів, типах блокових криптосистем, а також про конкретні шифри і властивості використовуваних операцій, вартості їх апаратної реалізації і внесення часу затримки.

Етап.5. Оцінювання споживаних ресурсів для реалізації алгоритму шифрування. Розглядаються варіанти програмної і апаратної реалізації. Здійснюється реалізація експериментальних шифраторів і їх програмних моделей.

Етап.6. Оцінювання продуктивності шифру. Визначається значення швидкості шифрування для різних варіантів програмної та апаратної реалізації. Якщо оцінки, отримані на кроках 5 та 6, не задовольняють значенням визначеним на кроці 1, то повторюється етап 4 з урахуванням результатів, отриманих на поточному етапі.

Етап.7. Розгляд стійкості до можливих типів криптоаналітичних атак. Розглядаються основні варіанти криптоанализа, а також інші можливі варіанти атак з використанням особливостей застосування (інженерний аналіз).

Етап.8. Зміна алгоритму з урахуванням попереднього аналізу. З урахуванням результатів, отриманих на етапі 7, здійснюється оптимізація основних вузлів кріптосхеми з метою підвищення трудомісткості найбільш ефективної атаки. Після цього виконується попередня оцінка стійкості модифікованого алгоритму. Якщо необхідно, повторити цей пункт кілька разів до отримання прийнятного попереднього значення стійкості.

Етап.9. Виконання детального аналізу модифікованого шифру. Якщо детальний аналіз виявив суттєві слабкості в модифікованій схемі, то потрібно повернутися до етапу 8 або навіть до етапу 4.

Етап.10. Виконання статистичних тестів і спеціальних експериментів. На цьому кроці здійснюється програмування алгоритму шифрування або його реалізація в програмованих логічних матрицях – програмованих логічних інтегральних схемах (ПЛІС) і проводяться стандартні статистичні тести і спеціальні експерименти, плановані з урахуванням результатів аналізу для перевірки повноти та адекватності теоретичного аналізу.

Важлива проблема створення поліпшеного стандарту симетричного шифрування – криптографічного алгоритму XXI століття була ініційована Національним Інститутом Стандартів і Технологій (NIST) США. Для цього в 1997 році NIST оголосив конкурс алгоритмів, які претендували на те, щоб стати стандартом, а також сформував мінімальні вимоги, які повинні бути виконані при розробці та поданні на конкурс таких алгоритмів. Стандарт симетричного шифрування AES (Advanced Encryption Standard) – це загальнодоступний, розповсюджуваний без обмежень по всьому світу симетричний криптоалгоритм, який зможе здійснювати надійний захист урядової інформації [1].

Було сплановано три етапи розгляду поданих алгоритмів. На першій конференції, присвяченій стандарту AES, в 1998 р винесено рішення про прийняття до відкритого обговорення 15 пакетів описів алгоритмів симетричного шифрування. На другому етапі було здійснено публічний аналіз відповідності кожного стандарту вимогам і оцінка якості криптографічних перетворень в цілому.

У серпні 1999 р проведена друга конференція, метою якої було узагальнення оцінок, зменшення кількості алгоритмів-кандидатів до 5, які потім розглядалися на третій конференції в квітні 2000 року. Розробник представляв свої оцінки щодо обчислювальної складності програмного і апаратного забезпечення, а також обчислювальну ефективність алгоритму, в тому числі і для 8-бітових процесорів. Для оцінки апаратного забезпечення рекомендується використовувати в якості показника число логічних елементів, для програмного забезпечення - тип процесора, тактову частоту його роботи, обсяг пам'яті, операційну система, і т. ін.

Оцінка швидкості роботи алгоритму представлялася у вигляді числа тактів роботи, необхідної для шифрування одного блоку даних, дешифрування одного блоку даних, розгортання (настройки) ключа, настройки алгоритму або його частини, зміни ключа для кожної з робочих довжин.

При створенні перспективного стандарту шифрування визначені і зафіксовані вимоги до нього. Мінімальні вимоги, яким повинні були відповідати на першому етапі алгоритми-кандидати AES, такі:

– криптоалгоритм повинен будуватися на використанні криптографії симетричних (секретних) ключів, тобто ключі джерела криптограми і одержувача повинні або збігатися або розраховуватися один з іншого не вище ніж з поліноміальною складністю;

– криптоалгоритм повинен будуватися з використанням блочного симетричного шифру з довжиною блоку lб = 128 біт;

– довжини початкових ключів повинні бути lk = 128, 192 і 256 біт;

– криптоалгоритм має працювати для різних комбінацій довжина повідомлення/довжина ключа, але обов'язково для комбінацій довжина блоку/довжина ключа 128/128, 128/192, 128/256 біт.

Відбір AES алгоритмів здійснювався з урахуванням виконання мінімальних вимог. Крім того, враховувалося нижче перераховане.

Реальна захищеність від криптоаналітичних атак. Основні методи криптоаналізу при цьому є: диференційний криптоаналіз, розширення для

диференціального криптоаналізу, пошук найкращої диференціальної характеристики, лінійний криптоаналіз, інтерполяційне вторгнення, вторгнення з частковим вгадування ключа, вторгнення з використанням пов'язаного ключа, вторгнення на основі обробки збоїв, пошук лазівок [19].

Розрахункова складність криптографічних алгоритмів для програмної і апаратної реалізацій (може оцінюватися швидкістю перетворень).

Вимоги до пам'яті при програмній і апаратній реалізаціях. При апаратній реалізації оцінюється числом логічних елементів, при програмній – кількістю потрібної оперативної і постійної пам'яті, в тому числі для всіляких платформ і середовищ.

Гнучкість алгоритму, тобто: змога роботи з іншими довжинами початкових ключів та інформаційних блоків безпеку реалізації в широкому діапазоні різноманітних платформ і додатків, включаючи 8-бітові процесори, можливість застосування криптографічного алгоритму в якості поточного шифру або генератора псевдовипадкових чисел, алгоритму хешування, для забезпечення справжності повідомлень (вироблення кодів аутентифікації) і т.ін., однакової складності як апаратної, так і програмної реалізації, а також програмно-апаратної реалізації.

Аналіз основних публікацій щодо дослідження властивостей алгоритмів-кандидатів AES і отримані авторами результати дозволяють порівняти представлені алгоритми відповідно до критеріїв і показників оцінки якості.

Існують основні методи криптоаналізу, які повинні застосовуватися до алгоритмів шифрування. Це досить складна проблема. Необхідно розглядати і оцінювати стійкість алгоритмів до атак звичайного і розширеного диференціального криптоаналізу, пошуку найкращої диференціальної характеристики, лінійного криптоаналізу, интерполяційного і итераційного вторгнення, вторгнення з частковим вгадуванням ключа і з використанням пов'язаного ключа, вторгнення на основі обробки збоїв, пошуку лазівок і т.ін. [19].

2 ОПИС ПРОВЕДЕНИХ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

2.1 Опис алгоритмів шифрування в блокчейн технології

Одна з перших публікацій алгоритму Діффі-Хелмана появилася в сімдесятих роках прошлого століття у статті Уитфилда Діффі і Мартіна Хелмана, де були введені основні категорії криптографії з відкритим ключем. Алгоритм Діффі-Хелмана не використовується для шифрування даних або формування ЕП. Його місія – в розподілі ключів. Алгоритм дає змогу двом або більше користувачам обмінюватися ключем без посередників, який може використовуватися для симетричного шифрування. Це стало першою криптосистемою, що дозволила захищати інформацію без необхідності використовувати секретні ключі, що мали передаватися по захищених каналах. Запропонована Діффі і Хеллманом схема відкритого розподілу ключів, здійснила справжню революцію в шифруванні, тому що вирішила основне питання класичної криптографії – проблема розподілу ключів [9].

RSA – криптографічний алгоритм з відкритим ключем, який базується на обчислювальної складності задачі факторизації великих цілих чисел. RSA є першим алгоритмом шифрування з відкритим ключем. Від перших букв прізвищ авторів цієї системи складається і її назва – Рональд Ривест, Аді Шамір і Леонард Адлеман – три вчених з технологічного інституту Массачусетса.

Після публікації статі Уитфилда Діффі і Мартіна Хеллмана «Нові напрямки в криптографії» в 1976 році та її вивчення (що заклало основи криптографії з відкритим ключем), Ривест, Шамір і Адлеман почали пошук математичної функції, яка б дозволила реалізувати модель системи, що описана в статті. Вченим вдалося винайти алгоритм, який ґрунтувався на тому, наскільки просто знайти великі прості числа і наскільки складно розкласти на множники твір двох великих простих чисел, після роботи над більше як сорока можливих варіантах.

Для шифрування використовується проста операція піднесення до степеня за модулем N. Для розшифрування необхідно обчислити функцію Ейлера від числа N, для цього необхідно знати розкладання числа N на прості множники (в цьому полягає завдання факторизації). У криптографічного системі RSA відкритий і закритий ключі складаються з пари цілих чисел [9]. Закритий ключ зберігається в секреті, а відкритий повідомляється іншому учаснику, або десь публікується [34].

Blowfish являє собою 64-бітовий блочний шифр із ключем зі змінною довжиною. Алгоритм створюється з двох частин – розгортання ключа і шифрування даних. Розгортання ключа трансформує ключ довжиною до 448 бітів в декілька масивів подключей загальний обсяг якого 4168 байтів.

Шифрування даних – це проста функція, яка виконується послідовно 16 раз. Кожен етап утворюється із залежним від ключа переставлянням і з залежним від ключа і даних підставлянням. Використовуються виключно складання і XOR 32-бітових слів. Єдині додаткові операції на кожному етапі – це чотири вилучення даних з індексованого масиву.

В алгоритмі Blowfish використовується чималоо підключей, які мають бути розрахованими до початка шифрування або дешифрування даних.

Advanced Encryption Standard – симетричний алгоритм блочного шифрування, прийнятий урядом США як стандарт в результаті змагання, проведеного між технологічними інститутами. Він замінив застарілий Data Encryption Standard, який більше не відповідав вимогам мережевої безпеки, що ускладнилися в XXI столітті. Поточний алгоритм, окрім абревіатури AES, ще інколи називають Rijndael - це анаграма з часток імен програмістів з Бельгії – Vinent Rijmen і Joan Daemen, які алгоритм AES розробили. Відверто кажучи, AES і Rijndael – не зовсім одне й те саме, оскільки AES має фіксований розмір блоку в 128 біт і розміри ключів в 128, 192 і 256 біт, в той час як для Rijndael можуть бути задані будь-які розміри блоку і ключа, від мінімуму в 32 біт до максимуму в 256 біт. Алгоритм AES був схвалений Агентством національної безпеки США як придатний для шифрування особливо секретної інформації. Однак, уряд

постановив, що AES повинен періодично піддаватися перевіркам і поліпшень, щоб надійно зберігати зашифровані дані. Інформація, визначена як секретна, повинна бути захищена за допомогою AES з довжиною ключів 128, 192 і 256 біт. Для інформації, визначеної як особливо секретна, ця довжина становить 192 або 256 біт. Суть AES в тому, що будь-яка «лобова атака» на захищені дані – тобто підбір усіх можливих паролей – в перспективі дуже сильно розтягується. Якщо уявити, що зломщик має в своєму розпорядженні величезними ресурсами, тобто цілою колекцією суперкомп'ютерів, то при ретельному старанні доступ до зашифрованих даних він міг би отримати через десятки років. Якщо ж в його розпорядженні нічого цього немає, то злом AES займе астрономічно довгий час.

2.2 Математичний опис алгоритмів

Алгоритм Діффі-Хелмана створений на труднощі обрахунку дискретних логарифмів. В ньому, як і в багатьох других алгоритмах з відкритим ключем, обрахунок виконуються по модулю якогось великого простого числа *P*. Спершу спеціальним чином добирається деяке натуральне число *A*, яке менше *P*. В разі потреби зашифрувати значення *X*, то разраховується

$$Y = A^x mod\ P. \tag{2.1}$$

При цьому, якщо є *X,* обрахувати *Y* не складно. Протилежне завдання вирахування *X* з *Y* є досить складним. Експонента *X* і звється дискретним логарифмом *Y*. Отже, знаючи про складність розрахунку дискретного логарифма, число *Y* можна незахищенно передавати навіть по каналу зв'язку що відкритий, тому що початкове значення *X* при великому модулі *P* буде майже неможливо

підібрати. На цьому математичному чиннику і започаткован алгоритм Діффі-Хелмана для формування ключа [9].

Припустимо, що 2 користувача, користувач 1 та користувач 2, прагнуть утворити загальний ключ для алгоритму симетричного шифрування. Насамперед вони мусять обрати велике просте число *P* та деяке особливе число *A*, котре *1<A<P- 1*, та таке, що усі числа, які знаходяться в інтервалі можуть бути презентовані як різні ступені *A mod P.* Усім абонентам системи ці числа мають бути відомі і можуть обиратися відкрито. Це будуть спільні параметри.

Після користувач 1 обирає число X_1, таке, що $X_1 < P$, яке потрібно утворювати за допомоги генератора випадкових чисел. Це буде закритий ключ користувача 1 і він мусить зберігатися в секреті. На підставі закритого ключа користувач 1 вираховує число:

$$Y_1 = A^{x_1} \, mod, \tag{2.2}$$

яке він відправляє користувачу 2. Користувач 2 надходить так само, генеруючи X_2 і вираховуючи

$$Y_2 = A^{x_2} \, mod \, P. \tag{2.3}$$

Цей результат відправляється користувачу 1.

У користувачів після цього є наступна інформація, яка наведена у таблиці 2.1.

Таблиця 2.1 – Параметри ключів шифрування

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

З чисел Y_1 і Y_2, а так і з особистих закритих ключів кожний користувач може згенерувати загальний секретний ключ Z для сеансу симетричного шифрування. Користувач 1:

$$Z = (Y_2)^{x_1} \ mod \ P. \tag{2.4}$$

Ніхто, окрім користувача 1, не може цього зробити, так як число X_1 секретно. Користувач 2 зможе отримати таке ж число Z, використовуючи свої закритий та відкритий ключі першого користувача:

$$Z = (Y_1)^{x_2} \ mod \ P. \tag{2.5}$$

Якщо увесь протокол формування загального секретного ключа виконан правильно, то значення Z у обох абонентів мають вийти однаковими. Причому, що найважливіше, не знаючи секретних чисел X_1 і X_2, зловмисник не зможе вирахувати Z. Він може спробувати обчислити Z, не знаючи X_1 і X_2, використовуючи тільки відкрито переданні значення P, A, Y_1 і Y_2.

В алгоритмі Діффі-Хелмана безпека утворювання загального ключа виходить з того, що, хоч відносно легко вирахувати експоненти по модулю простого числа, але дуже важко вичислити дискретні логарифми. Ця авдача вважається нерозв'язною для великих простих чисел розмір яких складає сотні і тисячі біт, що буде вимагати дуже великого витрачання обчислювальних ресурсів [9].

Перший і другий користувачі можуть застосовувати число Z як секретний ключ як для шифрування даних, так і для розшифрування.

Будь-яка пара абонентів каким чином може вирахувати секретний ключ, який знаний тільки їм.

На рисунку 2.1 наведено принцип роботи алгоритму RSA.

Рисунок 2.1 – Принцип роботи RSA

Спочатку відбувається генерація пари ключів – відкритого і закритого. Генерація здійснюється наступним способом:

– вибираються два простих великих числа p і q, при цьому вони не рівні;

– обчислюється модуль числа

$$N = p \times q; \tag{2.6}$$

– обчислюється значення функції Ейлера від модуля числа N

$$\varphi(N) = (p - 1) \times (q - 1); \tag{2.7}$$

– вибирається деяке число e – відкрита експонента – яке лежить в інтервалі $1 < q < \varphi(N)$ і є взаємно простим із значенням функції $\varphi(N)$.

– обчислюється число d – секретна експонента, причому воно є мультиплікативно зворотнім до числа e по модулю $\varphi(N)$

$$d \times e = 1\ (mod\ \varphi(N)). \tag{2.8}$$

В результаті виходить пара ключів: *(e, N)* – відкритий ключ і *(d, N)* – закритий ключ.

Користувач А і користувач В обмінюються повідомленнями в інтернеті. Щоб підтримувати переписку в секреті, вони використовують шифрування. Користувач В заздалегідь згенерував пару ключів, а потім передав відкритий ключ користувачеві А, який відправляє зашифроване повідомлення [32].

Шифрування: Користувач А шифрує повідомлення m за допомогою відкритого ключа другого користувача *(e, N)* і відправляє його:

$$c = e\ m = m^e\ mod\ (N). \tag{2.9}$$

Розшифрування: Прийнявши зашифроване повідомлення, користувач В розшифровує його, використовуючи закритий ключ *(d, N)*:

$$m = d\ c = c^d\ mod\ (N). \tag{2.20}$$

Blowfish є мережею Фейстель (Feistel), яка утворюється з 16 стадій. На вхід надається 64-бітовий елемент даних *x*.

Для шифрування:

– розбити *x* на дві 32-бітових частини;

– для i = 1 до 16;

– $x_L = x_L\ A\ P_{18}$;

– $x_R = F(x_L)\ A\ x\ R$;

– представити x_L и x_R (крім останнього етапу);

– $x_R = x_R\ A\ P_{17}$;

– $x_L = x_L\ A\ P_{18}$;

– об’єднати x_L и x_R.

На рисунку 2.2 наведено функцію *F*.

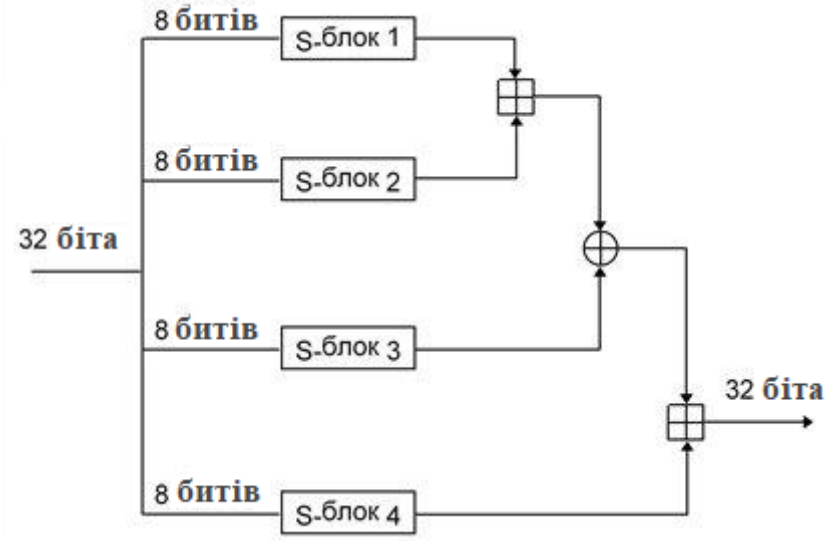

Рисунок 2.2 – Функція F

Розділити x_L на чотири 8-бітові частини: a, b, c і d

$$F(x_L) = \left(\left(S_{1,a} + S_{2,b} mod 2^{32}\right) A\ S_{3,c}\right) + S_{4,d} mod 2^{32}. \quad (2.11)$$

Дешифрування виконується таким же чином, що і шифрування, але P_1, P_2,. . ., P_{18} використовуються в протилежному порядку. У реалізаціях Blowfish, де потрібно дуже велика скорість, цикл мусить бути розгорнутий, а всі ключі мають зберігатися в кеші.

За допомогою спеціального алгоритму розраховуються підключення. Точна послідовність дій, що наведена нижче.

Етап.1. Спочатку *P*-масив, а потім чотири *S*-блоку по порядку ініціалізуються фіксованим рядком. Цей рядок складається з шістнадцяткових цифр *p*.

Етап.2. Виконується XOR P_1 з першими 32 бітами ключа, XOR P_2 з другими 32 бітами ключа, і так далі для всіх бітів ключа (до P_{18}). Використовується циклічно, поки для всього *P*-масиву не буде виконана операція XOR з бітами ключа.

Етап.3. Використовуючи підключи, отримані на етапах (1) і (2), алгоритмом Blowfish шифрується рядок з самих нулів.

Етап.4. P_1 і P_2 замінюються результатом етапу (3).

Етап.5. Результат етапу (3) шифрується за допомогою алгоритму Blowfish і змінених подключей.

Етап.6. P_3 і P_4 замінюються результатом етапу (5).

Етап.7. Далі в ході процеса усі елементи *P*-масиву і згодом по порядку усі чотири S-блока замінюються виходом постійно змінного алгоритму Blowfish.

Усього для формування всіх необхідних подключей потрібна 521 ітерація. Підключи можуть зберігатися додатками – немає потреби виконувати процес їх отримання багатократно [30].

Вважається, що використання в Advanced Encryption Standard ключа довжиною в 128 біт – є досить надійним захистом проти лобової атаки, тобто з суто математичної точки зору підібрати один правильний пароль з усіх можливих – важко здійснюваним завдання. Навіть незважаючи на деякі недоліки AES, зламати інформацію, яка захищена за допомогою цього алгоритму, практично нереально. Будь-який криптографічний алгоритм вимагає ключ розміром в ту чи іншу кількість біт, щоб зашифрувати дані, як показано на рисунку 2.3.

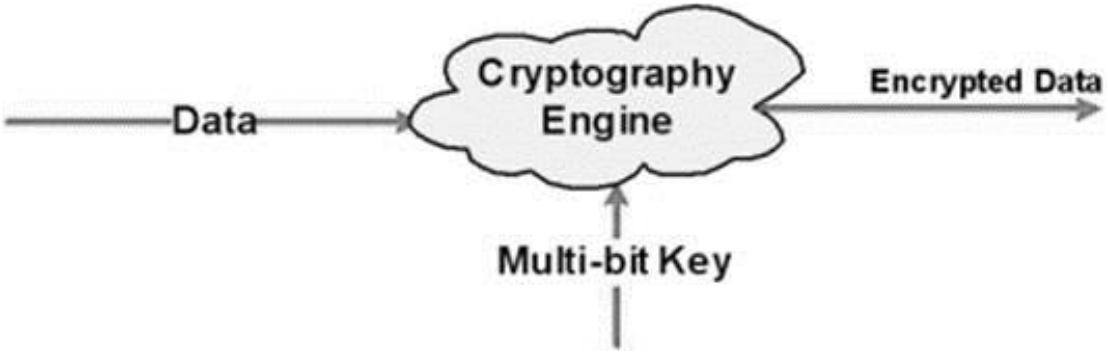

Рисунок 2.3 – Принцип роботи AES

Довжина ключа, що використовується при шифруванні і визначає практичну доцільність виконання повного перебору, адже інформацію зашифровану довшими ключами експоненціально складніше зламати, ніж з короткими [31]. Приклад перебору 4-бітного ключа наведено на рисунку 2.4:

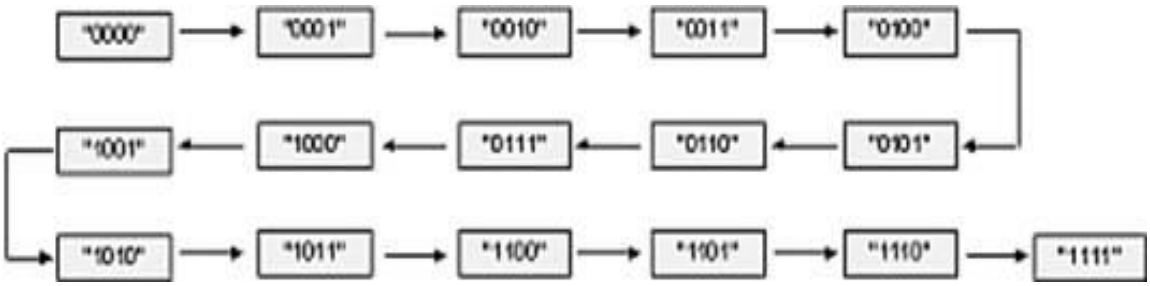

Рисунок 2.4 - Приклад перебору 4-бітного ключа

Буде потрібно максимум 16 стадій, щоб перевірити кожну можливу комбінацію, починаючи з «0000». Лобова атака за деякий час може пробити такий простий алгоритм. Зверніть увагу на те, що в міру збільшення розміру ключа кількість комбінацій зростає експоненціально. Математичні обчислення доводять, що розмір ключа в 128 біт надійним чином захищає від лобової атаки. Як бачимо, розміру ключа в 128 біт цілком достатньо, хоча цілком таємна інформація все одно шифрується з розміром в 256 біт.

3 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ

3.1 Вибір критеріїв оцінки алгоритмів шифрування

У криптографії поняття криптографічний стійкість (криптостійкість) традиційно визначається як стійкість до спроб дешифрування повідомлень, тобто кількісна міра того, наскільки легко криптоаналітик може розкрити шифр, яка вимірюється мінімальною кількістю операцій шифрування/розшифрування, необхідних для визначення ключа (або відкритого тексту), при найкращою з атак. Основне правило криптографії стосовно питання криптостійкості полягає в наступному: розтин шифру з метою прочитати закриту інформацію має обійтися набагато дорожче (вартість виражається в деяких ресурсах – число мікросхем для реалізації атаки, обсяг пам'яті і т.ін.), ніж реальна вартість інформації.

Поняття криптостойкости тісно пов'язано зі складністю криптоаналітичної атаки на алгоритм шифрування і характеризується за допомогою трьох величин:

– складність за даними – кількість перехоплених даних, необхідних для успішної криптоаналітичної атаки на алгоритм шифрування;

– обчислювальна складність – кількість операцій, необхідне для успішної атаки на алгоритм шифрування;

– складність по пам'яті – обсяг пам'яті, необхідний для успішної атаки.

Для оцінки стійкості можна взяти максимальну з цих величин або використовувати оцінку на основі узагальненого критерію.

Всі шифри з криптографічнї стійкості діляться на вчинені (що не мають ефективніших алгоритмів розтину, крім методу повного перебору) і недосконалі (не володіють абсолютною стійкістю), стійкість яких грунтується на обчислювальній складності рішення деякої математичної задачі. Для оцінки обчислювальної стійкості використовується кількість операцій *(w)* деякого типу, необхідних для дешифрування повідомлення або визначення ключа. Найбільш зручною операцією для отримання оцінок стійкості є цикл перевірки одного ключа.

Трудомісткість дешифрування залежить від характеру і кількості інформації, наявний в розпорядженні аналітика.

Аналіз на основі тільки шифртекста – аналітик знає механізм шифрування і йому доступний тільки шифртекст розміром n, що відповідає моделі зовнішнього порушника, який має фізичний доступ до лінії зв'язку.

Аналіз на основі заданого відкритого тексту – криптоаналітику відомий шифртекст і та чи інша частка вихідної інформації, а в окремих випадках і відповідність між шифртекст і вихідним текстом, тобто аналітик має у своєму розпорядженні зашифрованим повідомленням розміром n і відповідним йому відкритим текстом. Така атака можлива прішіфрованіі стандартних документів, підготовлених за стандартними формами, коли певні блоки даних повторюються і відомі.

Аналіз на основі довільно обраного відкритого тексту – криптоаналитик може ввести спеціально підібраний їм текст в шифрувальне пристрій і отримати криптограму, утворену під керуванням секретного ключа, тобто в розпорядженні аналітика є можливість отримати результат шифрування для довільно обраного ним масиву відкритих даних розміром n, що відповідає моделі внутрішнього порушника, коли в атаку на шифр залучаються особи, які не знають секретного ключа, але в відповідно до своїх службових повноважень мають доступ до пристрою шифрування.

Аналіз на основі довільно обраного шифртекста – противник має можливість підставляти для дешифрування фіктивні шифртекст, які вибираються спеціальним чином, щоб за отриманими на виході дешифратора текстам він міг з мінімальною трудомісткістю обчислити ключ шифрування, тобто в розпорядженні аналітика є можливість отримати результат розшифрування для довільно обраного ним зашифрованого повідомлення розміром n, що відповідає доступу до пристрою дешифрування.

Атака на основі адаптивних текстів – атакуючий багаторазово підставляє тексти для шифрування (або дешифрування), причому кожну нову порцію даних

вибирають в залежності від отриманого результату перетворення попередньої порції.

Середній обсяг роботи $W(N)$, необхідний для визначення ключа по криптограмі, що складається з N літер, виміряний в елементарних операціях називається робочою характеристикою шифру. Це значення береться як середнє по всім ключам і всіма повідомленнями з відповідними їм ймовірностями.

Нижні межі трудомісткості досягаються при деяких кінцевих значеннях параметра N, тому що при його необмеженому збільшенні трудомісткість аналізу буде прагнути до деякого межі, що зветься досягнутою оцінкою робочої характеристики. Як зазначено вище, для недосконалих шифрів не існує способи отримати точне значення трудомісткості аналізу, всі оцінки базуються на перевірках стійкості шифрів по відомим на поточний момент видам криптоаналізу, і немає гарантії, що в найближчому або більш віддаленому майбутньому не будуть розроблені нові методи аналізу, істотно її знижують. Тому стійкість недосконалих шифрів обґрунтовується емпірично як стійкість до відомих на сьогоднішній день видів криптоаналізу.

В даний час в криптоанализе виділяють наступні основні методи:

– повний перебір можливих ключів;
– частотний аналіз;
– диференціальний метод;
– лінійний метод.

Повний перебір можливих ключів. При криптоаналізі з відомим вихідним текстом, який є найбільш актуальним, єдиним опублікованим методом, що застосовується, залишається повний перебір ключів. Основним недоліком даного методу є експоненціальна залежність часу криптоаналізу від довжини ключа, тому, через велику трудомісткість цей метод обраний в якості верхньої межі при оцінці криптостійкості шифру – якщо завдання злому шифру можна вирішити тільки шляхом повного перебору, реалізація якого буде неприйнятна з фінансових або тимчасових міркуваннь, то даний шифр вважається стійким. Також цей метод

використовується для оцінки ефективності інших алгоритмів криптоаналізу – всі інші методи повинні забезпечувати значно меншу трудомісткість.

Існують оптимізації цього методу, пов'язані з використанням словникових атак. Ідея базується на тому, що відправник повідомлення віддасть перевагу використовувати легко запам'ятовується ключову послідовність на відміну від повністю випадкової. Тому в першу чергу необхідно випробувати найбільш ймовірні ключі (і їх комбінації) зі словника, що може істотно скоротити трудомісткість розтину шифру. Практичні дослідження показали високу ефективність такого підходу.

Частотний аналіз. Частотний аналіз враховує частотні характеристики текстів на реальних мовах, що володіють великою надмірністю і наявністю стійких частот поєднань символів. Принцип методу полягає в підрахунку частот символів в криптограмі і порівняння з обчисленими частотами для моделі відкритого тексту. Перевагою методу є можливість використання для шифрів, що розрізняються алфавітами для відкритого тексту і криптограми. Для побудови математичної моделі відкритого тексту використовується побудова частотних характеристик відкритих текстів, обчислених з необхідною точністю при дослідженні текстів достатньої довжини.

Відкритий текст розглядається як реалізація стаціонарного ергодичного випадкового процесу з дискретним часом і кінцевим числом станів і модель являє собою однорідний ланцюг Маркова. Більш високий порядок наближення відповідає більш зв'язному тексту, що отримується на виході моделі. Критерії розпізнавання будуються або на основі стандартних методів розрізнення статистичних гіпотез, або на основі заборонених *k*–грам (деякі рідко зустрічаються *k*–грами оголошуються забороненими і їх присутність в одержуваному тексті означає отримання «безглуздої» послідовності знаків). При використанні спеціальних алфавітів (для нетекстової інформації) потрібна побудова нових формалізованих критеріїв на відкритий текст, що враховують специфіку формування вхідної послідовності, що само по собі є складним завданням,

пов'язаної з додатковими дослідженнями. Побудова критеріїв для мови, що характеризується відсутністю статистичних залежностей (рівномірним розподілом ймовірностей появи символів), є практично нерозв'язним завданням.

Диференціальний метод криптоаналізу. Диференціальний криптоаналіз (ДКА) – це спроба розкриття секретного ключа блокових шифрів, яка заснована на повторному вживанні криптографічно слабкої цифрової операції шифрування (раундової функції) *r* разів. При аналізі передбачається, що на кожному циклі застовується свій підключ шифрування. ДКА може застовувати як обрані, так і відомі відкриті тексти. Успіх таких спроб розтину *r*-циклічного шифру залежить від існування диференціалів (*r-1*)-го циклу, які мають велику ймовірність. Ідея методу полягає в пошуку диференціалів для останнього циклу, на підставі яких можна визначити циклові з'єднання, після чого процедура повторюється [21].

Відмінна риса диференціального аналізу це те, що він практично не застовує алгебраїчні властивості шифру (лінійність, афинність, транзитивність, замкнутість і т.ін.), а заснован лише на нерівномірності розподілу ймовірності диференціалів.

Лінійний метод криптоаналізу. Метод передбачає, що криптоаналітику відомі відкриті і відповідні зашифровані тексти. Як правило, при шифруванні застовується додавання по модулю два тексту з ключем і операції розсіювання і перемішування. Метод заснований на пошуку кращої лінійної апроксимації (після всіх циклів шифрування) для вираження, що описує вихід шифру, після чого для знаходження кожного біта власне ключа необхідно вирішити систему лінійних рівнянь для відомих лінійних комбінацій цих біт, але ця процедура не представляє складності, так як складність рішення системи лінійних рівнянь описується поліномом не більше третього ступеня від довжини ключа [20].

Автоматизація методів криптоаналізу для перебору ключів за допомогою ЕОМ вимагає створення адекватної математичної моделі відкритого тексту для оцінки одержуваних результатів важливим параметром будь-якого криптографічного алгоритму, крім стійкості, є його швидкодія.

Для отримання об'єктивної оцінки швидкодії необхідно використовувати незалежні від реалізації характеристики алгоритмів шифрування, оскільки оцінка швидкодії алгоритму в програмній реалізації сильно залежить від оптимізації програми та орієнтації на роботу з певною архітектурою процесора, операційною системою, довжиною блоку і т.ін. Для порівняльної оцінки швидкодії розроблених алгоритмів пропонується використовувати методику на основі підрахунку кількості деяких елементарних операцій, які становлять цикл криптографічного перетворення (шифрування 1 блоку відкритого тексту). В результаті аналізу ітеративних блокових шифрів в якості елементарних операцій для оцінки швидкодії пропонується використовувати такі операції:

- гамування (XOR) – *g*;
- додавання – *s*;
- множення – *m*;
- зрушення – *r*;
- таблична підстановка (*S*-блоки алгоритмів) – *t*.

Склад використовуваних для оцінки елементарних операцій визначається найбільшою поширеністю в існуючих блокових шифрах. Для бітової перестановки, що зустрічається в деяких шифри, як найближеної оцінки пропонується використовувати операцію зсуву по числу бітів перестановки. Операції додавання і множення виконуються за певним модулем, що збігається з довжиною розрядної сітки, необхідної для подання даних, якими оперує алгоритм: якщо алгоритм орієнтований на роботу з 16-розрядними даними, то додавання і множення проводиться за модулем 216, для 32-розрядних – по модулю 232 і т.д.

Як прототипи для порівняння були обрані алгоритми DES, NewDES, оскільки для цих алгоритмів відомі характеристики стійкості і швидкодії і вони є найбільш дослідженими криптографічними алгоритмами. Алгоритм RSA, який є несиметричним алгоритмом, вимагає іншого підходу до оцінки.

Раунд алгоритму DES з урахуванням розширює і стискає табличні підстановки, складання з ключем і бітової перестановки можна охарактеризувати як:

$$Round_{DES} = 16 \times t + 2 \times g + 32 \times r. \quad (3.1)$$

З урахуванням числа раундів, початкової і завершальної перестановок, отримаємо:

$$DES = 2 \times 64 \times r + 16(16 \times t + 2 \times g + 32 \times r) = 640 \times r + 256 \times t + 32 \times g. \quad (3.2)$$

Для алгоритму NewDES, що складається з 17 раундів, раунд можна оцінити як:

$$Round_{NewDES} = 16 \times g + 8 \times f, \quad (3.3)$$

де f – раундова функція, яка об'єднує дані і ключ і складається з операцій гамування та підстановки.

Тоді повністю алгоритм NewDES можна описати таким чином:

$$NewDES = 17(16 \times g + 8(g + t)) = 272 \times g + 136 \times g + 136 \times t. \quad (3.4)$$

Отримані оцінки числа операцій, необхідних для шифрування одного блоку відкритих даних, можна об'єднати в таблиці 3.1.

Для отримання порівняльних оцінок по швидкодії в єдиних одиницях, як таку характеристику можна використовувати число тактів процесора, необхідне для виконання повного циклу криптографічного перетворення. Однак необхідно відзначити, що така оцінка є наближеною і не враховує особливості програмної

реалізації порівнюваних алгоритмів (оптимізації кроків алгоритму, організації структур даних і т.ін.).

Таблиця 3.1 – Порівняльна характеристика швидкодії алгоритмів існуючих блокових шифрів в елементарних операціях

Алгоритм	Швидкодія
DES	$DES = 640 \times r + 256 \times t + 32 \times g$
NewDES	$NewDES = 272 \times g + 136 \times f$

Введені в якості елементарних операції характеризуються різною трудомісткістю і, відповідно, різним числом тактів процесора для виконання в залежності від типу процесора, розміру операндів і їх місцезнаходженням. Як максимальних показників по трудомісткості операцій доцільно використовувати такі характеристики: операції g і s займають 1 такт, операція множення m – від 13 до 42 тактів, операція зсуву r – від 1 до 5 тактів, таблична підстановка t (розглянута як операція *mov*) – від 1 до 7 тактів (основою для таких характеристик служить час виконання цих операцій для процесора i486).

У відповідність з цим розглянемо вплив кожної з цих операцій на швидкодію розглянутих шифрів (див. рис. 3.1 – 3.3).

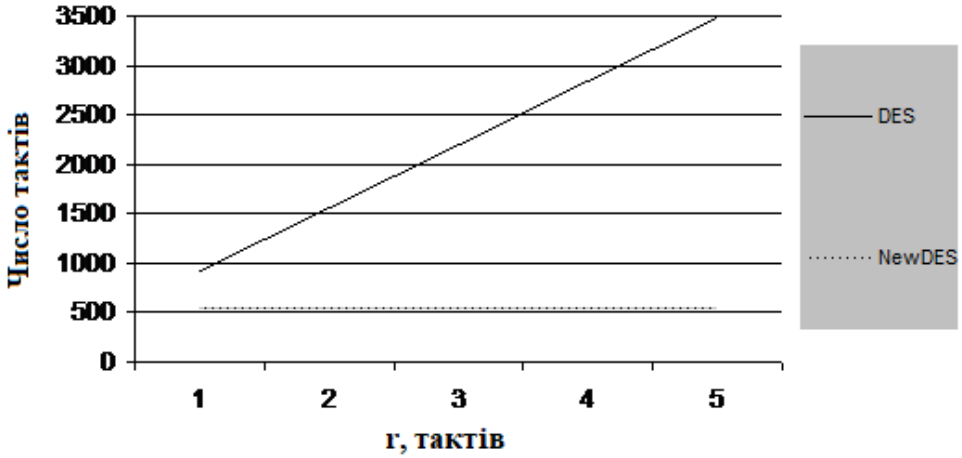

Рисунок 3.1 – Залежність швидкодії алгоритмів від числа тактів процесора, що необхідні для виконання операції зсуву *r (m = 13, t = 1)*

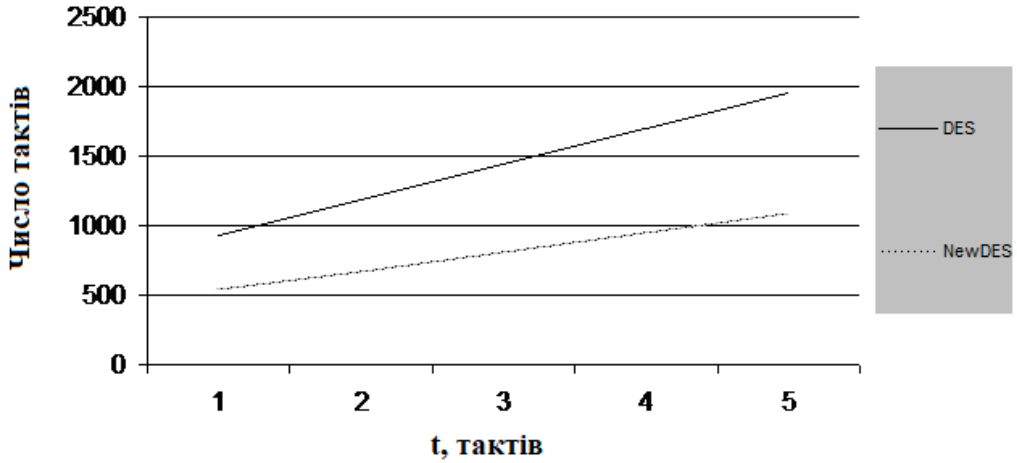

Рисунок 3.2 – Залежність швидкодії алгоритмів від числа тактів процесора, що необхідні для виконання операції пересилання *t (m = 13, r = 3)*

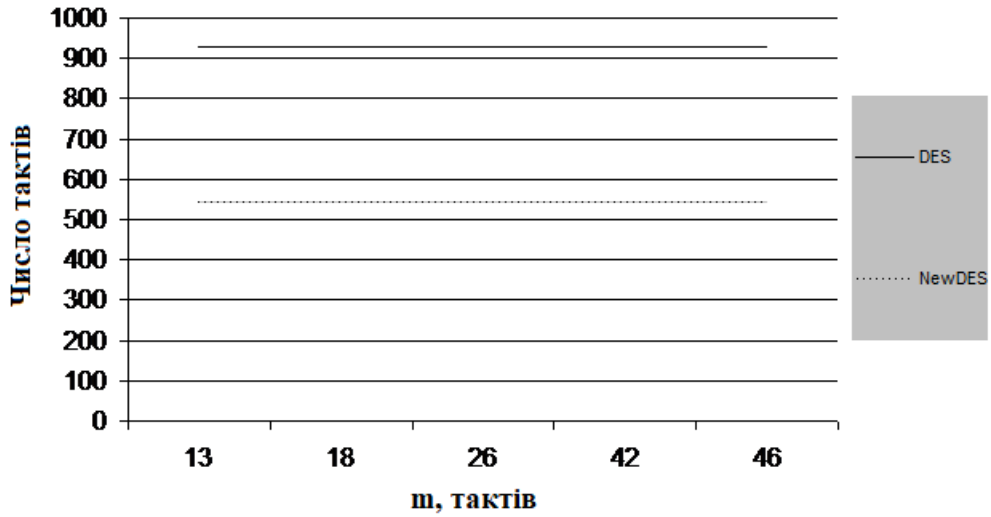

Рисунок 3.3 – Залежність швидкодії алгоритмів від числа тактів процесора, що необхідні для виконання операції множення *m (для t = 1, r = 3)*

Як при шифруванні/дешифрування, так і при створенні і перевірці підпису, алгоритм RSA по сутності, формується з піднесення до ступеню, яке здійснюється як ряд множень [7].

У прикладних додатках для відкритого ключа як правило обирається порівняно малий показник, а найчастіше групи користувачів вживають однаковий відкритий показник, проте кожен з різним модулем. (Однак, запроваджуються

деякі обмеження на головні подільники (чинники) модуля, коли відкритий показник постійний).

При цьому йде скоріше шифрування даних, аніж розшифрування, а перевіряння підпису – скоріше, аніж підписання. Коли k – число бітів в модулі, то як правило в алгоритмах, що використовуються для RSA, число кроків, яке потрібно щоб виконати операції з відкритим ключем, домірно другого ступеня k, а число кроків для операцій приватного ключа – третього ступеня k, число кроків для операції утворення ключів – четвертого ступеня k .

Методи «швидкого множення» – для прикладу, це методи, що засновані на швидке перетворення Фур'є (FFT – Fast Fourier Transform) – здійснюються меншою чисельністю ходів; однак вони не дістали широкого розповсюдження з огляду на складність ПЗ, та ще й тому, що вони реально працюють повільно з типовими розмірами ключів. Проте результативність і ефективність додатків і обладнання, що реалізують алгоритм RSA стрімко збільшується.

RSA алгоритм значно повільний, ніж DES та інші алгоритми блокового шифрування. Програмна реалізація DES працює скоріше, принаймні, в сто разів і від тисячі до десяти тисяч – в апаратній реалізації (в залежності від конкретного пристрою). В наслідок розробок, що активно ведуться, робота RSA алгоритму, імовірно, прискориться, але паралельно прискориться також і робота алгоритмів блочного шифрування [7].

3.2 Вибір алгоритмів для тестування у блокчейні

В якості висновку, зробленого на підставі аналізу різних алгоритмів шифрування, відзначимо, що при захисті інформації в процесі даних по мережі, можна використовувати будь-який з розглянутих алгоритмів.

До переваг алгоритму DES слід віднести його істотно більшу швидкодію. Як і всі блокові алгоритми, він працює в реальному часі, що дозволяє застосовувати їх при шифруванні даних «на льоту» – прозоро для користувача. Швидкодію алгоритму дозволяє реалізовувати підсистеми шифрування даних, які зможуть забезпечити шифрування обсягів даних, відповідних максимальної пропускної здатності каналу зв'язку. Навіть порівняно проста реалізація алгоритму забезпечує швидкість шифрування/дешифрування до 1 Мб/сек.

До недоліків використання даного алгоритму слід віднести те, що він належить до класу симетричних алгоритмів. При прийнятті рішення про вибір даного алгоритму для шифрування інформації, що передається по мережі інформації треба пам'ятати, що реальне використання потребують вирішення питання про передачу пароля на сторону клієнта. У разі якщо зловмисник перехопить шифрований текст, прослуховування телефонних каналів і каналів зв'язку може дозволити йому дізнатися пароль і отримати доступ до зашифрованої інформації. Крім того, використання одного і того ж пароля незручно для користувача, так як вимагає його запам'ятовування. Даний алгоритм можна застосовувати в високопродуктивних захищених мережах, в тих випадках, коли необхідно забезпечити швидке шифрування великих обсягів інформації, не витрачаючи при цьому ресурсів комп'ютера.

На відміну від алгоритму DES, алгоритм RSA належить до класу несиметричних шифрів. Це означає, що у кожного клієнта існує відкритий і закритий ключ, які використовуються при шифруванні. Обмін відкритими ключами може відбуватися з використанням звичайних, відкритих каналів зв'язку, так як інформації про нього недостатньо для дешифрування повідомлення. Для дешифрування повідомлення необхідний як публічний ключ, так і секретний.

Говорячи про стійкість до злому алгоритму RSA, відзначимо, що в даний момент є декілька способів зламування RSA. Особливо дієва атака: винайти секретний (private) ключ, що відповідає необхідному відкритому (public) ключу. Це

дасть дозвіл нападаючому зчитати усі повідомлення, які зашифровувались відкритим (public) ключем і підроблювати підписи.

Практично, завдання відтворення секретного (private) ключа тотожно завданню розкладання на множники (факторингу) модуля: можливо застосовувати *d* для пошуку співмножників *n*, і навпаки можливо застосовувати *n* для пошуку *d*. Потрібно відмітити, що поліпшення обладнання для вирахування саме по собі не послабить стійкість криптосистеми алгоритму RSA,в разі, якщо ключі матимуть задовільну довжину. Практично ж поліпшення устаткування збільшує стійкість криптосистеми.

Другий шлях зламу RSA полягає в знаходженні методу обчислення кореня ступеня. Збагнувши корінь, є моливість відкрити зашифровані повідомлення та підробити підписи, навіть не будучи обізнаним про приватний (private) ключ. Ця атака не є еквівалентною факторингу, проте на даний час незнайомі методи, які б дозволили зламати RSA таким чином. Проте, в особливих випадках, коли на основі однієї і тієї ж ознаки відносно невеликої величини шифрується досить велика кількість зв'язаних повідомлень, є можливість розкриття повідомлення. Наведені атаки – це єдині намагання розшифрувати всі повідомлення, що зашифровані даними ключем RSA [11].

Само собою, існують і атаки, спрямовані безпосередньо на криптосистему, на чутливі місця усієї системи комунікацій в цілому; такі атаки не можуть аналізуватися як злом RSA, так як говориться не про слабкість алгоритму RSA, а швидше про уразливість його певної реалізації.

Для прикладу, нападник зможе заволодіти секретним (private) ключем, якщо такий ключ зберігається без належних захисних заходів. Потрібно акцентувати, що недостатньо захистити виконання алгоритму RSA для повного захисту і застосувати заходи обчислювальної безпеки, іншими словами використовувати ключ достатньої довжини. В реальності ж найбільшу вдачу мають атаки, спрамовані на незахищені стадії управління ключами системи RSA [11].

До одних з найбільш істотних недоліків використання алгоритму RSA надежить його істотна трудомісткість. Криптосистема RSA застовується в різноманітних продуктах, на всіляких платформах і в багатьох сферах. В даний час криптосистема RSA вбудовується в велику кількість комерційних продуктів і їх число постійно збільшується. Операційні системи Apple, Microsoft, Sun і Novell також її використовують. В апаратному виконанні алгоритм RSA вже застосовується в захищених телефонах, на мережевих платах Ethernet та в смарт-картах, обширно вживається в криптографічному обладнанні Zaxus (Racal). Окрім того, алгоритм входить до складу всіх провідних протоколів для захищених комунікацій Internet, в тому числі S/MIME, SSL і S/WAN, а також застосовується в багатьох установах: в державних службах, в великій більшості корпорацій, в державних лабораторіях і університетах [26]. Настільки широке поширення дозволяє зробити висновок про перспективність використання даного алгоритму для шифрування даних, що передаються по мережі.

4 ОПИС РОЗРОБЛЕНОЇ ПРОГРАМНОЇ СИСТЕМИ

4.1 Вибір технологій

Основними перевагами Visual Studio є підтримка його редактора коду рефакторингу коду та компонента доповнення коду IntelliSense. Microsoft Visual Studio також постачається з низкою вбудованих інструментів, таких як дизайнер класів, дизайнер схем баз даних, веб-дизайнер та дизайнер форм, який створює програми GUI.

Компанії можуть робити веб- та мобільні додатки для Android та iOS за допомогою Microsoft Visual Studio. Мови програмування, які підтримує Microsoft Visual Studio, включають VB.NET (через Visual Basic .NET), C, C++ та C++/CLI (через Visual C++), F# (станом на Visual Studio 2010) та C# (через Visual C#). Також підтримуються XML/XSLT, HTML/XHTML, JavaScript та CSS. Підтримку M, Python та Ruby можна встановити окремо.

Отже, основним класом бібліотеки є клас Chain.cs. Він взаємодіє з рівнем даних використовуючи принципи інверсії управління і впровадження залежностей. Таким же чином реалізований підхід до алгоритму хешування [18].

Chain містить в собі список всіх блоків в сховище даних, а також підсписки для поділу блоків за типами (блоки даних, блоки з даними про користувачів, блоки з даними про IP адреси серверів). При створенні екземпляра класу ланцюжка блоків виконується звернення до глобальної мережі для синхронізації з іншими хостами. Також виконується отримання блоків з локального сховища. На даний момент не реалізовано коректного алгоритму злиття цих двох ланцюжків. Вибирається просто вибирається найбільш довгий ланцюг. Якщо не було отримано ні локальної, ні глобальної ланцюжка, то виконується створення нової ланцюжка складається з одного генезис-блоку. Генезис блок – це єдиний початковий блок ланцюжка. Він завжди генерується за ідентичними правилами у всіх примірниках блокчейн. Після

створення екземпляра ланцюжка здійснюється його повна перевірка на коректність, щоб упевнитися, що не було внесено жодних змін.

Клас блок реалізує структуру блоку даних. Він містить в собі всі необхідні поля, які повинні бути збережені в сховище даних. При створенні блоку виконується хешування вхідних даних, а також змішування всього блоку, що дозволяє гарантувати незмінність збережених даних і метаданих. При виконанні хешування беруться до уваги такі поля: версія, час створення блоку, хеш попереднього блоку, хеш збережених даних, хеш користувача. Результат зберігається в поле Hash блоку, що дозволяє легко перевірити його коректність [18].

Клас даних необхідний більше для зручності роботи. Він дозволяє зберігати строкові дані, отримувати хеш збережених даних, а також розділяти збережені дані по тип (дані про користувачів, дані про серверах, прості збережені текстові дані). Для коректного збереження контенту виконується сериализация і десеріалізацію в json формат.

Серіалізація – це процес перетворення об'єкта в потік байтів, десеріалізація – це зворотний процес, що дозволяє сформувати з потоку байтів готовий об'єкт. Якщо сформулювати це більш простою мовою, ми виконуєио збереження всіх значень об'єкта в спеціалізованому тестовому форматі json. Дана тема буде мною докладніше розглянута в одній з моїх наступних статей.

Даний клас є також необхідним тільки для зручності роботи з користувачами системи. Він зберігає важливі дані про користувачів, такі як логін, хеш пароля, права користувача. Дані про користувачів також зберігаються в серіалізовані json форматі, що дозволяє легко зберігати і відновлювати об'єкти.

Звернемо увагу, що важливою особливістю даної реалізації є використання спеціального інтерфейсу IHashable, який повинен реалізовувати всі класи, які піддаються хешуванню (такі як Block, User, Data). Цей інтерфейс реалізує всього одну властивість і один метод. Властивість строкове Hash гарантує, що у класа буде поле, в яке буде виконано збереження готового хешу об'єкта, а метод

GetStringForHash повертає рядок, в якому зберігаються всі тестові дані, важливі для хешування.

Цей інтерфейс використовується в іншому інтерфейсі IAlgorithm. Названий інтерфейс використовується для того, щоб дати можливість подальшого розширення можливостей застосування, шляхом легкого додавання додаткових алгоритмів хешування. Даний інтерфейс визначає один перевантажений метод GetHash, который повертає або хеш стоки, або хеш класу, що реалізовує інтерфейс IHashable.

В даний момент реалізований один алгоритм хешування Sha256 у відповідному класі.

Для коректності роботи класів, що реалізують дані інтерфейси використовується проектування за контрактом. Це дозволяє задати передумови і постумови для всіх класів, які реалізують дані інтерфейси. Завдяки цьому гарантується мінімальна перевірка параметрів у всіх спадкоємців [18].

1.2 Проектування системи

Це лінійна структура даних, як пов'язаний список, з двома важливими властивостями, хеш і попередній хеш. Ви можете уявити хеш блоку як відбиток пальця, який однозначно ідентифікує вміст блоку. Якщо щось змінити всередині блоку, це призведе до зміни хешу (виявлення змін). Попередній хеш створює ланцюжок блоків і робить блокчейн надійним. Якщо ви зміните блок, хеш блоку також зміниться, тому наступний блок матиме недійсний попередній хеш. Таким чином, зміна блоку зробить усі наступні блоки недійсними. Перший блок – це спеціальний блок, оскільки він не може мати дійсний попередній хеш (попереднього блоку немає). Цей блок називається Genesis Block, а попередній хеш встановлюється в рядок однієї або декількох нулів (див. рис. 4.1).

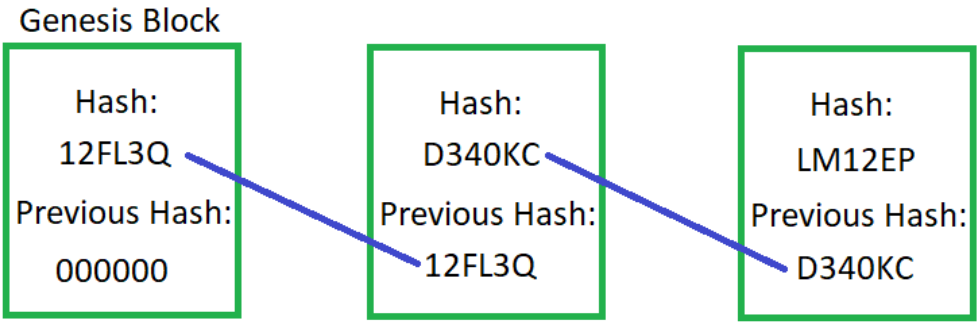

Рисунок 4.1 – Порядок блоків у Блокчейні

І в разі, якщо ми щось модифікуємо у другому блоці, хеш також зміниться, як на рисунку нижче (див. рис. 4.2).

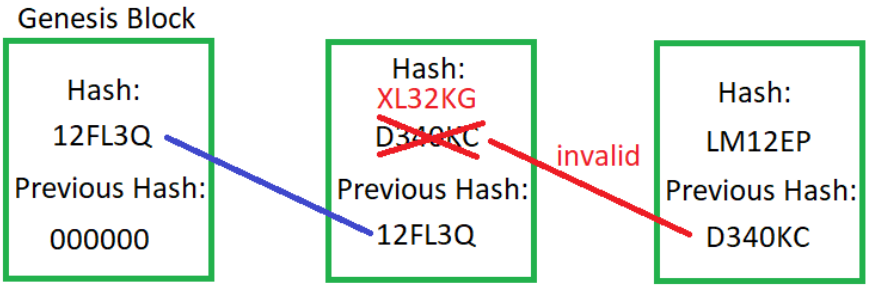

Рисунок 4.2 – Приклад помилки хешу у блокчейні

Ефект цієї незначної модифікації полягає в тому, що наступний блок матиме недійсний попередній хеш, тому всі наступні блоки будуть недійсними. Але цього недостатньо для запобігання фальсифікації. Ви можете сказати, що, змінивши блок, ми перерахуємо всі хеші для наступних блоків. Так, це було б рішенням, тому творці blockchain винайшли деякі правила, роблячи важкий і трудомісткий перерахунок дійсних хешей [13].

Щоб зробити хеш-розрахунок важкою роботою, винахідники blockchain запровадили так званий Proof-of-Work. В основному, перевірка роботи – це лише придумане обмеження, щоб зробити складніший розрахунок. Це обмеження говорить про те, що хеш кожного блоку повинен починатися з *X* числа нулів. Це робить обчислення по-справжньому важким, оскільки у вас немає кращого рішення, ніж груба сила, вгадайте та перевіряйте. Таким чином, ви повинні перевірити всі можливості, щоб знайти хеш, який містить *X* кількість провідних

нулів. Це ще один приклад, коли ми захищаємо щось, використовуючи дуже жорсткі обчислення.

Ще один спосіб блокування ланцюгів – це їх розподіл. Він використовує мережу P2P, де всім дозволено приєднатися. Приєднавшись до цієї мережі, ви отримаєте повну копію блокчейна (на даний момент для Bitcoin блокчейн є приблизно 200 Гб). Коли хтось створить новий блок, він буде відправлений усім учасникам. Кожен одноранговий перевірятиме, що блок не був підроблений, і якщо ні, то він буде доданий до ланцюжка, інакше він буде відхилений. Таким чином, щоб додати блок до ланцюга, 51% учасників повинні погодитися з тим, що блок не був підроблений.

Отже, коли ми створюємо нашу функцію хешування, тепер ми повинні знайти спосіб обчислити хеш нашого блоку. Основна проблема, з якою ми стикаємося – це знайти спосіб обчислити хеш для списку транзакцій, що існують у поточному блоці. Як ви, можливо, вже знаєте, дерево Меркле – це те, що тут вживається у використанні (див. рис. 4.3).

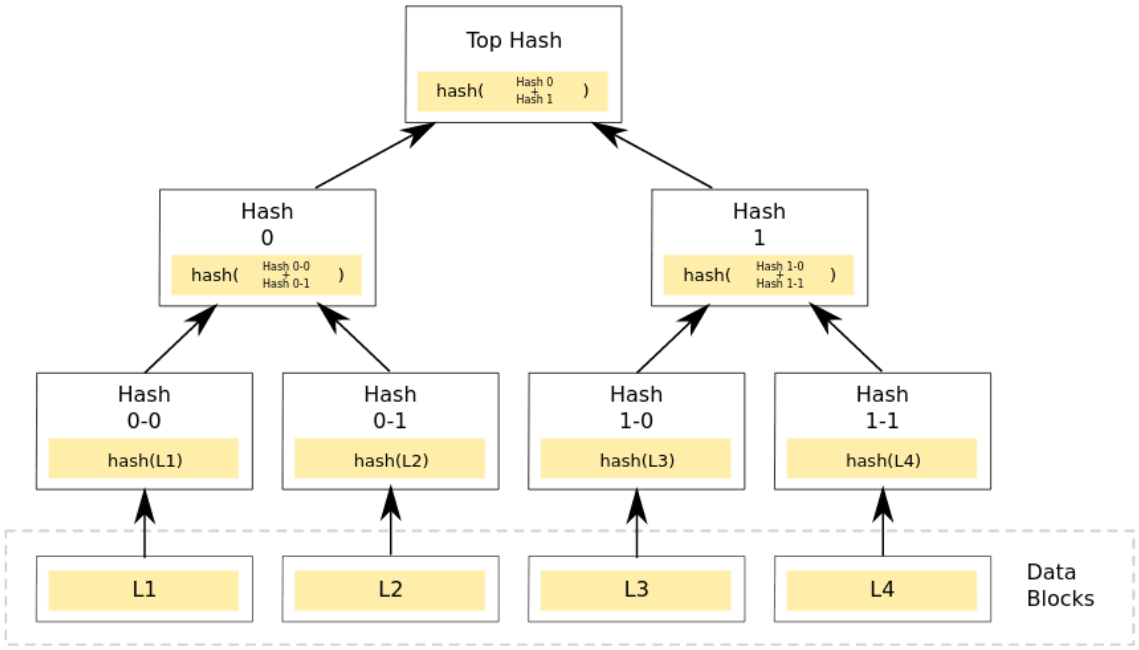

Рисунок 4.3 – Дерево Меркле

Тож для того, щоб «зламати» ланцюг, доведеться не просто перерахувати всі хеши, але й обдурити більше 50% користувачів. Це (наразі) неможливо. Ось чому блокчейн настільки безпечний, його забезпечують не уряд чи інші партії, а важкі математичні проблеми.

4.3 Розробка та впровадження

Блокчейн було створено за допомогою мови програмування C#, блокчейн складається з блоків, які у свою чергу зберігають у собі транзакції, які користувачі відправляють один одному.

На рисунку 4.4 наведено код програми створення блокчейн.


```
{
    "chain": [
        {
            "Index": 0,
            "Timestamp": "2020-05-09T13:22:20.9663666Z",
            "Transactions": [],
            "Proof": 100,
            "PreviousHash": "1"
        },
        {
            "Index": 1,
            "Timestamp": "2020-05-09T16:07:07.9209088Z",
            "Transactions": [
                {
                    "Amount": 1,
                    "Recipient": "9a351b1bb0a44251a6154a11b31852cf",
                    "Sender": "0"
                }
            ],
            "Proof": 14342,
            "PreviousHash": "eb3ebb1768c39df32b915050555817a1c02f3e4f338297b4a931a3b566a3542e"
        },
        {
            "Index": 2,
            "Timestamp": "2020-05-09T16:07:11.2662323Z",
            "Transactions": [
                {
                    "Amount": 1,
                    "Recipient": "9a351b1bb0a44251a6154a11b31852cf",
                    "Sender": "0"
                }
            ],
            "Proof": 145310,
            "PreviousHash": "f32a325c255cc94b7c2e1048ab172b42d4377f188089ccf097985b3124d5584c"
        },
```


Рисунок 4.4 – Блокчейн

У нашому випадку, блокчейн використовується для онлайн гаманця і користувачі відправляють гроші один одному, які додаються у транзакції. У кожного користувача є свій публічний та приватний ключі, маючи публічний ключ іншого користувача, ми можемо відправити йому транзакцію, але не можемо мати доступ до його гаманця, адже для цього потрібен приватний ключ.

До транзакії додаєтсья публічний ключ відпраквника, публічний ключ отримувача, та кількість грошей.

Приклад створення транзакції наведено на рисунку 4.5.

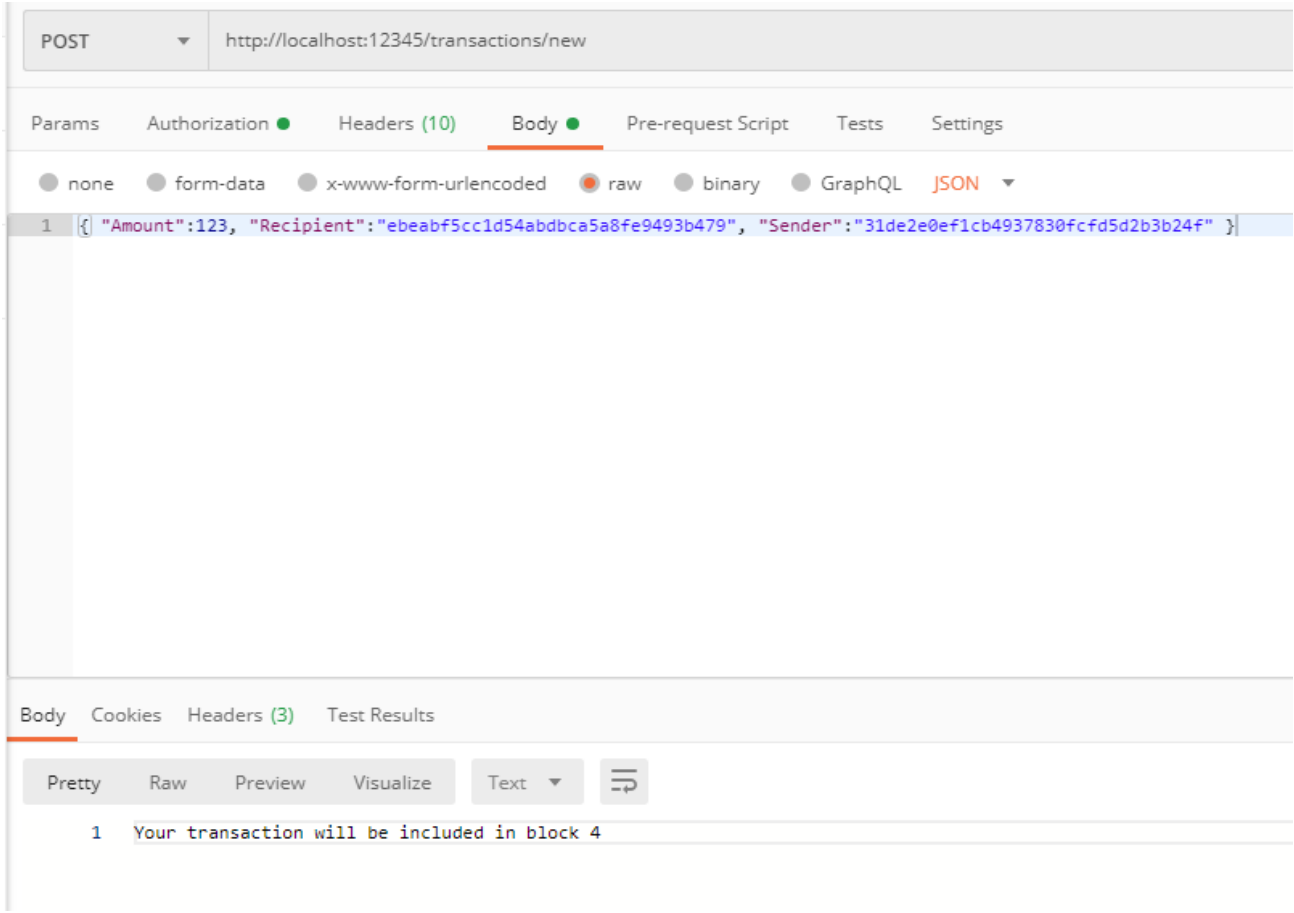

Рисунок 4.5 – Приклад створення транзакції

Після створення транзакції, нам необхідно додати її до блоку, але не практично кожну транзакцію додавати до одного блоку, тому кожен блок має свою розмірність, за замовчуванням, кожен блок важить 2 мб, але може масштабуватися. Отже, після додавання транзакцій на 2 мб, блок можна додавати до блокчейну [28].

Приклад створення нового блоку з доданою транзакцією наведено на рисунку 4.6.

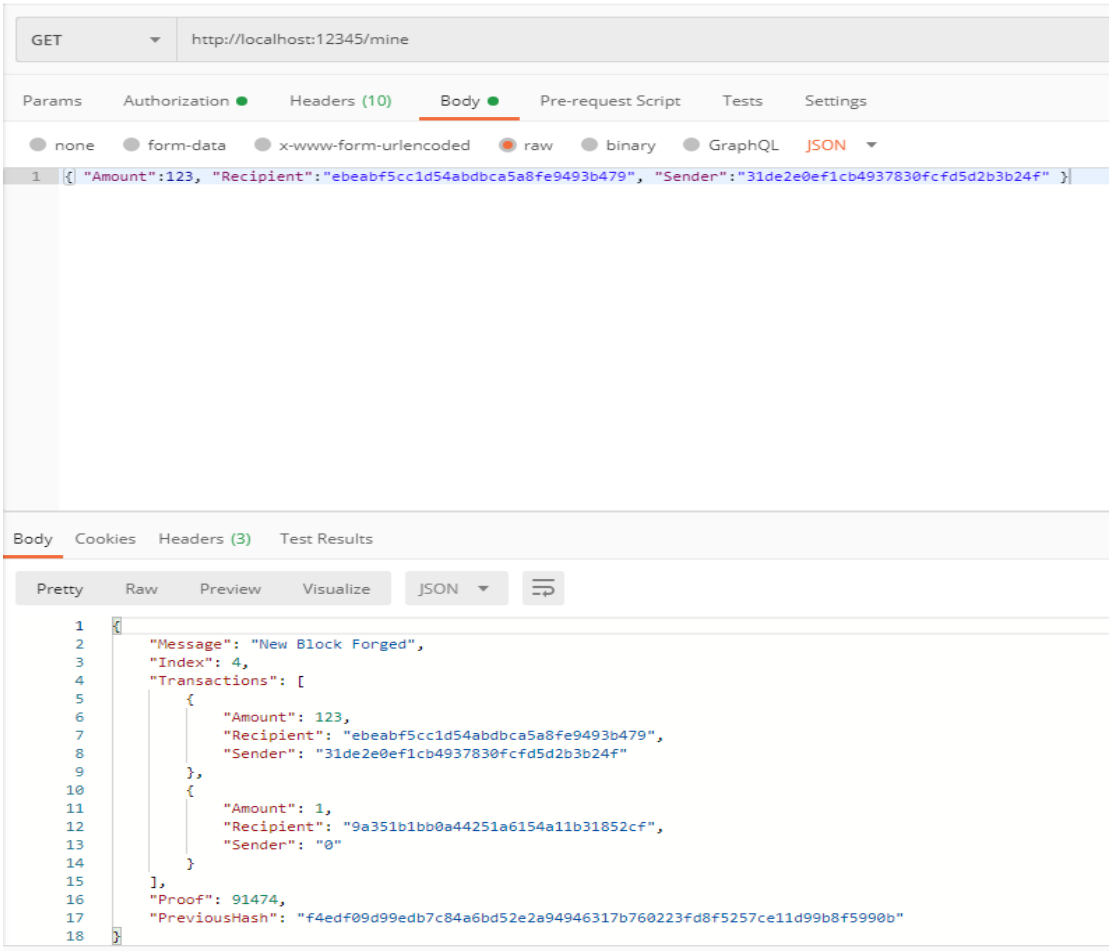

Рисунок 4.6 – Додавання транзакції до блоку

Після того, як блок було створено, потрібно аби усі учасники підтвердили його валідність, кожен бере хєш нового блоку та перевіряє, чи він співпадає, та чи він зберігає в собі хєш минулого блоку. Якщо валідність підтверджую 51% користувачів, блок додається до блокчейну, та усі транзакції вважаються виконаними. Після цього увесь процес починається по новому [10].

Також, завдяки тому, що дані у блокчейні не можна змінювати, або видаляти, нікто не зможе підробити дані, або перехопити транзакцію.

Приклад додавання нового блоку з транзакцією до блокчейну наведено на рисунку 4.7.

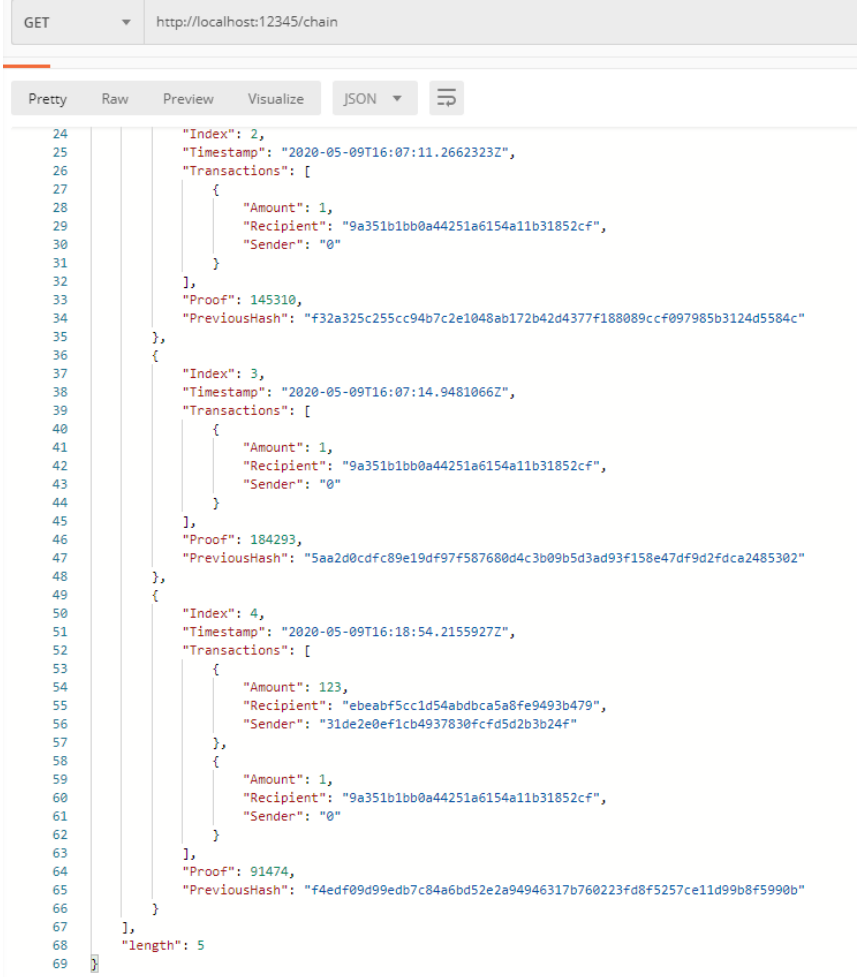

Рисунок 4.7 – Додавання нового блоку до блокчейну

На рисунках 4.8 – 4.10 надано графічний інтерфейс онлайн гаманця та продемонстровано увесь функціонал, який було описано вище.

Рисунок 4.8 – Створення нового блоку, використовуючи інтерфейс

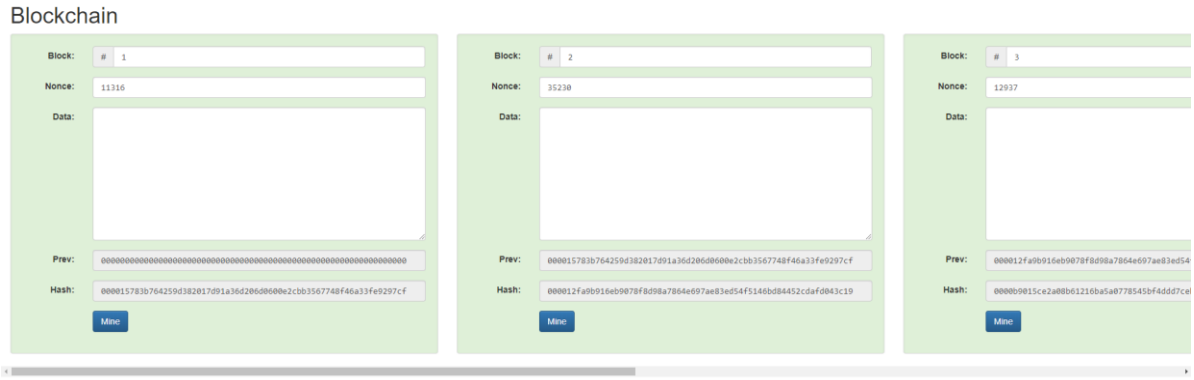

Рисунок 4.9 – Наглядне зображення блокчейну

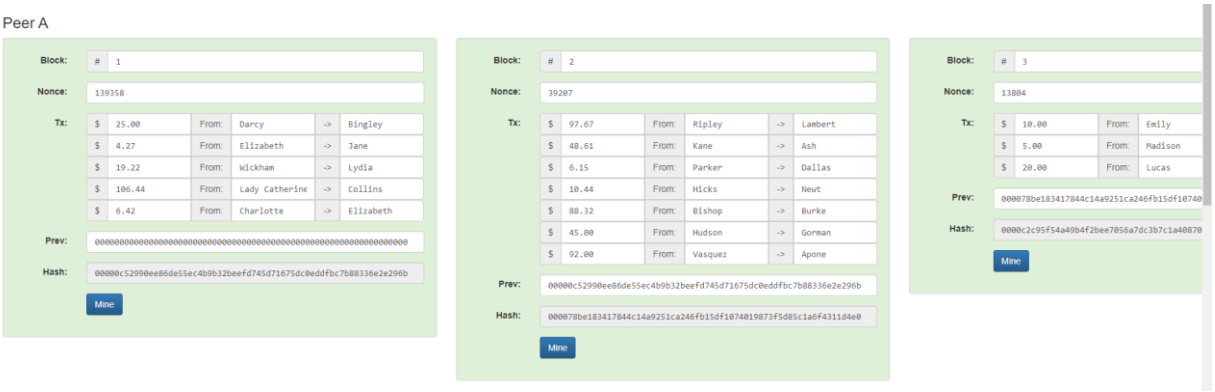

Рисунок 4.10 – Остаточний вигляд блокчейну з транзакціями

Як результат, було розроблено програмну систему на базі блокчейну з використанням різних алгоритмів шифрування та аналізом роботи деяких з них для отримання даних.

5 ОПИС МОЖЛИВОСТІ ВИКОРИСТАННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Кожна система безпеки повинна забезпечувати пакет функцій безпеки, які можуть забезпечити секретність системи. Ці функції зазвичай називають цілями системи безпеки. Ці цілі можна перерахувати за наступними п'ятьма основними категоріями:

– аутентифікація: це означає, що перед надсиланням та отриманням даних за допомогою системи слід підтвердити особу одержувача та відправника;

– секретність або конфіденційність: зазвичай ця функція (функція) – це те, як більшість людей ідентифікують захищену систему, що означає, що лише автентифіковані люди здатні інтерпретувати зміст повідомлення (дати) і ніхто інший;

– цілісність: цілісність означає, що вміст повідомлених даних гарантується таким, що не містить будь-якого типу змін між кінцевими точками (відправник і одержувач), основна форма цілісності - це контрольна сума пакетів у пакетах IPv4;

– непередача: ця функція передбачає, що ні відправник, ні одержувач не можуть помилково заперечувати те, що вони надіслали певне повідомлення;

– надійність та доступність сервісу: оскільки захищені системи зазвичай піддаються нападу зловмисників, це може вплинути на їх доступність та тип послуг для користувачів, такі системи повинні забезпечувати спосіб надання своїм користувачам якості послуг, які вони очікують.

Таблиця 5.1 містить показники швидкості для деяких найпоширеніших криптографічних алгоритмів. Усі були закодовані в C#, складені з Microsoft Visual Studio (оптимізація всієї програми, оптимізація для швидкості, генерування коду P4) та працювали на процесорі Intel Core I5 2.1 ГГц під Windows 10, підпрограми використовувались для багатоточного додавання та віднімання. Властивості SSE2 використовувались для багатоточного множення.

З таблиці 5.1 можна помітити, що не всі режими були випробувані для всіх алгоритмів. Тим не менш, ці результати добре мають вказувати про те, як повинні виглядати представлені результати порівняння.

Таблиця 5.1 – Результати порівняння

Алгоритм	МБ/Секунду	Затрачений час (мілісекунди)	Опрацьовано магабайт (2^20 байтів)
Blowfish	64.386	3.976	256
AES (128-bit key)	61.010	4.196	256
AES (192-bit key)	53.145	4.817	256
AES (256-bit key)	48.229	5.308	256
AES (128) CTR	57.710	4.436	256
AES (128) OFB	52.925	4.837	256
AES (128) CFB	47.601	5.378	256
AES (128) CBC	55.447	4.617	256
DES	21.340	5.998	128
(3DES)DES-XEX3	20.783	6.159	128
(3DES)DES-EDE3	9.848	6.499	64

Також показано, що Blowfish та AES мають найкращі показники серед інших. І обидва, як відомо, мають краще шифрування (тобто сильніші проти атак даних), ніж інші два.

У таблицях 5.2 і 5.3 наведені результати експериментів, які проводилися на двох різних машинах: I-5 2,66 МГц і I-5 2,4 ГГц.

Таблиця 5.2 – Порівняльні часи виконання (у секундах) алгоритмів шифрування в режимі ECB на машині I-5 2,66 МГц

Вхідний розмір (байт)	AES	BF	DES	3DES
20,527	39	19	24	72
36,002	74	35	48	123
45,911	94	46	57	158
59,852	125	58	74	202

Кінець Таблиці 5.2

Вхідний розмір (байт)	AES	BF	DES	3DES
69,545	143	67	83	243
137,325	285	136	160	461
158,959	324	158	190	543
166,364	355	162	198	569
191,383	378	176	227	655
232,398	460	219	276	799
Середній час	228	108	134	383
Байт/сек	491	1,036	835	292

Таблиця 5.3 – Порівняльний час виконання (в секундах) алгоритмів шифрування в режимі ECB на машині I-5 2,4 ГГц

Вхідний розмір (байт)	AES	BF	DES	3DES
20,527	4	2	2	7
36,002	6	3	4	13
45,911	8	4	5	17
59,852	11	6	7	23
69,545	13	7	9	26
137,325	26	14	17	51
158,959	30	16	20	60
166,364	31	17	21	62
191,383	36	19	24	72
232,398	44	24	30	87
Середній час	21	11	14	42
Байт/сек	5,320	10,167	7,988	2,663

З результатів легко помітити, що Blowfish має перевагу перед іншими алгоритмами з точки зору пропускної здатності. Результати показали, що Blowfish має дуже хороші показники порівняно з іншими алгоритмами. Також було показано, що AES має кращі показники, ніж 3DES та DES. Дивовижно це також

показує, що 3DES має майже 1/3 пропускної здатності DES, або іншими словами, для обробки однакового обсягу даних йому потрібно 3 рази більше часу, ніж DES.

Були проведені експерименти для порівняння продуктивності різних алгоритмів шифрування, реалізованих всередині .NET Core. Їх результати близькі до показаних у таблицях 5.1–5.3 (див. рис. 5.1).

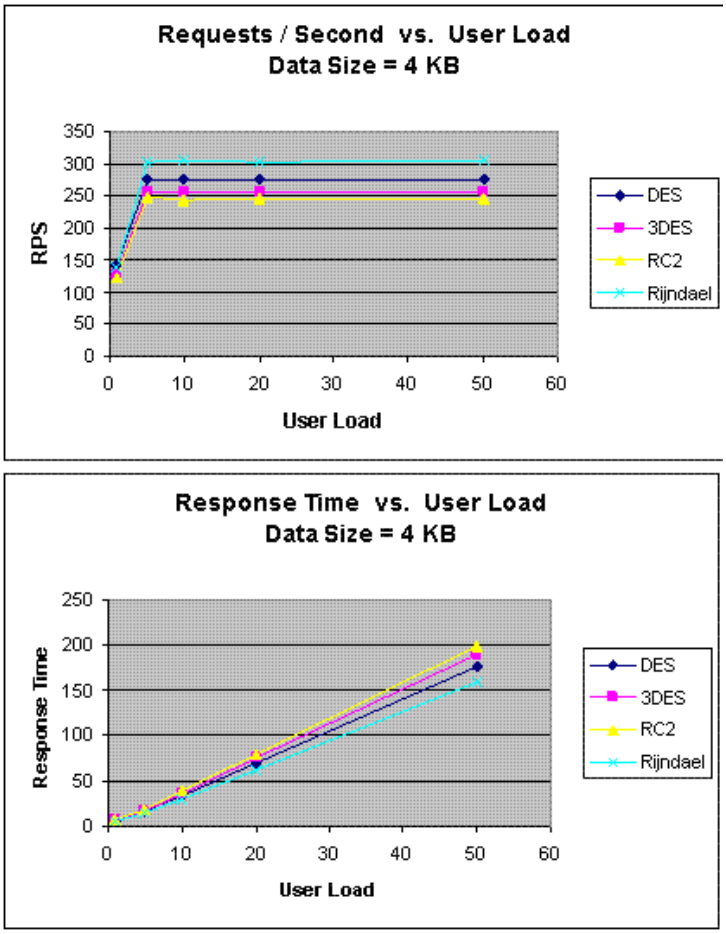

Рисунок 5.1 – Результати порівняння

Порівняння проводили за такими алгоритмами: DES, Triple DES (3DES), RC2 та AES (Rijndael). Результати показують, що AES випереджав інші алгоритми як у кількості процесів запитів в секунду при різних навантаженнях користувачів, так і в часі відповіді в різних ситуаціях із завантаженням користувача.

У реалізації реалізовані керовані обгортки для DES, 3DES та Rijndael, доступні в System.Security.Cryptography, які охоплюють некеровані реалізації, доступні в CryptoAPI. Це DESCryptoServiceProvider,

TripleDESCryptoServiceProvider і RijndaelManaged відповідно. У System.Security.Cryptography, яка була використана в тестах, є лише чиста керована реалізація Rijndael.

У таблиці 5.4 показані параметри алгоритмів, що використовуються в цьому експерименті.

Таблиця 5.4 – Налаштування алгоритмів

Алгоритм	Розмір блока (Біт)	Розмір ключа (Біт)
DES	64	64
3DES	64	192
Rijndael	128	256
Blowfish	64	448

3DES та AES підтримують інші налаштування, але ці параметри представляють максимальні параметри безпеки, які вони можуть запропонувати. Більша довжина ключів означає, що потрібно докласти більше зусиль, щоб порушити захист зашифрованих даних.

Оскільки тест оцінювання призначений для оцінки результатів при використанні блок-шифру, через обмеження пам’яті на тестовій машині (1 Гб) тест розбиває блоки даних про завантаження на менші розміри. Дані навантаження поділяються на блоки даних і вони створюються за допомогою класу RandomNumberGenerator, доступного в просторі імен System.Security.Cryptography.

Експерименти проводяться з використанням Intel core i-5 64-бітового процесора з 8 ГБ ОЗУ. Програма моделювання складається за допомогою стандартних налаштувань у .NET 2019 Visual Studio для програм C# Windows. Експерименти були проведені кілька разів, щоб переконатися в тому, що результати є послідовними та справедливими для порівняння різних алгоритмів.

Для оцінки продуктивності порівняних алгоритмів необхідно визначити параметри, для яких алгоритми повинні бути протестовані.

Оскільки особливості безпеки кожного алгоритму як його сили проти криптографічних атак вже відомі та обговорюються. Вибраним фактором для визначення продуктивності є швидкість алгоритму для шифрування/дешифрування блоків даних різного розміру.

Розглядаючи різні розміри блоків даних (від 0,5 МБ до 20 МБ), алгоритми оцінювались з точки зору часу, необхідного для шифрування та дешифрування блоку даних. Усі реалізації були точними, щоб переконатися, що результати будуть відносно справедливими та точними.

Програма приймає три входи: алгоритм, режим шифрування та розмір блоку даних. Після успішного виконання відображаються дані, що генеруються, шифруються та розшифровуються. Зауважте, що більшість персонажів не можуть з'являтися, оскільки вони не мають представлення символів. Ще одне порівняння проводиться після успішного процесу шифрування/дешифрування, щоб переконатися, що всі дані обробляються правильним шляхом, порівнюючи згенеровані дані (оригінальні блоки даних) та дешифрований блок даних, що генерується в процесі.

Перший набір експериментів проводили в режимі ECB, результати показані на рисунку 5.2. Результати показують перевагу алгоритму Blowfish над іншими алгоритмами з точки зору часу обробки. Це також показує, що AES споживає більше ресурсів, коли розмір блоку даних порівняно великий. Результати, показані тут, відрізняються від результатів, отриманих раніше, оскільки розміри блоків даних, використані тут, значно більші, ніж ті, що використовуються в їх експерименті.

Тут також можна помітити, що 3DES вимагає завжди більше часу, ніж DES через свою характеристику трифазного шифрування. Blowfish, хоча він має довгий ключ (448 біт), перевершує інші алгоритми шифрування. DES і 3DES, як відомо, мають черв'якові отвори у своєму захисному механізмі, Blowfish та AES, з іншого боку, поки що не мають.

Ці результати не мають нічого спільного з іншими навантаженнями на комп'ютер, оскільки кожен окремий експеримент проводився кілька разів, в результаті чого був досягнутий майже однаковий очікуваний результат. Впровадження DES, 3DES та AES в .NET вважається найкращим.

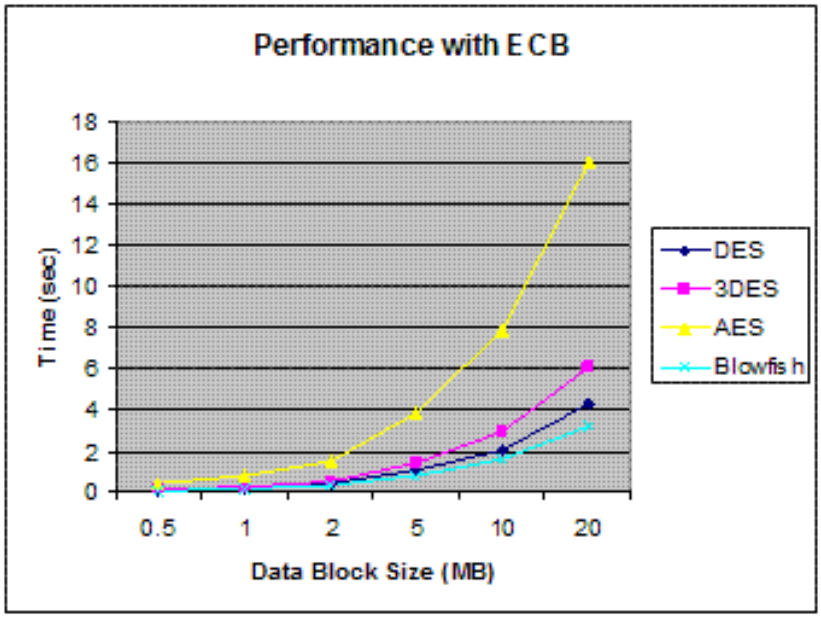

Рисунок 5.2 – Результати роботи в режимі ECB

Як очікується, CBC вимагає більше часу на обробку, ніж ECB, через його ключовий характер. Результати, показані на рисунку 5.3, також вказують на те, що додатковий час, що додається, не є важливим для багатьох застосувань, знаючи, що CBC значно кращий, ніж ECB з точки зору захисту. Різницю між двома режимами важко помітити неозброєним оком, результати показали, що середня різниця між ECB та CBC становить 0,059896 секунди, що порівняно невелика.

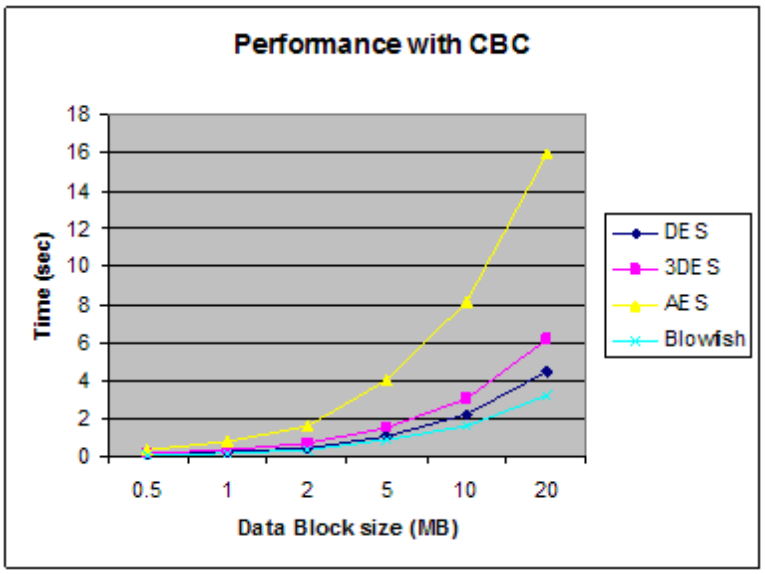

Рисунок 5.3 – Результати роботи в режимі CBC

AES показав низькі результати роботи порівняно з іншими алгоритмами, оскільки він вимагає більшої потужності для обробки. Використання режиму CBC додало додаткового часу на обробку, але в цілому це було відносно незначно, особливо для певного застосування, яке вимагає більш безпечного шифрування до відносно великих блоків даних.

На підставі усіх вищеотриманних даних, було складено таблицю 5.5 з усіма алгоритмами, які можуть використовуватися з технологією блокчейн та надано коротку характеристику кожного з них.

Таблиця 5.5 – Порівняння криптографійних алгоритмів

Алгоритм	Ким створений	Рік	Розмір ключа	Розмір блоку	Раунд	Структура	Складність	Функції
DES	IBM	1975	64 бітів	64 бітів	16	Фейстель	Ні	Не достатньо сильний
DH	Вітфілд Діфф і Мартін Гелман	1976	Змінна	-	-	Фейстель	-	Хороша безпека та швидкість
E-DES	IBM	1977	1024 бітів	128 бітів	16	Фейстель	-	Хороша безпека та швидкість
RSA	Рівест Шамір Адлеман	1977	1024 - 4096	128 бітів	1	Алгоритм відкритого ключа	Ні	Відмінна безпека та низька швидкість

Кінець Таблиці 5.5

Алгоритм	Ким створений	Рік	Розмір ключа	Розмір блоку	Round	Структура	Складність	Функції
T-DES	IBM	1978	112 або 168	64 бітів	48	Фейстель	Так	Відмінна безпека та швидкість
ECC	Ніл Кобліц та Віктор Міллер	1985	Більше, ніж симетричні та змінні	Змінна	1	Алгоритм відкритого ключа	Так	Відмінна безпека та швидкість
EEE	Тахер Ельгамал	1985	1024 бітів	-	-	Алгоритм відкритого ключа	Так	Досить забезпечний і швидкісний
RC4	Рон Рівест	1987	Змінна	40-2048	256	Фейстель потік	Так	швидкий шифр
RC2	Рон Рівест	1987	8,128,64 за замовчуванням	64 бітів	16	Фейстель	-	Гарна та швидка безпека
BLOWFISH	Брюс Шнайер	1993	32-448	64 бітів	16	Фейстель	Так	Швидкий шифр в SSL
SEAL	Філіп Рогавей та Дон Коппер-Сміт	1994	160 бітів	32 бітів	2	Алгоритм відкритого ключа	Так	Не сильна безпека і швидка швидкість
DSA	NIST	1997	Змінна	-	-	Алгоритм відкритого ключа	Так	Хороша безпека та швидкість
RC6	Рон Рівест та ін	1998	128 біт до 256 біт	128 бітів	20	Фейстель	Так	Хороша безпека
AES	Джоан Дейман та Інсент Ріджмен	1998	128,192,256 бітів	128 бітів	10,12, 14	Перестановка заміни	Так	Безпека відмінна. Це найкраще в забезпеченні безпеки та шифрування

Представлені результати моделювання показали, що Blowfish має кращу продуктивність, ніж інші поширені алгоритми шифрування, що використовуються. Оскільки Blowfish досі не має жодних відомих слабких місць безпеки, що робить його відмінним кандидатом, щоб його розглядали як стандартний алгоритм шифрування.

ВИСНОВКИ

В ході написання випускної кваліфікаційної роботи були вирішені всі поставлені завдання.

По-перше, проаналівані і порівняні існуючі алгоритми і системи шифрування, а потім на підставі проведеного аналізу були відібрані чотири алгоритми для подальшого дослідження.

По-друге, були досліджені смарт контракти та можливість поєднання смарт контрактів з обраними алгоритмами шифрування.

По-третє, був розроблений власний блокчейн на прикладі онлайн гаманця для користувачів, в якому усі транзакції будуть проходити через блокчейн.

По-четверте, було вивчено як алгоритми шифрування працюють з алгоритмами та вивчено криптостійкість, та швидкодія цих алгоритмів у блокчейні.

По-п'яте, був обраний найкращий алгоритм для блокчейн – BlowFish, який виявився найбільш захищеним, швидким та показав найкращі результати на нагрузочне тестування.

Нинішній час вимогливо ставиться до технічних рішень, особливо до питання приватності та анонімності даних, в тому числі коли справа стосується фінансової діяльності. Блокчейн технологія і справді є революційною та дає можливість не модифікувати старе ставлення, а подивитися на питання взаємовідносин користувачів та сторонніх інстанцій абсолютно під іншим кутом.

Раніше завдання полягало в тому, як знайти для власної системи найбільш надійних постачальників ресурсів, регуляторів та верифікаторів, тепер – яку форму блокчейн технології використосувати, адже його спільнота ефективно заміщає всі вище перелічені ролі в системі.

Для цього не потрібно будувати модель з довершеною технологією, бо під час реалізації на практиці все одно прийдеться чимось поступатися на перевагу

пріоритетним характеристикам. Саме тому було вибрано синтез адаптивних підходів, що передбачав:

– видокремлення бізнес-стратегій та сфер, де блокчейн не може бути використан чи застосован тільки частково з огляду на певні ліміти технології;

– узагальнене та ретельне оформлення вимог користувачів вищезгаданих систем;

– будову нових моделей блокчейну, які в тих чи інакших умовах сможуть забезпечувати максимум вимог користувачів.

Було підготовлено перелік потенційних рішень на досліджувану тему. Незважаючи на вже реалізовані алгоритми збереження приватності, є щонайменше чотири різних аналізи, які розроблені для виявлення конфеденційної інформації в середовищі цієї криптосистеми. Цей аналіз було успішно здійснено внаслідок прозорості даних у блокчейні, а також питанню ліквідності та ідентифікації поведінки користувачів. Проте потрібно зробити так, щоб система водночас ефективно працювала і при цьому задовольняла широке коло вимог користувачів.

Нові підходи, що орієнтовані не тільки на збереження конфіденційності клієнтів, а й на поліпшення технології блокчейн взагалі, а саме: підвищення децентралізації мережі, швидкодію транзакцій, можливість мікро-операцій тощо. Важливо пам’ятати, що поняття приватності не статична та фіксована річ. Це ціль, за яку постійно ведеться боротьба між криптографами та зловмисниками. Абсолютної приватності не існує. Тому наша задача – адаптуватися до технологічного середовища, яке змінюється, та пропонувати нові алгоритми й підходи, що будуть забезпечувати надійність та ефективність роботи системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Асиметрична схема. URL: https://helpiks.org/4-30251.html (дата звернення: 06.03.2020).

2. Використання хеш-функцій. URL: https://helpiks.org/4-30249.html (дата звернення: 15.03.2020).

3. Про електронний цифровий підпис. // База даних «Законодавство Уркаїни» / ВР України. URL: https://zakon.rada.gov.ua/laws/show/852-15 (дата звернення: 15.03.2020).

4. Ефект від впровадження wms. URL: https://stud.com.ua /172359/logistika/efekt_vprovadzhennya (дата звернення: 10.03.2020).

5. Єлізаров А. Б. Аналіз алгоритмів електронно-цифрового підпису підчас передачі інформації. URL: http://www.rusnauka.com/22_AND_ 2016/Informatica/4_215303.doc.htm (дата звернення: 03.03.2020).

6. Качко О. Г. Оптимизация алгоритмов для современных стандартов метода RSA // Журнал «Прикладная радиоэлектроника». 2008. С. 252–256.

7. Кесавулу Р. Анализ производительности криптографических алгоритмов в информационной безопасности. URL: https://www.ijert.org/performance-analysis-of-cryptographic-algorithms-in-the-information-security (дата звернення: 01.04.2020).

8. Клімушин П. С. Механізми забезпечення довіри в національній системі електронних цифрових підписів. URL: http://www.kbuapa.kharkov.ua/e-book/tpdu/2013-2/doc/1/08.pdf (дата звернення: 05.04.2020).

9. Комп’ютерна криптографія. URL: http://dspace.tneu.edu.ua/retrieve/ 49411/%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%97_%D0%9A%D0%9 A.pdf (дата звернення: 05.04.2020).

10. Комплекс навчально-методичного забезпечення навчальної дисципліни «Технологія конструктивного мислення для створення програмних продуктів» підготовки магістра спеціальність 121-Інженерія програмного забезпечення :

спеціалізація «Програмне забезпечення систем». – ХНУРЕ: розроб. В. І. Каук. – Харків, 2017. – 117 с.

11. Криптоаналіз блочних та потокових шифрів. URL: https://infopedia.su/8xb3f.html (дата звернення: 15.04.2020).

12. Лапіна М. А. Правове регулювання відносин у сфері електронного документообігу. URL: http://yport.inf.ua/pravovoe-regulirovanie-otnosheniy-sfere.html (дата звернення: 15.04.2020).

13. Мазурова О. О. Розширення функціоналу CASE-засобів проектування баз даних // Міжнародна науково-технічна конференція «Інженерія програмного забезпечення 2018». 2018. С. 42.

14. Методичні вказівки до практичних занять з дисципліни «Винахідництво та авторське право» («Інтелектуальна власність») для студентів усіх форм навчання / упоряд. : З. В. Дудар, В. І. Каук, В. В. Голян ; М-во освіти і науки України, ХНУРЕ. – Харків : ХНУРЕ, 2017. – 44 с.

15. Мінгальова Ю. І. Електронний цифровий підпис як головний елемент електронного документообігу. URL: http://eprints.zu.edu.ua/13979/1/Mingaleva1.pdf (дата звернення: 15.04.2020).

16. Назаров А. С., Г. Ю. Терещенко. Blockchain in public administration // Информационные системы и технологии (ИСТ- 2018): материалы 7-й Межд. Науч.-техн. конф. Коблево - Харьков, 2018. МОНУ, ХНУРЭ. 2018. С. 350-351.

17. Назаров А. С., Г. Ю. Терещенко. Decentralized system in IT industry // 23-й Міжн. молодіжний форум «Радіоелектроніка та молодь у XXI столітті», Збірник Матеріалів форуму. 2019. С. 91 – 92.

18. Новітні технології систем криптографії та технології blockchain. URL: http://www.dut.edu.ua/uploads/p_421_86928920.pdf (дата звернення: 18.04.2020).

19. Обґрунтування вимог, побудування та аналіз перспективних симетричних криптоперетворень на основі блочних шифрів. URL: http://ena.lp.edu.ua:8080/bitstream/ntb/27209/1/21-124-141.pdf (дата звернення: 11.04.2020).

20. Ростовцев А. Г. Методи криптоаналізу класичних шифрів. URL: http://elartu.tntu.edu.ua/bitstream/123456789/10553/2/ConfTNTU_2011_Kozak_R-Napriamky_rozvytku_kryptoanalizu_76.pdf (дата звернення: 11.03.2020).

21. Ростовцев А. Г. Програмування, легко про складне. Пошук. Методи криптоаналізу класичних шифрів, Криптографія, Security & Hack. URL: http://easy-code.com.ua/2010/12/metodi-kriptoanalizu-klasichnix-shifriv/ (дата звернення: 10.04.2020).

22. Симетрична схема. URL: https://helpiks.org/4-30250.html (дата звернення: 09.03.2020).

23. Терещенко Г. Ю. Aplication of the technology Blockchain in economy // Матеріали Міжнародної Наук.-техн. конф. «Інформаційні системи та технології» (ІСТ – 2017). 11-16 вересня. Коблево-Харків. 2017. С. 287–288.

24. Терещенко Г. Ю. Blockchain and smart contracts in cargo transportation // Матеріали II міжнародної Науково-практичної конференції «Теорія і практика актуальних наукових досліджень» (м. Одеса, 28-29 квітня 2018 р.). Херсон : Видавництво «Молодий вчений». 2018. С. 163–166.

25. Терещенко Г. Ю. Програмна система для перевірки достовірності якості товарів // Рішення про реєстрацію авторського права на твір № 85231 від 15.06.2018р. Свідоцтво № 79820.

26. Тирупалу У. Анализ производительности криптографических алгоритмов в информационной безопасности // IJERT. 2020. С. 1– 6.

27. Шаронова Н. В., Кириченко І. В., Терещенко Г. Ю. Проблеми і перспективи практичного застосування інформаційної технології blockchain в smart-контрактах // Інтелектуальні системи та інформаційні технологі (ISIT-2019). Матеріали Міжн. Наук.-практ. Конф. Одеса, 19 − 24 серпня 2019 р. С. 214–219.

28. Bashir I. Mastering Blockchain: Distributed ledger technology, decentralization, and smart contracts explained. – 2nd Edition, 2018. – 658 c.

29. Bilous N., Kyrychenko I., Tereshchenko G. Copyright protection using blockchain // Бионика интеллекта. Информация, язык, интеллект. Х.: ХНУРЭ. №1 (92), 2019. С. 52–58.

30. Blowfish. URL: https://uk.wikipedia.org/wiki/Blowfish (дата звернення: 15.04.2020).

31. Polk W., Dodson D., Burr W Cryptographic Algorithms and Key Sizes for Personal Identity Verification. – Kindle Edition, 2020. – 21 p.

32. F. Rodriguez-Henriquez, N. A. Saqib, A. D. Pérez, C. K. Koc. Cryptographic Algorithms on Reconfigurable Hardware (Signals and Communication Technology). – 2007th Edition, 2006. – 362 p.

33. Dooley J. F. A Brief History of Cryptology and Cryptographic Algorithms (SpringerBriefs in Computer Science). – 2013th Edition, Kindle Edition, 2013. – 112 p.

34. Dooley J. F. History of Cryptography and Cryptanalysis: Codes, Ciphers, and Their Algorithms (History of Computing). – Kindle Edition, 2018. – 303 p.

35. Koc C. K. Cryptographic Engineering. – 2009th Edition, 2009. – 540 p.

36. Mehta N., A. Agashe, P. Detroja. Blockchain Bubble or Revolution: The Present and Future of Blockchain and Cryptocurrencies. – Kindle Edition, 2019. – 335 p.

37. Tapscott D., A. Tapscott, J. Cummings. Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World. – 2018. – 432 p.

38. Tereshchenko G. Blockchain in education // 14-th International Scientific Conference «Intellectual Systems for Decision Making and Problems of Computational Intelligence» (ISDMCI'2018'). Conference Proceedings. Kherson: PP Vyshemirsky V. S., 2018. 2018. С. 16– 18.

39. Tereshchenko G. Usage of blockchain technology in marketing // Konferencji Miedzynarodowej Naukowo-Praktycznej «Rzwnój i praktyka. Inżynieria i technologia» (29.12.2017). Zakopane (PL): Wydawca: Sp. z o.o. «Diamond trading tour». 2017. С. 5– 11.