МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СТУДЕНТСЬКА НАУКОВА РОБОТА

на тему

«АЛГОРИТМ ЗАХИСТУ ЦІЛІСНОСТІ ВІДЕОДАНИХ НА ОСНОВІ ТЕХНОЛОГІЇ БЛОКЧЕЙН»

шифр роботи «ВІДЕОЧЕЙН»

АНОТАЦІЯ

Актуальність роботи. В інформаційну епоху існує безліч засобів запису відеофайлів. Це різноманітні прилади відеоспостереження, відеореєстратори, відеокамери та інші засоби фіксації та збереження потоку відеокадрів, які щоденно генерують колосальний обсяг даних на цифрових носіях.

Цілісність відеофайлів вкрай необхідна в ситуаціях, коли вони застосовуються для вирішення спірних моментів або використовуються як складова частина доказової бази, тощо. Оптимальним варіантом для вирішення цього завдання є використання криптографічних хеш-функцій. Незалежно від розміру файлу, розмір хеш-суми становитиме від 128 до 512 біт.

Але даний підхід перевірки цілісності файлу є вразливим до фальсифікації. З розвитком обчислювальних потужностей зростає швидкість злому бітового рядка фіксованої довжини. Це ставить під сумнів цілісність будь-якого відеофайлу.

Вирішенням цієї проблеми є побудова системи захисту цілісності даних на основі технології блокчейн.

Використання технології блокчейн дозволяє вирішити проблему довіри при передачі, збереженні чи використанні відеофайлів зацікавленими сторонами.

Мета і завдання дослідження.

Метою наукової роботи є розробка та дослідження алгоритму забезпечення цілісності відеофайлів на основі технології блокчейн.

Для досягнення мети необхідно вирішити наступні основні завдання:

1. Аналіз технології блокчейн та властивостей відеоданих.
2. Визначення алгоритмів роботи блокчейну, хешування та представлення моделі функціонування системи.
3. Розробка та реалізація алгоритму захисту цілісності відеоданих і дослідження кількісних та якісних параметрів системи.

Використана методика дослідження. У ході досліджень використано алгоритми криптографічного хешування, технологію блокчейн, алгоритми побудови бінарних дерев, методи та алгоритми обробки відеоданих.

Наукова робота складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи 54 сторінки. Основний зміст викладений на 30 сторінках друкованого тексту, містить 18 рисунків та 5 таблиць. Список використаних джерел 23 найменування.

Наукова новизна одержаних результатів полягає в розробці алгоритму захисту цілісності відеоданих на основі технології блокчейн, що забезпечує достовірність відеофайлу або потокових відеоданих за рахунок розподіленого зберігання значень хеш-функцій.

Ключові слова: блокчейн, захист відеоданих, дерево Меркле, цілісність відеопотоку, хешування.

ЗМІСТ

ВСТУП

В даний час значно зросла кількість засобів відеореєстрації подій та процесів, які широко використовуються в різних галузях, зокрема, в системах безпеки дорожнього руху, відеомоніторингу транспорту і спецтехніки, системи контролю доступу та ін. Однак, при використанні результатів відеоспостереження різними зацікавленими сторонами, необхідний інструмент який буде захищати відеоматеріали від несанкціонованої зміни в процесі їх отримання, зберігання та передавання будь-якими користувачами (авторизованими, неавторизованим), тобто забезпечить їх цілісність.

В ролі такого інструменту може виступити технологія блокчейн.

Блокчейн - це розподілена структура даних, що складається з послідовності блоків, в якій кожний блок містить хеш-суму попереднього блоку, утворюючи таким чином ланцюг блоків. Блокчейн працює як розподілена база даних, яка здійснює облік усіх операцій в мережі.

Оскільки хеш-сума кожного відеокадру є складовою частиною при обчисленні хеш-функції наступного кадру, то модифікація будь-якого кадру призведе до зміни хеш-суми всього відеофайлу.

Використання технології блокчейн при захисті відеоданих забезпечить надійне підтвердження достовірності та цілісності відеоінформації.

1 ТЕХНОЛОГІЯ БЛОКЧЕЙН ТА ЇЇ ЗАСТОСУВАННЯ

1.1 Аналіз технології блокчейн

Технологія зберігання інформації блокчейн набула значної популярності в останні роки. Неможливо заперечувати її значимість в суспільстві, в якому збільшується потреба у підтвердженні достовірності і захисті збереженої інформації. Провідні фахівці переконані, що дану технологіюможна впроваджувати в усіх галузях. Використання блокчейна може зробити переворот в концепції урядового управління, економічних послуг, промисловості, тощо [1].

Блокчейн – розосереджений журнал запису транзакцій (мінімальних операцій, завершених повністю), що представляє собою складову більш широкої обчислювальної інфраструктури, яка крім того повинна містити в собі функції зберігання, комунікації, обслуговування файлів і архівації даних про транзакції. Будь-який з блоків пов'язаний з попереднім і має цифровий підпис, що унеможливлює заміну або видалення даних після того, як їх внесли в систему. З цього випливає висновок, що реєстр, який не можна змінити – це дуже потрібна технологія для численних сфер діяльності, і в першу чергу у галузях безпеки інформації та фінансах[2].

Концепція блокчейна була запропонована Сатоши Накамото в 2008-му році. Вперше реалізована вона була в 2009-му році в якості компонента криптовалюти –біткоін. В цьому випадку за допомогою технології блокчейн реєструвалися всі транзакції, що здійснюються з біткоінами. Саме блокчейн дозволив виключити з системи обігу біткоінів третю сторону – центральний сервер, банк або іншу довірену особу.

Початково технологія блокчейн передбачала повну свободу і незалежність ланцюга, в якому немає єдиного адміністратора. Однак, інтерес до нової технології з боку великих компаній і фінансових інституцій призвів до

розробки більш централізованих форм блокчейна, коли при збереженні розподілених даних присутня централізована система контролю.

Такі трансформації дозволяють говорити про різні види блокчейна:

- публічний блокчейн;

- блокчейн, який належить консорціуму;

- повністю приватний блокчейн.

Вони відрізняються рівнем доступу до інформації учасників блокчейн-мережі, а також їх можливістю впливати на хід подій.

На сьогоднішній день існують десятки вдалих реалізації на основі технології блокчейн [3]:

- Ідентифікація людини. На базі технології блокчейн працюють сервіси у сфері ідентифікації та підтвердження прав доступу. Вони створюють цифровий аналог посвідчення особи. До таких платформ входять BlockVerify, OneName, HYRP і інші.

- Авторське право. Стартап Ascribe використовує доповняльний реєстр, в якому музиканти, художники, розробники можуть зберігати авторські права у формі зашифрованих ідентифікаторів.

- Голосування. В даний момент відкритий реєстр використовується лише в приватних голосуваннях. Однак в університеті штату Вірджинія хочуть впровадити технологію, що базується на блокчейн, та забезпечити зниження ймовірності фальсифікації до нуля.

- Юриспруденція та управління. В даному напрямі блокчейн має безмежний потенціал. В ідеальному випадку можливе створення системи зі звітністю представників місцевої і державної влади, зберігання даних про бюджет. Уже зараз є платформи типу Borderless, що об'єднують в собі економічні та юридичні послуги.

- Музична сфера. Стартап Bittunes надає виконавцям композицій можливість зберегти за собою права і займатися продажем власних робіт.

Доступний і ряд інших проектів, націлених на розповсюдження незалежної музики і просування виконавців.

- Благодійність. Блокчейн та його властивість запису і збереження даних дуже ефективний в сфері благодійності. Наприклад, в платформі GiveTrack доступна відкрита інформація про пожертвування до фондів і їх витратах.

- Нерухомість. Інтеграція технології блокчейн в сферу нерухомості здатна її в значній мірі вдосконалити. Це забезпечить прискорення процесів покупки-продажу, з'явиться інструмент надійного зберігання даних про права на власність, тощо.

Переваги та недоліки технологїі блокчейн представлені в таблиці 1.1.

Таблиця 1.1 Переваги та недоліки блокчейну[4]

Недоліки	Переваги
Висока масштабованість – якщо на блокчейн біткоін перенести частину транзакцій Visa, то його розмір становив би сотні терабайт	Децентралізація – прямий обмін даними, рівними між собою учасниками мережі
Шахрайство – передача інформації в мережі блокчейн відбувається необоротно. Це унеможливлює повернення чи скасування операції, навіть якщо вона була проведена помилково	Надійність – хакерські атаки або підміна даних неможлива у зв’язку із специфікою використовуваних технологій
Атака 51% – наявність в блокчейні біткоїну 51% обчислювальних потужностей підконтрольних одній групі чи особі, призведе до порушення цілісності	Прозорість – повний доступ та можливість перегляду будь-якого блоку чи транзакції без жодних обмежень
	Універсальність – технологія блокчейн може використовуватися не лише у фінансовому секторі, а і в інших галузях

1.2 Основні характеристики відеоданих

Цифрове відео – це сукупність технологій запису, передачі, збереження і обробки звуку та зображень. Ключова різниця із аналоговим відео полягає в тому, що звук та відеосигнал кодуються та пересилаються не в початковому вигляді, а лише після аналогово-цифрової модифікації в потоки звукових та відеоданих. У більшості випадків цифрове відео піддається компресії для зменшення обсягу даних, призначених для передачі і зберігання. Цифрове відео може поширюватися на різних відеоносіях, за допомогою цифрових інтерфейсів у вигляді потоку або файлів [5].

Кількість або частота кадрів (fps-frames per second) – це число стаціонарних зображень, які змінюють одне одного при демонстрації однієї секунди відеоматеріалу і створюють процес переміщення предметів в кінематографі і телебаченні [6].

Цифрові комп'ютерні відеодані високої якості зазвичай використовують частоту 30 кадрів в секунду. Граничні частоти миготіння, які сприймається людиною, знаходяться в діапазоні від 39 до 42 Герц. Багато відеокамер нового покоління здатні знімати з частотою до 120 кадрів в секунду. Особливі професійні апарати, призначені для надшвидкої зйомки, здатні знімати з частотою до тисячі кадрів в секунду. Такі зйомки застосовуються в процесі наукових експериментів або детальних дослідів [5].

Коли відбувається оцифровування аналогового сигналу зі стандартною чіткістю, то роздільна здатність становить 720 × 576 пікселів для стандарту розкладання в Європі 625/50 (PAL і SECAM), при частоті кадрів 50 Герц (два поля, 2 × 25); і 720 × 480 пікселів для стандарту розкладання в Америці 525/60 (NTSC), при частоті 59,94 Герц (два поля, 2 × 29,97). Сучасний стандарт цифрового телебачення HDTV з високою роздільною здатністю пропонує до 1920 × 1080 з частотою оновлення 50 Герц (60 Гц для США) з прогресивною розгорткою. Тобто 1920 пікселів на рядок, 1080 рядків [5].

На рисунку 1.1 представлено основні стандарти подання відеоданих [5].

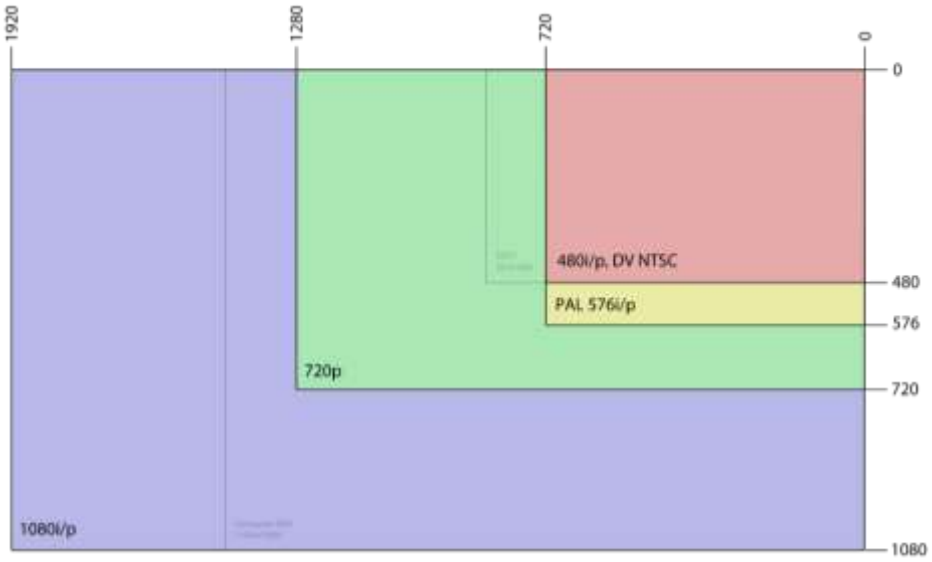

Рисунок 1.1 – Основні стандарти подання відеоданих

Відеопотік – це послідовність кадрів певного формату, закодована в потік бітів. Швидкість передачі нестиснутого відеопотоку з чергуванням рядків розрядністю 10 біт і колірної дискретизації 4:2:2 стандартної чіткості становитиме 270 Мбіт/с. Такий потік виходить, якщо скласти складові частоти дискретизації на розрядність кожної компоненти: 10×13,5+10×6,75×2=270Мбіт/с. Однак, розрахунок розміру отримуваного файлу, що містить нестиснутий відеопотік, виконується іншим чином. Зберігається тільки активна частина рядка відеосигналу. Для подання в просторі Y ', Cr, Cb розраховуються наступні складові [7]:

- число пікселів в кадрі для одної компоненти = 720×576=414720;

- число пікселів в кадрі для усіх кольорових компонент =360×576=207360;

- кількість бітів в кадрі = 10×414720+10×207360×2=8294400=8,29 Мбіт;

- швидкість передачі даних (BR) = 8,29×25=207,36 Мбіт/сек;

- розмір відео = 207,36*3600=746496 Мбіт = 93312 Мбайт = 93,31 Гбайт = 86,9 Гбайт.

Для стиснення цифрових мультимедіа файлів використовуються спеціальні програми – кодеки. Це своєрідна формула, яка визначає, яким чином можна "упакувати" відеоконтент. Кодеки виконують і зворотну операцію розкодування, в цьому випадку їх називають декодерами. Найбільш популярними відеокодеками є наступні: DivX, XviD, H.261, H.263, H.264 та інші [7].

Кодеки перетворюють дані в особливий файл, який називають контейнером. Контейнер – це спеціальна оболонка, в якій зберігається зашифрована за допомогою кодеків інформація. Медіаконтейнери – це і є формати відеофайлів, які містять дані про свою внутрішню структуру.

1.3 Огляд існуючих рішень

1.3.1 Проект OriginStamp

OriginStamp – це веб-сервіс довірених часових міток, який використовує децентралізований блокчейн для зберігання анонімних, захищених від несанкціонованих позначок часу для будь-якого цифрового вмісту[8]. OriginStamp дозволяє користувачам хешувати файли, електронні листи чи звичайний текст, а згодом зберігати створені хеші у блокчейні, а також отримувати та перевіряти часові позначки, які були допущені до блокчейну.

OriginStamp безкоштовний і простий у користуванні. Це дозволяє кожному, наприклад, студентам, дослідникам, авторам, журналістам чи художникам довести, щоб вони були джерелом певної інформації в певний момент часу.

1.3.2 Emernotar – захист документів за допомогою блокчейна

Сервіс Emernotar призначений для захисту цілісності цифрових активів (файлів чи медіа) і авторства за допомогою технології блокчейн. Сервіс побудований на базі технології Emercoin – EmerNVS. З її допомогою можна створювати пари "Ім'я-Значення". "Ім'я" використовується для швидкого

пошуку в блокчейні, а "Значення" зберігає дані, які користувачі додають і оновлюють. Обмеження на тип даних відсутні, тому існує можливість завантажувати документи, мультимедійні файли, статті, контракти, тощо[9].

1.3.3 Проект AcronisNotary

Технологія була реалізована в рамках функціоналу продукту для індивідуальних користувачів Acronis True Image 2017 New Generation. З його допомогою можна перевіряти автентичність збережених в резервних копіях даних[10].

Наприклад, музиканти або художники можуть підтверджувати дату і час створення своїх творів, надаючи в якості доказу сертифікат Acronis Notary[11] із зазначенням даної інформації, поряд з метаданими про сам твір.

2 АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ЗАХИСТУ ЦІЛІСНОСТІ ВІДЕОДАНИХ

2.1 Алгоритм роботи блокчейну

З технічної точки зору, блокчейн є новою технологією, основою якої є наступні поняття: P2P мережі, асиметрична криптографія і розподілений консенсус, заснований на вирішенні математичної задачі. Жодна з цих ідей не є новою сама по собі.

Блокчейн можна розглядати як синхронізовану базу даних реплікованустільки ж раз, скільки вузлів в мережі, або як суперкомп'ютер, утворений комплексом всіх CPU / GPU, що входять в його вузли [12].

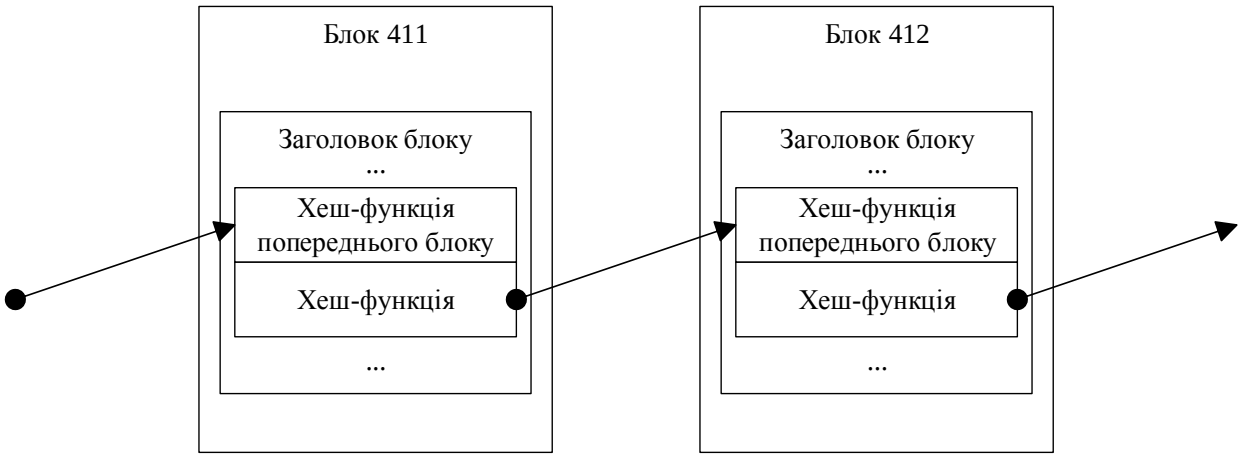

Рисунок 2.1 – Загальна структура ланцюга в блокчейні

Кожен фрагмент ланцюжка містить хеш-функцію заголовку попереднього блоку (рисунок 2.1). Поки він не буде розшифрований, блок не закриється.

Кожен блок посилається на попередній. Тобто, якщо факт F знаходиться в блоці 21, і факт E в блоці 22, то факт E розглядається всією мережею як наступний за фактом F. Перед додаванням до блоку факти знаходяться на розгляді, тобто непідтверджені (рисунок 2.2) [3].

Деякі вузли в ланцюжку створюють нові локальні блоки з непідтвердженими фактами. Вони змагаються, щоб дізнатися, чи стане їх локальний блок наступним блоком в ланцюзі для всієї мережі шляхом кидання

гральних кісток (рисунок 2.3) [4]. Якщо вузол викидає дві шістки, то він отримує можливість опублікувати його локальний блок, і всі факти в цьому блоці стають підтвердженими. Цей блок надсилається всім вузлам в мережі. Всі вузли перевіряють, чи блок правильний, додають його до своїх копій ланцюга і намагаються побудувати новий блок з новими непідтвердженими фактами [3].

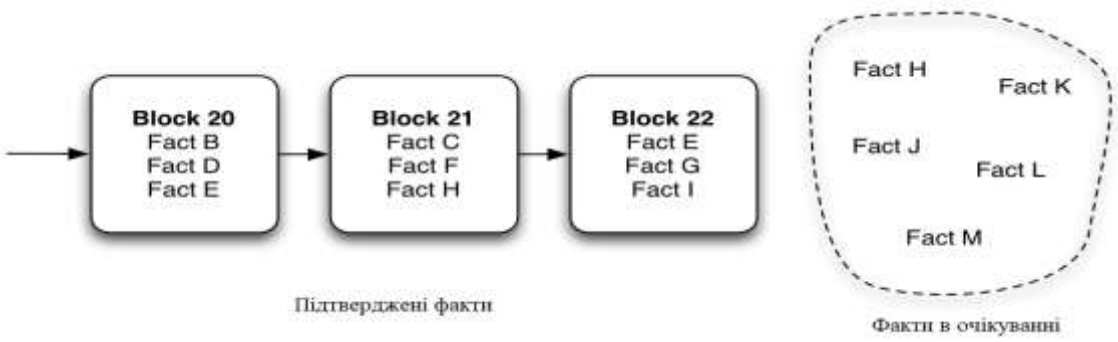

Рисунок 2.2 – Розподіл фактів в блокчейні

Але насправді вузли не просто кидають пару гральних кісток. Завдання, яке вирішують майнери в блокчейні передбачає кидання величезної кількості гральних кісток. За задумом, виявлення випадкового ключа для перевірки блоку малоймовірно. Це запобігає шахрайству і робить мережу безпечною (до тих пір, поки зловмисник не має контролю більш ніж над половиною вузлів в мережі). Як наслідок, нові блоки будуть публікуватися в ланцюг через фіксований інтервал часу. У біткоіні блоки публікуються, в середньому, кожні 10 хвилин [4].

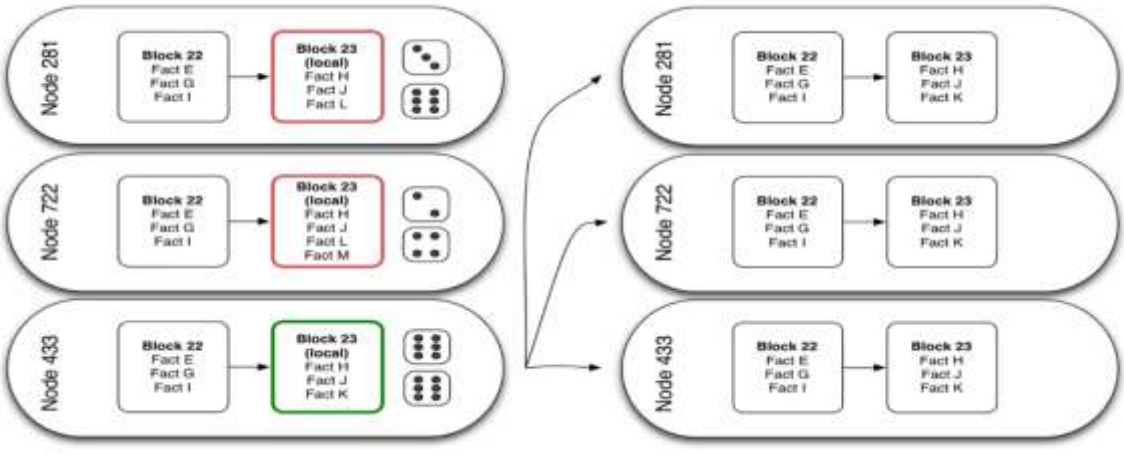

Рисунок 2.3 – Визначення наступного блоку в мережі

Децентралізація блокчейну надає можливість здійснювати передачу даних між суб'єктами, які представляють різні юрисдикції та країни просто за домовленістю між собою. Безпосередньо, без будь-яких регуляторів чи посередників. Блокчейн побудований за принципом, що унеможливлює заблокування операції. Таким чином децентралізація дозволяє кожному користувачеві відчувати себе незалежним [5].

Доступ до блокчейну здійснюється за допомогою спеціальних ключів, які гарантують надійність всієї мережі. Такий ключ є у кожного користувача. Вінявляє собою сукупність криптографічних записів. Кожен ключ унікальний, що гарантує неможливість хакерських атак та заміну даних. Для здійснення атаки хакерам необхідно отримати доступ до всіх учасників децентралізованої мережі.

Механізми, що забезпечують дієздатність і надійність блокчейна – це алгоритми PoW або Proof of Work (виконаної роботи) і PoS або Proof of Stake (підтвердження частки). Вони забезпечують консенсус в блокчейні.

2.2 Алгоритми хешування

Основним примітивом, оброблюваним в блокчейні, є поняття блоку, який представляє собою структуру даних, що включає транзакції, часову мітку і інші важливі дані.

Хеш-функція, або функція згортки – це функція, що здійснює перетворення масиву вхідних даних довільної довжини в бітову строку встановленої довжини, що виконується певним алгоритмом. Перетворення, яке виконує хеш-функція, називається хешуванням. Початкові дані називаються вхідним масивом, «ключем» або «повідомленням». Результат перетворення (вихідні дані) називається «хешем», «хеш-кодом», «хеш-сумою», «зведенням повідомлення» [13].

У загальному випадку в основі побудови хеш-функції лежить ітеративна послідовна схема. Ядром алгоритму є стискаюча функція – перетворення k вхідних біт в n вихідних біт, де n – розрядність хеш-функції, а k – довільне число, більше n. При цьому стискаюча функція повинна задовольняти всім умовам криптостійкості [13].

Вхідний потік розбивається на блоки по (k – n) біт. Алгоритм використовує тимчасову змінну розміром в n біт, за початкове значення якої береться якесь загальновідоме число. Кожен наступний блок даних об'єднується з вихідним значенням функції на попередній ітерації. Значенням хеш-функції є вихідні n біт останньої ітерації. Кожен біт вихідного значення хеш-функції залежить від усього вхідного потоку даних і початкового значення. Таким чином досягається лавинний ефект (рисунок 2.4) [13].

При проектуванні хеш-функцій на основі ітеративної схеми виникає проблема з розміром вхідного потоку даних. Розмір вхідного потоку даних повинен бути кратний (k – n). Як правило, перед початком алгоритму дані розширюються заздалегідь відомим способом.

Крім однопрохідних алгоритмів, існують багатопрохідні алгоритми, в яких ще більше посилюється лавинний ефект. У цьому випадку дані спочатку повторюються, а потім розширюються до необхідних розмірів.

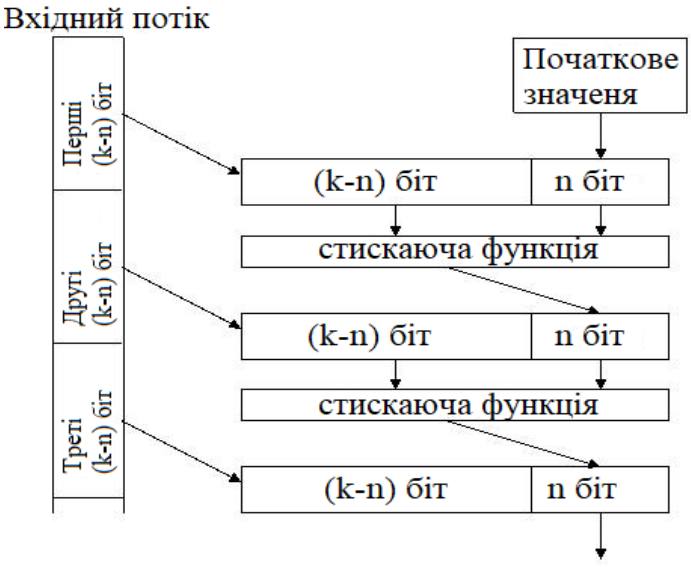

Рисунок 2.4 – Ітеративна послідовна схема

Ідеальною криптографічною хеш-функціяй є така криптографічна хеш-функція яка має п’ять основних властивостей [14]:

1. Детермінованість. При однакових вхідних даних результат виконання хеш-функції буде однаковим (одне і те саме повідомлення завжди породжуєодне й те ж значення хешу);

2. Висока швидкість обчислення значення хеш-функції для будь-якого заданого повідомлення;

3. Неможливість згенерувати повідомлення з його хеш-значення, за винятком спроб створення всіх можливих повідомлень;

4. Наявність лавинного ефекту. Невелика зміна в повідомленнях має змінити хеш-значення так широко, що нові хеш-значення не збігаються зі старими хеш-значеннями;

5. Неможливість знайти два різних повідомлення з однаковими хеш-значеннями.

Таким чином, ідеальна криптографічна хеш-функція, у якій довжина n (тобто на виході n біт), для обчислення прообразу повинна вимагати як мінімум 2^n операцій.

2.3 Модель функціонування системи

Реалізація алгоритму роботи системи представлена у вигляді блок-схеми на рисунку 2.5.

В загальному випадку, алгоритм захисту відеоданих на основі технології блокчейн складається з наступних кроків:

1. вхідний відеоряд розбивається на потік кадрів;

2. для першого кадру обчислюється хеш-сума;

3. хеш-сума кожного наступного кадру складається із поточного кадру та хеш-суми попереднього кадру;

4. кожна хеш-сума записується в дерево Меркле, з якого формується блок для блокчейну.

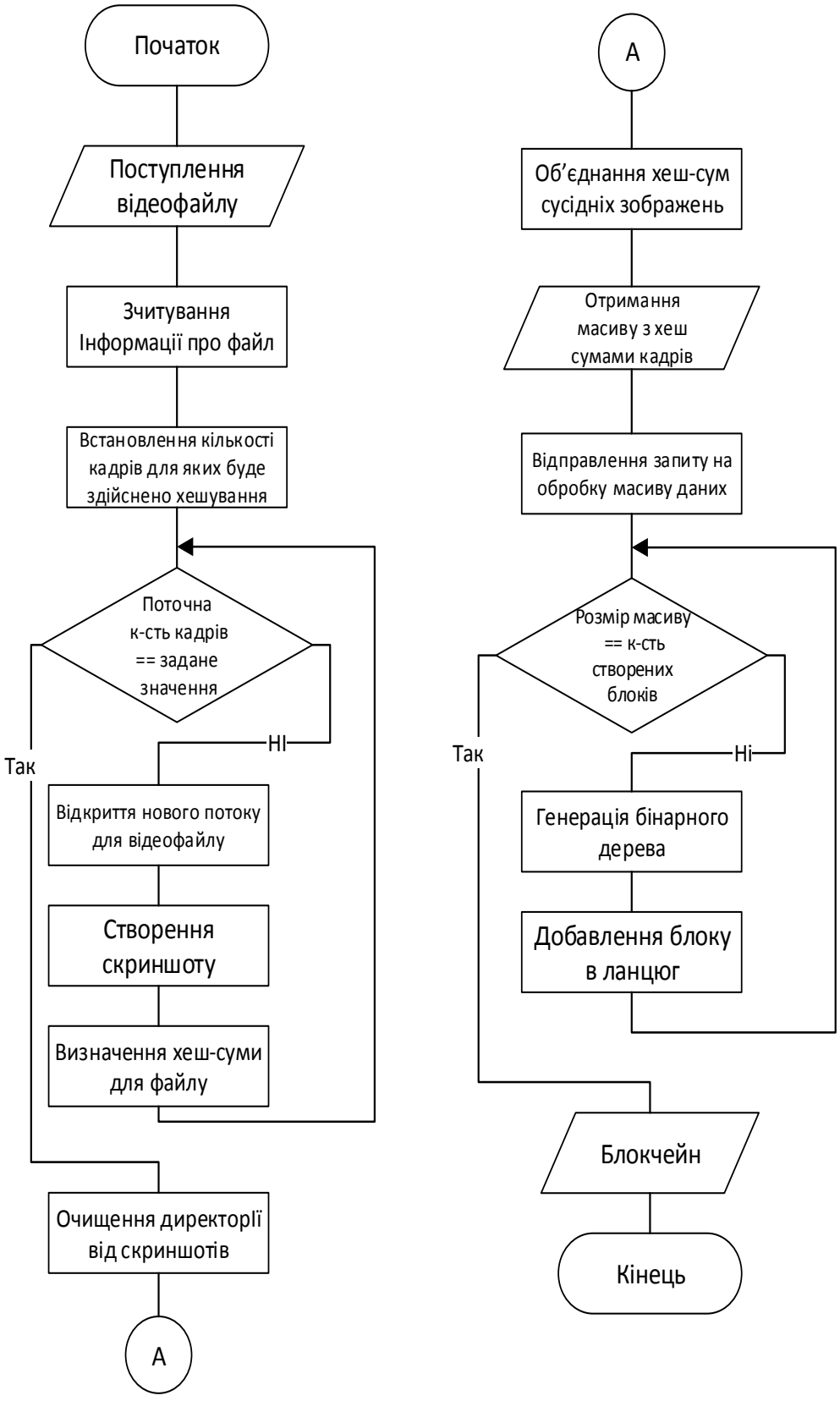

Рисунок 2.5 – Блок-схема алгоритму роботи системи

Дерево Меркле (рисунок 2.6) дозволяє отримати «відбиток» всіх транзакцій в блоці, а також ефективно верифікувати транзакції [15].

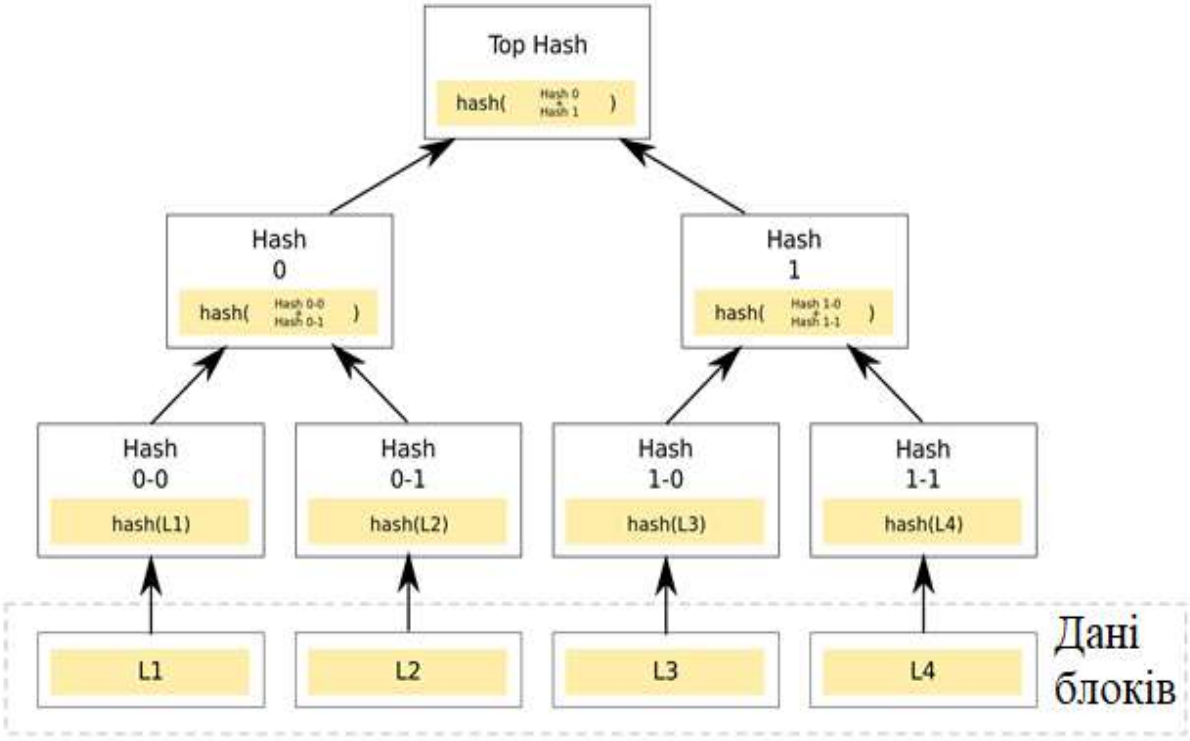

Рисунок 2.6 – Бінарне хеш-дерево

Дана блок-схема демонструє послідовність дій при виконанні алгоритму забезпечення цілісності відеофайлу не включаючи особливостей проектування під вибрану програмну платформу та не містить проміжних технічний блоків (попередній запуск веб-сервера, підключення вузлів, тощо).

3 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ СИСТЕМИ

3.1 Структура системи захисту цілісності відеоданих

Виходячи із блок-схеми алгоритму роботи системи представленому в попередньому розділі (рисунок 2.5), була спроектована та реалізована наступна структура програмної реалізації алгоритму (рисунку 3.1).

Система складається з клієнтської та серверної частини, які реалізовані на платформі Node.js, що заснована на engine V8 і забезпечує перетворення JavaScript з вузькоспеціалізованої мови в мову загального призначення[16].

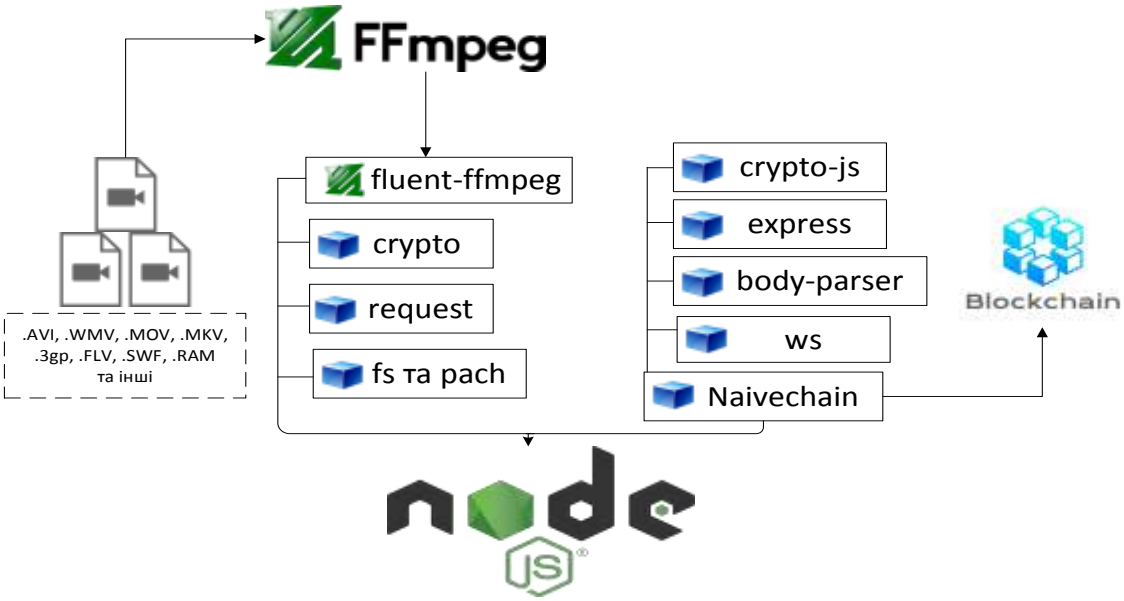

Рисунок 3.1 – Структура системи

Щоб забезпечити ефективну роботи системи, був необхідний гнучкий інструмент для обробки відеофайлів. В якості такого інструменту використано Ffmpeg — це комплекс вільних комп'ютерних програм та програмних бібліотек для маніпуляцій з цифровими відео- та аудіо-матеріалами — запис, конвертація та пакування у різні формати контейнерів [17].

Ffmpeg надає велику кількістю даних про файл та можливості для покадрового розбиття відеопотоку. Ffmpeg підтримує понад 340 різноманітних форматів файлів та функціонує на усіх поширених платформах [18].

Для кращої інтеграції в систему, взаємодія з Ffmpeg відбувається за допомогою бібліотеки "fluent-ffmgep" яка абстрагує складне використання ffmpeg з командного рядка в гнучкий та простий у використанні модуль node.js [19].

Фізичну взаємодію з виділеними за допомогою Ffmpeg кадрами забезпечують модулі "fs" та "pach", вони дозволяють задати їх розміщення та в результаті очищають директорії у потрібний момент. Із стандартних модулів node також використано "crypto", основне призначення якого обробка зашифрованих даних. Він дозволяє згнерувати хеш-функції для отриманих кадрів. В якості алгоритму хешування обрано SHA-2.

Отримані хеш-суми за допомогою бібліотеки "request" (Simplified HTTP client) передаються для подальшого опрацювання в Naivechain.

Для кожного відеофайлу, вершина дерева якого відправляється в блокчейн, формується окремий текстовий документ, заголовок якого аналогічний вершині, а вміст складається із хешів кожної з гілок. Такий файл займає мінімальний об’єм фізичного простору пам’яті, а його збереження дозволяє в разі розбіжності між збереженим хешем в блокчейні та повторною перевіркою відеофайлу на предмет розбіжностей, виявити пошкоджені(змінені) фрагменти матеріалу[20].

Для забезпечення збереження блоків із бінарним деревом кадрів відеофайлу у блокчейні використовується Naivechain, структура якого представлена на рисунку 3.2.

Naivechaine – це бібліотека, що забезпечує реалізацію блокчейна із наступними ключовими складовими [21] :

- HTTP-інтерфейс для керування вузлами;
- використання Websockets для зв’язку з іншими вузлами (P2P);
- максимально прості протоколи в P2P-комунікації;
- немає proof-of-work or proof-of-stake: блок може бути добавлений в блокчейн без конкуренції.

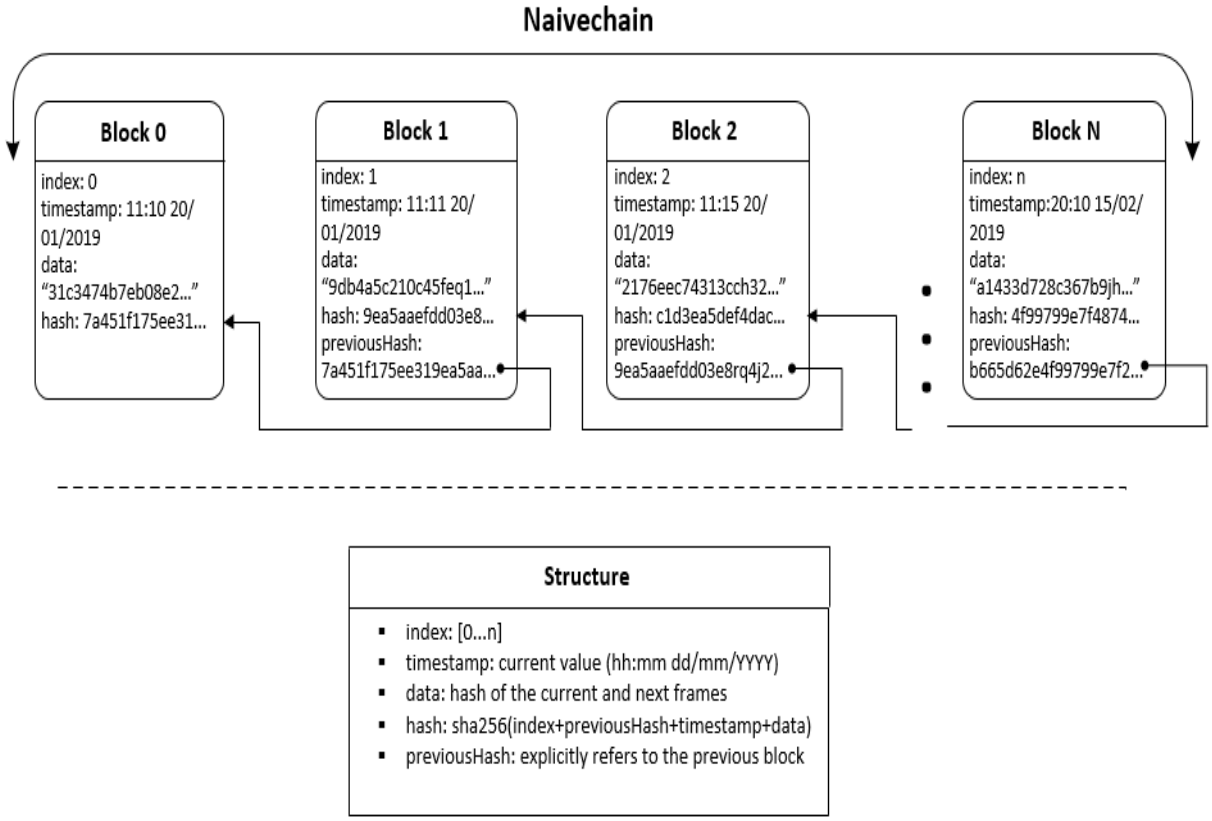

Рисунок 3.2 – Структура Naivechain

Програмним каркасом для веб-реалізації служить Express.js, отримані дані від модуля "request" обробляються за допомогою "BodyParser" та використовуються в подальшому як складова частина блоків, які формують ланцюг.

Мінімалістична структура Naivechaine компенсується ефективністю використовуваних модулів та простотою при модифікації під розроблюваний алгоритм. Сам модуль розповсюджується безкоштовно, а вихідний код доступний в репозиторії GitHub.

За замовчуванням блокчейн доступний для переглядуза адресою"/block", у відповідь на звернення, повертається файл у JSON форматі, його вигляд представлено на рисунку 3.3. Першим йде батьківський блок (Genesisblock), який розглядається як окремий випадок (Block 0 на рисунку 3.2).

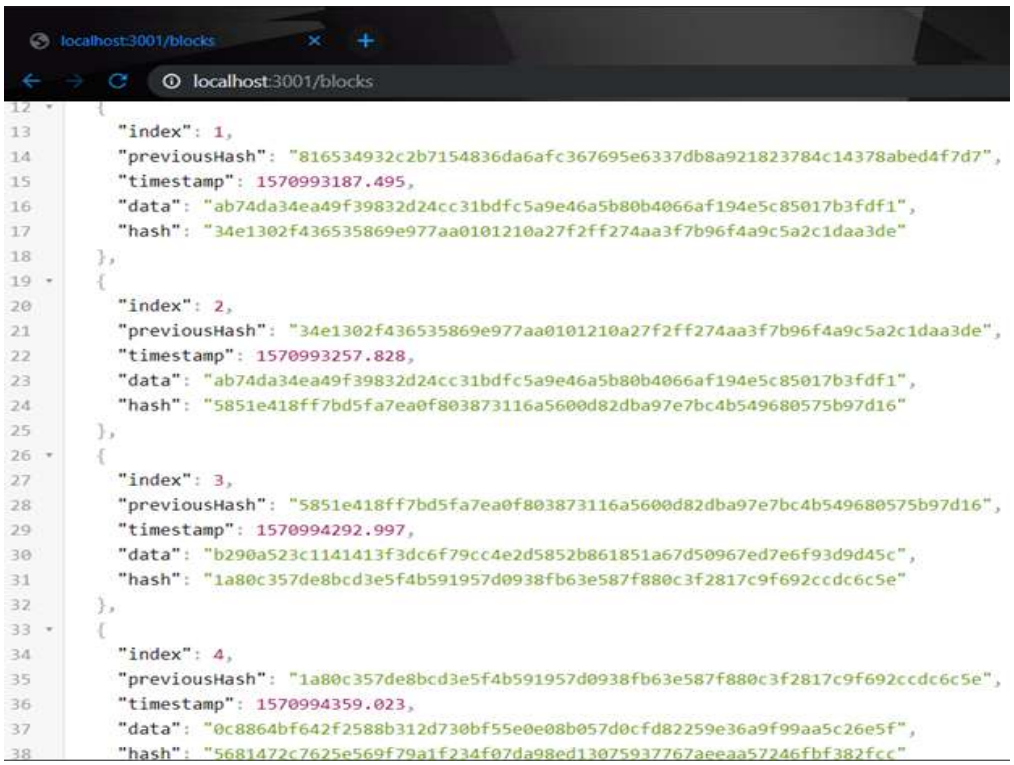

Рисунок 3.3 – Фрагмент блокчейну з вершинами дерев Меркле для відеофайлів

3.2 Алгоритм функціонування системи в реальному часі

Модифікація системи для впровадження роботи в реальному часі значно розширює можливості застосування алгоритму в контексті використання блокчейну для захисту відеоданих, та відкриває нові сфери для застосування пропонованого підходу з підвищення надійності більшого масиву категорій даних.

Якщо у випадку обробки даних з відеореєстратора чи веб-камери не виникає питань стосовно їх достовірності, то у випадку передачі потокового сигналу, наприклад, з віддаленої IP-камери неможливо гарантувати достовірність та повноту отриманих даних.

Збереження в блокчейні потокових відеоданих з вихідного джерела має ряд недоліків, серед яких необхідно виділити наступні:

- неможливість гарантувати отримання всіх кадрів без пошкоджень;
- відсутність контролю за достовірністю оригіналу;

- відсутність чіткої стандартизації для організації універсального програмного рішення;

- складність в забезпеченні синхронності даних.

Один із варіантів вирішення цих спірних моментів – забезпечувати цілісність відеоданих на етапі поступлення кадрів в систему. Крім вирішення вище згаданих проблем, даний підхід дозволяє без обмежень модифікувати дані в залежності від поточних цілей системи та з врахуванням можливостей апаратної складової.

Вдосконалена структура системи з внесеними змінами представлена на рисунку 3.4.

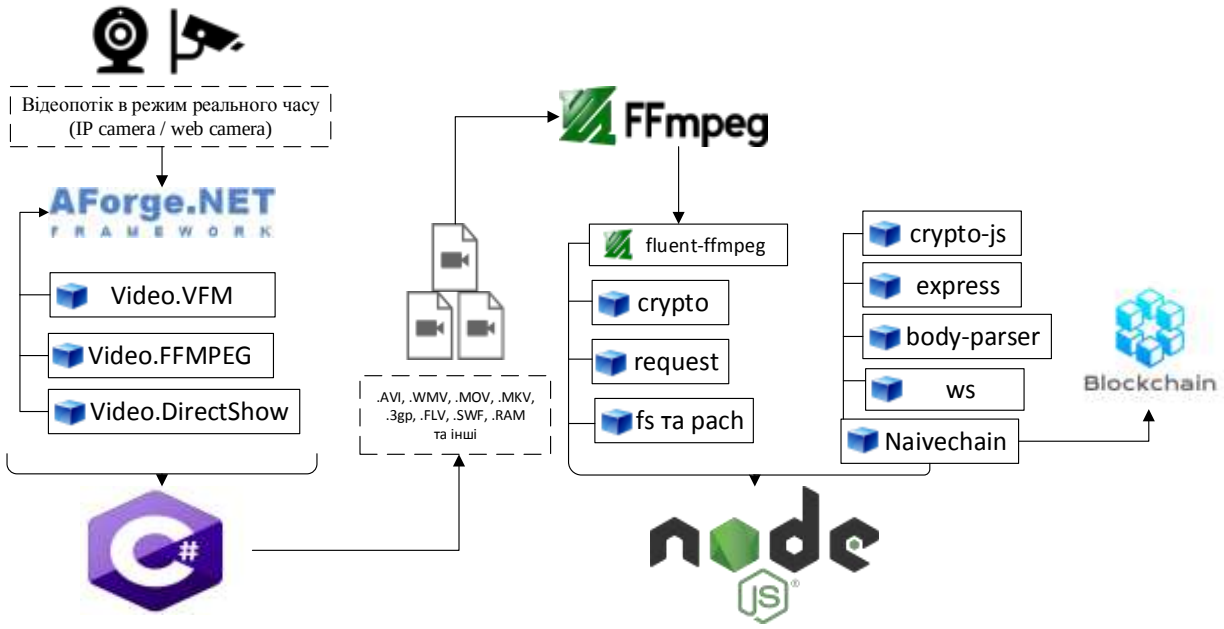

Рисунок 3.4 – Структура системи з модулем для роботи в реальному часі

Специфікація Node.js не дає змоги повноцінно забезпечити покадрову обробку пакетів даних, тому доцільним є інтегрування спорідненого модуля [22].

AForge.Net – це програмна платформа з відкритим кодом на C#, створена для розробників та дослідників у галузі комп'ютерного зору та штучного інтелекту – обробка зображень, нейронні мережі, генетичні алгоритми, нечітка логіка, машинне навчання, тощо [23].

Блок-схема алгоритму роботи в режимі реальному часі представлена на рисунку 3.5, а при завершенні функціонування, дані переходять під управління node.js, алгоритм роботи якої наведений у попередньому розділі (рисунок 2.5).

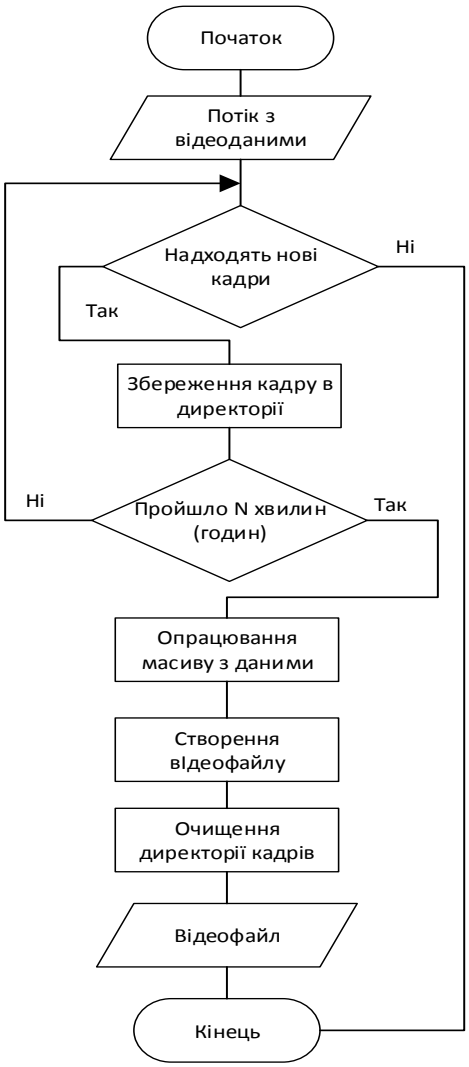

Рисунок 3.5 – Блок-схема алгоритму роботи системи в режимі реального часу

При запуску запису з веб-камери або підключенні до IP-камери розпочинається процес покадрового збереження вхідних даних в Bitmap об’єктах, які перетворюють ці кадри в зображення формату .png на фізичному носії.

Інтерфейс користувача для можливості вибору джерела трансляції або зміни протоколу розроблений в WindowsForms та WPF, приклади його реалізації представлені на рисунку 3.6.

З певною періодичністю, паралельно до процесу збереження вхідних кадрів, підключається бібліотека VFM, завдання якої зібрати масив зображень для створення відеофайлу. Вхідними параметрами для нього служать [23]:

- шлях збереження;
- роздільна здатність (яка не може перевищувати значень початкових кадрів);
- частота кадрів (framerate);
- алгоритм стиснення цифрового відеопотоку (відеокодек);
- бітрейт (video BitRate).

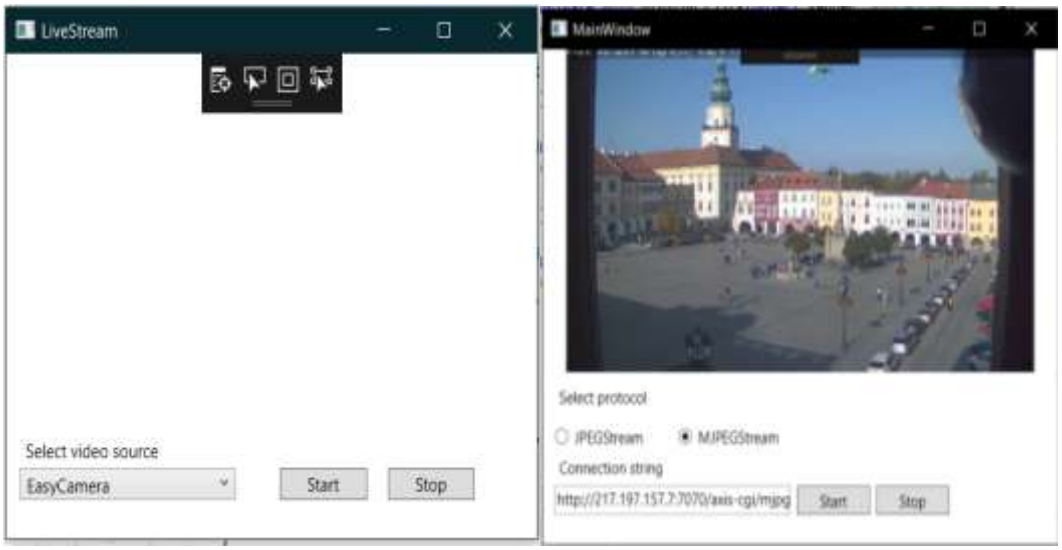

Рисунок 3.6 – Приклади інтерфейсу користувача

Після формування відеофайлу, директорія очищається від використовуваних в процесі створення матеріалу зображень.

Процес створення відеофайлу можна проводити на окремому апаратному забезпеченні та масштабувати якість кінцевого матеріалу в залежності від можливостей та потреб системи.

В результаті отримується готовий відеофайл, повний цикл створення якого прозорий та контролюється, що надає можливості для подальшої обробки згідно із алгоритмом представленим на рисунку 2.5.

3.3 Дослідження забезпечення цілісності відеофайлів

Від характеристик апаратної складової залежить стабільність та швидкодія розробленої системи. Найбільше навантаження відбувається в процесі розбиття відеопотоку на окремі кадри. Від оптимального вибору кількості кадрів залежить швидкодія системи в цілому та надійність використовуваного підходу для захисту відеоданих.

Наступні дослідження були проведені з використанням CPUIntel(R) Core(TM) i7-7700HQ, відповідно при зміні процесора часові характеристики будуть відрізнятись.

В таблиці 3.1 представлені результати дослідження затрат часу відносно кількості кадрів для яких генерується хеш-сума. Для дослідження був використаний відеофайл з розширенням 720p та тривалістю 58:15[15].

Таблиця 3.1 – Дослідження затрат часу на створення скриншотів

	кількість кадрів	час (секунд)
1 кадр кожні 10с	352	99,418
1кадр кожні 5с	703	200,284
1 кадр в секунду	3515	975,39
5 кадрів в секунду	17575	4945,258

Побудувавши на основі таблиці діаграму (рисунок 3.7), можна побачити її прямолінійність, виходячи з отриманих даних, випливає, що в середньому за 1 секунду створюється 3.55 кадра. Звідси можна визначити часові затрати на обробку аналогічно файлу.

Наприклад, обробка відеофайлу на частоті 25кадрів/секунда займе 87875/3.5=25107 секунд або 6.97годин для відеофайлу тривалістю 58:15 у якості 720p [20].

Рисунок 3.7 – Діаграма затрат часу (в секундах) на створення скриншотів

В таблиці 3.2 представлено часові затрати на генерацію кадрів з частотою 1 в секунду для відеофайлів з розширеннями: 720p, 1080p та UHD. Рисунок 3.8, демонструє порівняння кількості згенерованих скриншотів для різних розширень за одиницю часу.

Таблиця 3.2 Затрати часу для відеофайлу із різним розширенням

Якість	час (секунд)
720p (1280×720)	975,39
1080p (1920×1080)	1710,43
UHD (3840 × 2160)	3216,77

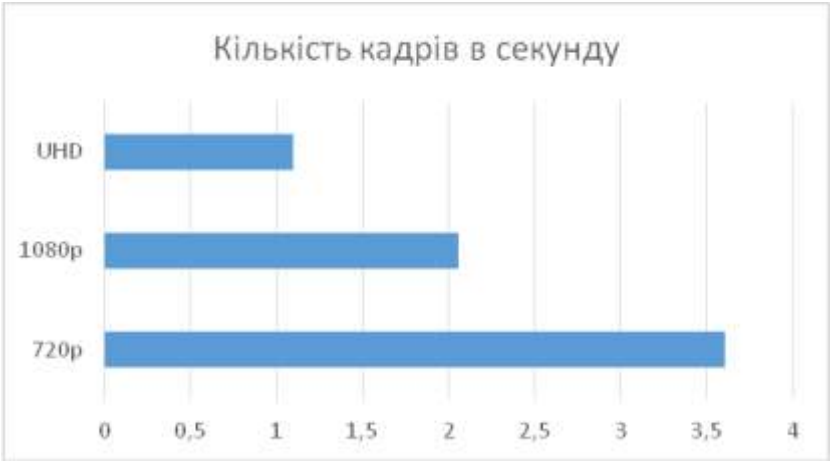

Рисунок 3.8 – Діаграма порівняння генерації скриншотів за одиницю часу

В дослідженнях представлених нижче використовуються наступні статичні параметри для створюваних відеофайлів: відеокодек – H263P, бітрейт – 1 000 000.

Для дослідження кількісних параметрів роботи модуля в реальному часі було обрано одну із віддалених IP-камер, яка представлена на веб-сервісі Opentopia. Камера виробництва Axis, а формат передачі даних – MJPEG.

На вхід системи почергово направлявся потік кадрів із камери протягом 5 хвилин. Для кожного наступного етапу відбувалось зменшення розширення кардів, отримані результати представлені в таблиці 3.3.

Виходячи із діаграми (для таблиці 3.3) на рисунку 3.9, прослідковується закономірність в збільшенні кількості кадрів, що надходять від камери при зменшенні розширення вхідних кадрів.

Таблиця 3.3 – Продуктивність обробки даних з IP-камери

Розширення	Кількість кадрів	Розмір кадрів (MB)	Час створення відеофайлу (секунд)	Розмір файлу (MB)
1280x720	2184	2382,5	49,2	16,9
800x600	2531	1564,2	35,7	19,6
640x480	4254	1677,6	40,5	32,3
320x200	7083	732,5	21,4	23,8

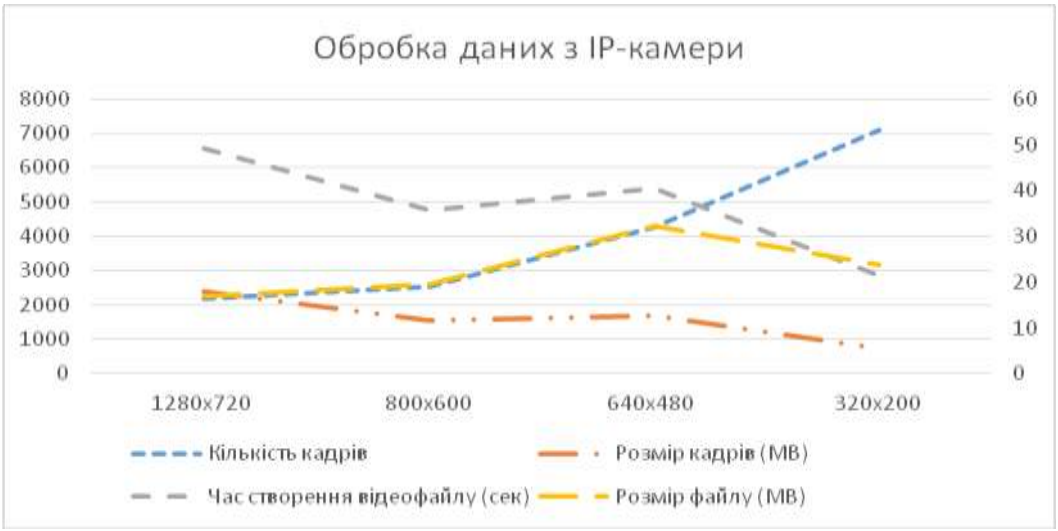

Рисунок 3.9 – Обробка даних з IP-камери при різних розширеннях зображення

Відповідно, при зменшенні розширення кадру, пропорційно зменшується загальний об’єм пам’яті які вони займають та час необхідний для їх перетворення в відефайл, але сам розмір файлу зростає, оскільки збільшується густина кадрів на одиницю часу.

Діаграма представлена на рисунку 3.10 представляє показник FPS для даного дослідження.

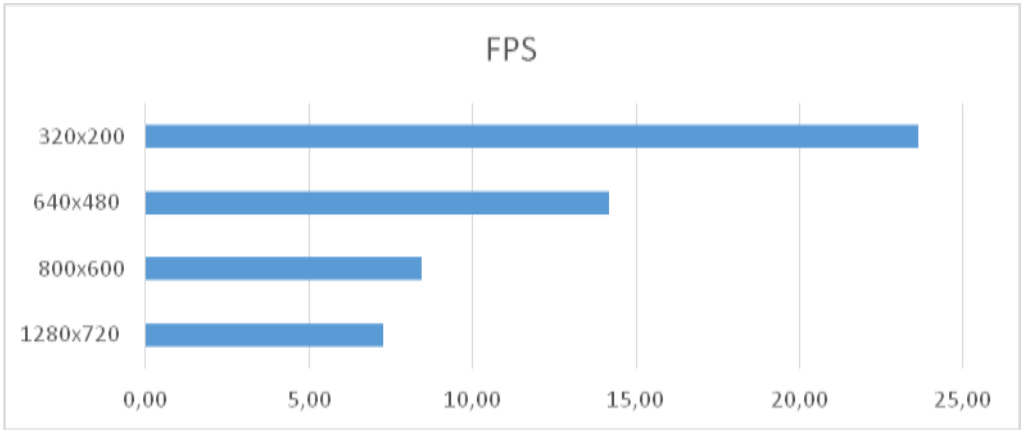

Рисунок 3.10 – Кількість кадрів в секунду для IP-камери

Для наступного дослідження була використана вебкамера з роздільною здатністю 1280x720 та частотою кадрів за секунду – 16.

В експерименті використовувався один відеоряд, в якому відбувалось зменшення FPS для визначення різниці в затратах часу на створення відеофайлу.

Результат дослідження представлений в таблиці 3.4.

Таблиця 3.4 – Затрати часу та пам'яті при змінних значеннях FPS

FPS	Час (секунд)	Розмір файлу (MB)
16	117	36,6
10	70	36,7
5	36	37
1	7	18

Дослідження демонструє зменшення затрат часу на створення відеофайлу при зменшення кількості кардів та незначні зміни у їх розмірі при тривалості відеоматеріалу в 5 хвилин.

Діаграма порівняння затрат часу при створенні відеофайлу з заданням різної частоти кадрів представлена на рисунку 3.11.

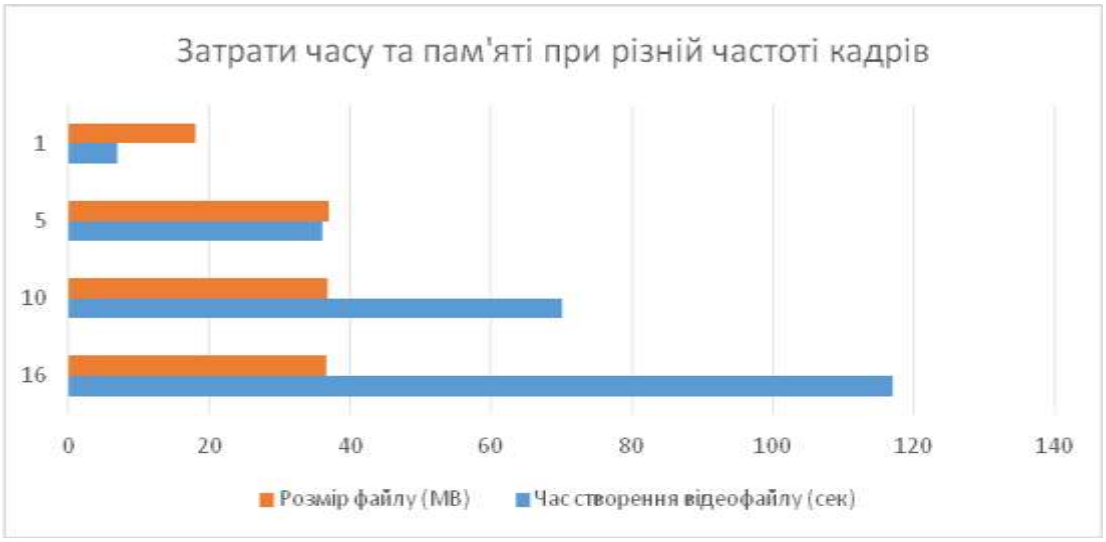

Рисунок 3.11 – Діаграма затрат часу та пам’яті на відеофайл при різному FPS

На основі проведених вище досліджень, можна оцінити ефективність роботи алгоритму, для якого було визначено часові затрати при різних типах операцій та об'єм використовуваної пам'яті для найбільш поширених стандартів якості зображення, що дає змогу масштабувати розгортання і використання системи в залежності від доступних ресурсів та потреб.

ВИСНОВКИ

В роботіпроаналізовано особливості технології блокчейн, її переваги та недоліки, розглянуто галузі використання та реалізації. Проаналізовано існуючі систем із схожим алгоритмом функціонування. Отримані дані в подальшому враховані при реалізації наступних етапів та лежать в основі алгоритмічного забезпечення системи.

Досліджено алгоритм роботи блокчейну, визначено принцип функціонування, використовувані технології та їх взаємозв'язок. Проведено аналіз криптографічних хеш-функцій та алгоритмів їх діяльності, визначено вимоги та властивості необхідні для найбільш оптимального хешування.Розроблено модель функціонування системи, яка представлена покроковим алгоритмом та блок-схемою.

Представлено програмну реалізацію алгоритму із детальним описом структури системи та розроблено модуль роботи в режимі реального часу, що забезпечує розширення можливостей системи та застосування алгоритму.

Проведено ряд досліджень на основних стандартах якості зображення при різних типах вхідного відеосигналу, що підтверджують ефективність розробленого алгоритму та дозволяють проводити масштабування апаратної складової системи в залежності до поточних потреб та можливостей у відповідності до результатів досліджень.

Розроблено програмну систему на основі алгоритму захисту цілісності відеоданих з використанням технології блокчейн, що дало можливість вирішити проблему довіри при передачі відеоданих та забезпечує змогу перевірки цілісності матеріалу в будь-який момент часу.

Архітектура розробленої системи забезпечує функціонал для гнучкого розподілення апаратних ресурсів між модулями, а можливість роботи в режимі реального часу відкриває доступ для забезпечення достовірності даних з веб-камер, IP-камер, відеореєстраторів та інших джерел, які забезпечують потокову передачу покадрового відеосигналу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Арефьева А. С., Гогохия Г. Г. Перспективы внедрения технологии блокчейн // Молодой ученый. — 2017. — №15. — С. 326-330.

2. О.Н. Жданов. М. Методика выбора ключевой информации для алгоритма блочного шифрования:Монография//НИЦ ИНФРА-М., 2013. – 88 с.

3. Блокчейн технология [Електронний ресурс]. – Режим доступу: https://ru.bitcoinwiki.org/wiki/Блокчейн/.

4. Объяснение блокчейна для веб-разработчиков [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/323128/.

5. Цифрове відео [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Цифрове_відео/.

6. What is fps (frames per second)? [Electronic resource]. – Mode of access: https://whatis.techtarget.com/definition/fps-frames-per-second.

7. Форматы видеофайлов [Електронний ресурс]. – Режим доступу: https://spravochnick.ru/informatika/kodirovanie_informacii/formaty_videofaylov/.

8. Trusted Timestamping with OriginStamp [Electronic resource]. – Mode of access: https://originstamp.org/home.

9. Emernotar – защита документов с помощью блокчейна [Електронний ресурс]. – Режим доступу: https://emercoin.com/ru/news/emernotar-zashchita-dokumentov-s-pomoshchyu-blokcheyna.

10. Обзор Acronis True Image 2017 New Generation [Електронний ресурс]. – Режим доступу: https://www.comss.ru/page.php?id=3971.

11. Acronis Notary [Electronic resource]. – Mode of access: https://notary.acronis.com/.

12. Andreas M. Antonopoulos. Mastering Bitcoin. Programing the open blockhain. 2nd Edition. // O'RELLY. – 2017. – 405 c.

13. Алгоритмы хеширования - простое объяснение сложного [Електронний ресурс]. – Режим доступу: https://vc.ru/crypto/47132-algoritmy-heshirovaniya-prostoe-obyasnenie-slozhnogo.

14. Cryptographic hash function [Electronic resource]. – Mode of access: https://en.wikipedia.org/wiki/Cryptographic_hash_function/.

15. Merkle tree [Electronic resource]. – Mode of access: https://en.wikipedia.org/wiki/Merkle_tree/.

16. YatskivV., Yatskiv N.,Bandrivskyi O.Proof of Video Integrity Based on Blockchain. International Conference on Andvanced Computer Information technologies, ACIT`2019, June 5-9, 2019., Ceske Budejovice, Czech Republic, 2019. – P. 431-434

17. FFmpeg [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/FFmpeg.

18. About FFmpeg [Electronic resource]. – Mode of access: https://ffmpeg.org/about.html.

19. Fluent ffmpeg-API for node.js [Electronic resource]. – Mode of access: https://github.com/fluent-ffmpeg/node-fluent-ffmpeg.

20. Яцків В.В., Яцків Н.Г., Бандрівський О.Є. Захист цілісності відеофайлів на основі технології блокчейн. Міжнародна науково-практична конференція Фізико-технічні проблеми передавання, оброблення та зберігання інформації в інфокомунікаційних системах. 8-10 листопада 2018 р., Чернівці, Україна, 2018. –С.167

21. Naivechain - a blockchain implementation in 200 lines of code [Electronic resource]. – Mode of access: https://github.com/lhartikk/naivechain.

22. ASP.NETCorevsNode.JS [Електронний ресурс]. – Режим доступу: https://itvdn.com/ru/blog/article/aspnet-vs-nodejs.

23. AForge.NET open source framework [Electronic resource]. – Mode of access: https://www.codeproject.com/Articles/16859/AForge-NET-open-source-framework.

Додаток А

Лістинг програмного коду

1. Програмний модуль для обробки зображень та надсилання даних в блокчейн:

```
var crypto = require("crypto");
const fs = require("fs");
const path = require("path");
const ffmpeg = require("fluent-ffmpeg");
var request = require('request');
let StartTime,EndTime;
let videoController=function(file){
   function fileInfo() {
      ffmpeg.ffprobe(file,function(err, metadata) {
         var audioCodec = null;
         var videoCodec = null;
         var fileName = null;
         var fileW = null;
         var fileH = null;
         var createD= null;
         metadata.streams.forEach(function(stream){
            if (stream.codec_type === "video"){
               videoCodec = stream.codec_name;
               fileW=stream.width;
               fileH=stream.height;
            } else if (stream.codec_type === "audio"){
               audioCodec = stream.codec_name;
            }
         });
         fileName=metadata.format.filename;
         createD=metadata.format.tags.CREATION_TIME;
         console.log("-Video codec: %s\n-Audio codec: %s", videoCodec, audioCodec);
         console.log("-Width: %s Height: %s",fileW,fileH);
         console.log("-File Name:", fileName);
```

```
        console.log("-Create Date:", createD);
        });
 }
console.log('*Модуль роботи з відеофайлами підключено');
console.log('==============================');
let hashS='';
let hashList = [];
let hashSumList = [];
const count = 120;
const timestamps = [];
const startPositionPercent = 1;
const endPositionPercent = 99;
const addPercent = (endPositionPercent - startPositionPercent) / (count-1);
let i = 0;
if (!timestamps.length) {
    let i = 0;
    while (i < count) {
        timestamps.push(`${startPositionPercent + addPercent * i}%`);
        i = i + 1;
    }
}
function createHashSum() {
    console.log('Лист хешів:');
    hashList.forEach(function(item,i,hashList){
        console.log('i=',i,'  Element=',item);
        })
    for(i=1;i<count-1;i++){
        hash256 = crypto.createHash("sha256");
        hash256.update(hashList[i]+hashList[i+1]);
        hashSumList[i]=hash256.digest('hex');
    }
     console.log('Суми хеш-функцій для кардів:');
    hashSumList.forEach(function(item,i,hashSumList){
     //   console.log('i=',i,'  Element=',item);
```

```
        })
        console.log('================================');
        console.log('*Розпочато процес видалення скриншотів');
        delAllFromFolder();
  addBlocks();
    }
    function addBlocks(){
        var url = " http://localhost:3001/mineBlock";
        hashSumList.forEach(function(item,i,hashSumList){
      request.post(
    url,
    { json: { 'data': item} },
    function (error, response, body) {
        if (!error && response.statusCode == 200) {
        }    }
);          })    }
    function delAllFromFolder(){
    const directory = './myvideo/screens';
    fs.readdir(directory, (err, files) => {
      if (err) throw err;
      for (const file of files) {
        fs.unlink(path.join(directory, file), err => {
          if (err) throw err;
        });
      }
    });
        console.log('-Усі файли видалено');
        console.log('================================');
        console.log('*Модуль взаємодії з відеофайлами завершив роботу');
        console.log('================================');
        console.log('Початок:-',StartTime,' Кінець:- ',EndTime);
    }
    function createHash(i){
          var s = fs.ReadStream('./myvideo/screens/myfile'+i+'.png');
```

```
        var hash256 = crypto.createHash("sha256");
        s.on('data', function(d) {
        hash256.update(d);
        });
        s.on('end', function() {
        var generated_hash = hash256.digest('hex');
        console.log( 'Згенеровано хеш-функцію для зображення #'+i+': '+generated_hash);
        //hashS+=generated_hash;
        hashList[i]=generated_hash;
        });
    }
    function takeScreenshots(file) {
        ffmpeg(file)
        .on("start", () => {
            if (i < 1) {
                console.log(`*Розпочато процес створення скриншотів та визначення їх хеш-функцій (SHA256)`);
                console.log('==============================');
                console.log('Час початку:');
                StartTime=new Date().getTime();
            }
        })
        .on("end", () => {
            i = i + 1;
            if (i < count)
            {
                console.log(`Створено скриншот: ${i}`);
                createHash(i);
                takeScreenshots(file);
            } else {
                console.log('Час кінця:');
                EndTime=new Date().getTime();
                console.log('*Усі скриншоти успішно створено');
            }
```

```
        })
        .screenshots({
            count: 1,
            timemarks: [timestamps[i]],
            filename: `%b-${i + 1}.jpg`
        },path.join(path.dirname(file), 'screens'));
   }
 function hashImms()
 {
 while(i<count){
   setTimeout(createHash, 2000,i);
   i++;
}  }
  hashImms();
};
videoController();
```

2. Програмний модуль для работи в режимі реального часу:

```
namespace AForge.Wpf
{
   /// <summary>
   /// Interaction logic for MainWindow.xaml
   /// </summary>
   public partial class MainWindow : Window, INotifyPropertyChanged
   {
      #region Public properties
      public ObservableCollection<FilterInfo> VideoDevices { get; set; }
      public FilterInfo CurrentDevice
      {
         get { return _currentDevice; }
         set { _currentDevice = value; this.OnPropertyChanged("CurrentDevice"); }
      }
      private FilterInfo _currentDevice;
      #endregion
```

```
#region Private fields
private IVideoSource _videoSource;
#endregion
public MainWindow()
{
   InitializeComponent();
   this.DataContext = this;
   GetVideoDevices();
   this.Closing += MainWindow_Closing;
}
private void MainWindow_Closing(object sender, System.ComponentModel.CancelEventArgs
e)
{
   StopCamera();
}

private void btnStart_Click(object sender, RoutedEventArgs e)
{
   StartCamera();
}
int i = 0;
private void video_NewFrame(object sender, Video.NewFrameEventArgs eventArgs)
{
   try
   {
      BitmapImage bi;
      using (var bitmap = (Bitmap)eventArgs.Frame.Clone())
      {
         bi = bitmap.ToBitmapImage(i);
         i++;
      }
      bi.Freeze();
      Dispatcher.BeginInvoke(new ThreadStart(delegate { videoPlayer.Source = bi; }));
   }
```

```
            catch (Exception exc)
            {
                MessageBox.Show("Error on _videoSource_NewFrame:\n" + exc.Message, "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
                StopCamera();
            }
        }
        private void btnStop_Click(object sender, RoutedEventArgs e)
        {
            StopCamera();
        }
    private void GetVideoDevices()
        {
            VideoDevices = new ObservableCollection<FilterInfo>();
            foreach (FilterInfo filterInfo in new FilterInfoCollection(FilterCategory.VideoInputDevice))
            {
                VideoDevices.Add(filterInfo);
            }
            if (VideoDevices.Any())
            {
                CurrentDevice = VideoDevices[0];
            }
            else
            {
                MessageBox.Show("No video sources found", "Error", MessageBoxButton.OK,
MessageBoxImage.Error);
            }
        }
        private void StartCamera()
        {
            if (CurrentDevice != null)
            {
                _videoSource = new VideoCaptureDevice(CurrentDevice.MonikerString);
                _videoSource.NewFrame += video_NewFrame;
```

```
    _videoSource.Start();
    }
}
private void StopCamera()
{
    if (_videoSource != null && _videoSource.IsRunning)
    {
        _videoSource.SignalToStop();
        _videoSource.NewFrame -= new NewFrameEventHandler(video_NewFrame);
    }
    try
    {
        using (VideoFileWriter writer = new VideoFileWriter())
        {
            writer.Open(@"c:\test\myfile.avi", 1280, 720, 16, VideoCodec.H263P, 1000000);
            for(int count =0; count < i; count++)
            {
                var file =@"\frames\myfile"+count+".png";
                try
                {
                    using (Bitmap image = Bitmap.FromFile(file) as Bitmap)
                    {
                        if (image != null)
                        {
                            writer.WriteVideoFrame(image);
                      }
                    }
                }
                catch
                { continue; }
            }
            writer.Close();
            Process.Start("cmd.exe", "/K " + @"npm start --prefix C:\videochain\");
        }     }
```

```
            catch (Exception exc)
            {
                MessageBox.Show("Error on _videoSource_NewFrame:\n" + exc.Message, "Error",
MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
        #region INotifyPropertyChanged members
        public event PropertyChangedEventHandler PropertyChanged;
        protected void OnPropertyChanged(string propertyName)
        {
            PropertyChangedEventHandler handler = this.PropertyChanged;
            if (handler != null)
            {
                var e = new PropertyChangedEventArgs(propertyName);
                handler(this, e);
            }
        }
    }
}
static class BitmapHelpers
    {
        public static BitmapImage ToBitmapImage(this Bitmap bitmap,int i)
        {
            BitmapImage bi = new BitmapImage();
            bi.BeginInit();
            MemoryStream ms = new MemoryStream();
            bitmap.Save(ms, ImageFormat.Bmp);
            bitmap.Save("frames/myfile"+i+".png", ImageFormat.Png);
            ms.Seek(0, SeekOrigin.Begin);
            bi.StreamSource = ms;
            bi.EndInit();
        return bi;
        }
    }
```

Додаток Б

Копії публікацій

Вилучено оргкомітетом для дотримання анонімності рецензування роботи

Додаток В

Довідки про впровадження

Вилучено оргкомітетом для дотримання анонімності рецензування роботи