КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (33), 2026
ISSN 2663 – 4023
DOI 10.28925/2663-4023.2026.33.1159
УДК 004.056:004.02

Долгова Наталя Геннадіївна
к.т.н., доцент,
доцент кафедри кібербезпеки та інформаційних технологій,
Харківський національний економічний університет імені Семена Кузнеця, Харків, Україна
ORCID: 0000-0002-8950-8200
natalya.dolgova@hneu.net

МОДЕЛЬ СИСТЕМИ ДОКУМЕНТООБІГУ З КРИПТОГРАФІЧНО ВЕРИФІКОВАНИМ ЛАНЦЮГОМ СТАНІВ ДОКУМЕНТА

Анотація. У статті розроблено модель системи документообігу, орієнтовану на середовища, у яких критично важливими є цілісність історії документа, контроль його життєвого циклу та можливість незалежної перевірки виконаних операцій. Актуальність дослідження зумовлена тим, що традиційні системи електронного документообігу переважно спираються на централізовані журнали подій і прикладну логіку, що не усуває ризиків ретроактивної модифікації історії документа та зниження її доказової цінності. Метою роботи є побудова моделі системи документообігу, у якій цілісність історії документа забезпечується за рахунок криптографічно верифікованого ланцюга його станів і фіксації доказових атрибутів подій у permissioned DLT. Запропонована модель поєднує архітектурний і формальний рівні подання системи. На архітектурному рівні виокремлено користувацький, прикладний, доказовий і контентний рівні. На формальному рівні документ подано як послідовність криптографічно пов'язаних станів, у якій кожен новий стан містить хеш стану, метадані, часову мітку, хеш вмісту та посилання на попередній стан, що забезпечує цілісність і простежуваність усієї історії документа. Для реалізації доказового рівня використано смартконтракт реєстрації станів документа та permissioned DLT на базі Hyperledger Besu. Експериментальну валідацію моделі виконано на локальному стенді з використанням консенсусу QBFT, зовнішнього сховища та програмних модулів формування і повної верифікації історії документа. Результати експериментів підтвердили здатність моделі виявляти ретроактивні зміни вмісту, метаданих, підписів і часових атрибутів, локалізувати перший порушений стан та забезпечувати близьке до лінійного зростання часу повної верифікації зі збільшенням довжини ланцюга станів. Порівняльне оцінювання із централізованим журналюванням показало перевагу запропонованого підходу за критеріями виявлення втручання, локалізації порушення та незалежної перевірюваності результату. Практичне значення роботи полягає у можливості використання запропонованої моделі як основи для побудови корпоративних систем документообігу.

Ключові слова: захищений документообіг; блокчейн; смарт-контракт; permissioned DLT; Hyperledger Besu; ланцюг станів документа; криптографічна верифікація; верифікація цілісності документа.

ВСТУП

Документи обмеженого доступу використовуються в державних, корпоративних та інфраструктурних інформаційних системах, де критично важливими є цілісність змісту, контрольованість життєвого циклу та можливість подальшої перевірки всіх операцій обробки. За таких умов особливого значення набувають механізми суворого розмежування доступу, фіксації історії змін, юридично значущого підписання та незалежної перевірки дій користувачів і системних сервісів. У більшості сучасних систем електронного документообігу цілісність історії обробки забезпечується на рівні прикладної логіки, журналів подій і адміністративного контролю. Проте такий підхід передбачає високий рівень довіри до оператора системи та не виключає ризиків непомітної модифікації журналів, ретроспективної зміни версій документів і зниження доказової сили історії їх обробки.

Актуальність дослідження зумовлена потребою у розробленні моделей захищеного документообігу, в яких безпека забезпечується не лише шифруванням і розмежуванням доступу, а й

криптографічно перевірюваною структурою історії станів документа. Попри розвиток блокчейн-орієнтованих підходів, механізмів незмінюваного аудиту та розподіленої фіксації подій, наявні рішення здебільшого охоплюють окремі аспекти захисту, верифікації або зберігання даних. Тому науковий і практичний інтерес становить побудова моделі системи документообігу, здатної поєднати модульну архітектуру, рівневу організацію функцій безпеки, керований життєвий цикл документа та доказову цілісність історії його обробки.

Постановка проблеми. Для середовищ з обмеженим доступом актуальною залишається задача побудови цілісної моделі захищеного документообігу, яка поєднувала б конфіденційність, розмежування доступу, керування життєвим циклом документа та доказову цілісність історії його обробки. Існуючі рішення переважно охоплюють окремі аспекти цієї задачі: захист вмісту документа, фіксацію подій обробки або забезпечення простежуваності на рівні журналів і транзакцій. При цьому узгоджене представлення документа, його метаданих, підписів, часових атрибутів і зв'язків між послідовними станами в межах єдиної перевірюваної моделі залишається недостатньо опрацьованим.

У зв'язку з цим актуальною є задача розроблення моделі захищеної системи документообігу, яка забезпечувала б не лише конфіденційність і розмежування доступу, а й доказову цілісність усієї історії життєвого циклу документа. Така модель має підтримувати створення, погодження, підписання, зберігання, архівування, а також зміну, обмеження й відкликання прав доступу зі збереженням криптографічно перевірюваного зв'язку між станами документа. Розв'язання цієї задачі потребує поєднання модульної архітектури системи, рівневої організації функцій безпеки, зовнішнього захищеного зберігання контенту та розподіленої фіксації доказових даних.

Аналіз останніх досліджень і публікацій. У дослідженнях, присвячених цифровому документообігу, дедалі помітнішим є перехід від традиційних централізованих систем до розподілених моделей, у межах яких безпека документа розглядається не лише як задача зберігання, а і як задача доказовості операцій упродовж усього його життєвого циклу. Одним із ранніх прикладів такого підходу є робота [1], у якій запропоновано децентралізовану систему керування документами на основі blockchain і secret sharing. Автори зосереджуються на розподіленому зберіганні та контролі доступу, показуючи, що відмова від єдиної довіреної точки зберігання підвищує стійкість документної системи до компрометації та несанкціонованого втручання.

У дослідженні [2] запропоновано функціональну структуру системи кіберзахисту критичної інфраструктури, засновану на модульній організації та багаторівневій архітектурі. Розглядаються елементи функціональної структури СКЗ і взаємозв'язки між модулями та рівнями системи, що дає змогу формалізувати розподіл функцій безпеки та підвищити керованість архітектури. Запропонований підхід демонструє ефективність структуризації складних систем через виокремлення функціональних модулів і архітектурних рівнів. Отримані результати можуть бути використані під час розроблення моделей захищених систем електронного документообігу, де модульна організація та рівневий розподіл функцій сприяють інтеграції механізмів контролю доступу, криптографічного захисту й аудиту операцій.

Окрім досліджень, що описують конкретні архітектурні рішення, окремий напрям становлять роботи, які узагальнюють підходи до проєктування blockchain-орієнтованих document management systems. Авторами [3] виконано систематичний огляд досліджень, присвячених проєктуванню blockchain-орієнтованих систем керування документами та реалізації paperless-сценаріїв. У роботі проаналізовано архітектурні підходи, моделі зберігання даних і механізми забезпечення цілісності документів у розподілених системах. Результати аналізу показують, що для paperless-рішень найбільш характерною є гібридна архітектура, яка передбачає розділення шарів зберігання та доказовості: вміст документа розміщується у зовнішньому сховищі (off-chain), тоді як у блокчейні (on-chain) фіксуються криптографічні хеші, посилання та події життєвого циклу. Цей висновок має важливе значення для розроблення моделі системи документообігу, оскільки підтверджує доцільність архітектурного підходу, за якого блокчейн використовується не як основне сховище документів, а як доказовий шар або рівень, що забезпечує незмінність і перевірюваність історії обробки.

Практична реалізація такого підходу в прикладному документообігу простежується в роботі [4], присвяченій використанню blockchain і OCR в електронному урядуванні на прикладі оброблення вхідних рахунків. Автори демонструють поєднання автоматичного вилучення даних із документа, фіксації ключових реквізитів у блокчейні та подальшого супроводження документа в процесі оброблення. Тим самим підтверджується, що blockchain може бути інтегрований у реальний document workflow не лише як засіб реєстрації факту, а і як механізм підвищення простежуваності операцій з документами.

Розвиток цього підходу стосовно юридично значущих документів представлено в роботі [5]. У статті механізм blockchain-backed verification використовується для підвищення довіри та сумісності під час керування юридичними документами в мультихмарному середовищі. Важливим для цього дослідження є те, що коректність документа підтверджується поза межами одного сховища: за наявності

документа в кількох доменах оброблення довіра підтримується за рахунок незалежного шару верифікації. Це узгоджується з підходом, у межах якого зберігання контенту та фіксація доказової інформації архітектурно розділяються.

Подальший розвиток досліджень пов'язаний уже не просто з фіксацією операцій, а з побудовою цілісних framework для документних процесів. У дослідженні [6] запропоновано blockchain-based integrated document management framework для будівельної галузі. Автори використовують блокчейн для підтримки погодження документів, незмінюваної реєстрації змін і контролю історії версій. Ця публікація є значущим міжнародним аналогом за логікою захищеного workflow, проте предметно обмежена будівельним середовищем і не формалізує універсальну модель документа як ланцюга криптографічно пов'язаних станів.

Близьку постановку задачі можна побачити в роботі українських авторів, присвяченій open document flow на основі blockchain technology для підвищення кібербезпеки бухгалтерської системи [7]. У цій публікації блокчейн також розглядається як засіб підвищення прозорості документообігу та контролю змін. Водночас основний акцент зроблено на організаційному та прикладному аспектах document flow, тоді як формальна модель стану документа й окрема процедура доказової перевірки історії версій залишаються поза межами розгляду.

Поряд із дослідженнями, що розглядають document management systems як цілісні архітектури, у літературі представлені роботи, у яких об'єктом аналізу безпосередньо виступає модель документа. Найближчою в цьому напрямі є стаття, де запропоновано техніку chained document HTML для trusted document communication [8]. Документ у цьому підході трактується як верифікований об'єкт, а blockchain, цифровий підпис і правила конформності HTML спільно забезпечують контроль цілісності вмісту та форми подання. Для SDMS це важливий концептуальний орієнтир, оскільки він демонструє можливість криптографічного зв'язування документних подань; однак запропоноване рішення орієнтоване на спеціалізований формат chained HTML і не охоплює повний життєвий цикл документа в захищеній системі.

Якщо в низці робіт увага приділяється структурі та моделі документа, то в інших дослідженнях акцент переноситься на доказовість подій його оброблення. Так, у публікації [9] запропоновано tamper-proof distributed log system на основі блокчейна. Автори показують, що blockchain може виконувати роль незмінюваного доказового шару для журналів, критичних до прихованого редагування. Хоча предмет дослідження стосується насамперед захищеного журналювання, ця робота є важливою для SDMS, оскільки підтверджує придатність розподіленого реєстру для фіксації подій, які згодом мають перевірятися на предмет підміни.

Окремий напрям досліджень присвячено верифікації цілісності динамічно змінюваних даних. У статті [10] запропоновано blockchain-based dynamic and traceable data integrity verification scheme для середовища smart home. Автори розглядають цілісність даних в умовах оновлень і експериментально оцінюють обчислювальні накладні витрати процедури перевірки. Для цього дослідження ця робота є важливою тим, що підтверджує практичну здійсненність перевірки цілісності не лише статичного об'єкта, а і змінюваної послідовності станів.

Питання про те, які саме атрибути мають входити до доказової моделі стану, дістало розвиток у роботі [11]. Автори пропонують blockchain-based dynamic time-encapsulated data auditing, де часові мітки включаються до складу верифікованих тегів. Тим самим час розглядається не як вторинний службовий параметр, а як повноцінний елемент доказової процедури. Для SDMS цей результат є особливо значущим, оскільки часові атрибути документа розглядаються як один із контрольованих компонентів стану поряд із вмістом, метаданими та підписами.

Якщо ж розширити задачу верифікації від окремих атрибутів до узгодженості розподіленого подання документа, то ключового значення набуває співвідношення on-chain і off-chain шарів. Саме цій проблемі присвячено роботу [12], де запропоновано framework for dynamic blockchain-based data auditing. Автори прямо ставлять питання про відповідність даних, зафіксованих у блокчейні, фактичним зовнішнім даним інформаційної системи. Для архітектури SDMS цей аспект є принциповим, оскільки доказовий запис про стан документа має сенс лише за умови збереження його узгодженості із зашифрованим контентом у зовнішньому сховищі.

Ширший архітектурний контекст задає стаття [13], у якій запропоновано криптографічно орієнтовану архітектуру безпечного обміну та циркуляції data assets в uncontrolled environment. Автори описують захист повного життєвого циклу даних, включаючи зовнішнє зберігання, спільний доступ, довірене виконання та міждоменне повернення. Ця робота є важливою вже на системному рівні: вона підтверджує, що криптографічні сервіси, аудит, політики доступу та керування ключами мають проєктуватися як взаємопов'язані компоненти єдиної архітектури, а не як ізольовані модулі.

Оскільки захищений документообіг неможливий без формалізованого керування правами, наступний блок публікацій стосується access control. У статті [14] розглядаються моделі RBAC, ABAC, IAM, MAC, DAC, а також гібридні й blockchain-based підходи до керування доступом до інформаційних ресурсів підприємства. Для цього дослідження ця робота є цінною тим, що обґрунтовує необхідність урахування ролей, політик і контексту оброблення чутливої інформації. Водночас документ у ній не розглядається як криптографічно пов'язана послідовність станів.

Робота [15] присвячена гібридній системі керування обліковими даними на основі DID, IPFS і blockchain. Автори аналізують децентралізовані ідентифікатори, пов'язані з ними облікові дані, прив'язку даних в IPFS і проблеми керування ключами. У публікації представлено варіант гібридної децентралізованої архітектури, однак її фокус зосереджений на ідентифікаційних даних, а не на документному workflow і не на верифікації історії версій документа.

Розглянуті дослідження відображають розвиток підходів до побудови захищених систем керування документами – від архітектур document management systems і blockchain-орієнтованих систем зберігання до моделей документа, механізмів верифікації цілісності даних, узгодження on-chain і off-chain подань і систем децентралізованої ідентифікації.

Водночас більшість робіт зосереджується на окремих аспектах захищеного керування даними й документами, таких як незмінність записів, аудит подій, перевірка цілісності або архітектура розподілених систем. При цьому інтеграція архітектури захищеного документообігу для середовищ з обмеженим доступом, розділення контентної та доказової частин, формальної моделі документа як ланцюга криптографічно пов'язаних станів і експериментальної перевірки сценаріїв ретроактивної модифікації атрибутів документа в наявних дослідженнях представлена обмежено. Це визначає актуальність розроблення моделі захищеної системи документообігу SDMS і моделі документа у вигляді ланцюга криптографічно пов'язаних станів (Document State Chain).

Метою статті є розроблення та обґрунтування моделі захищеної системи документообігу SDMS (Secure Document Management System), яка забезпечує криптографічно перевірювану цілісність життєвого циклу документа за рахунок розділення контентного та доказового рівнів, використання формальної моделі документа у вигляді ланцюга криптографічно пов'язаних станів та застосування розподіленого реєстру для фіксації доказової інформації про події обробки документа.

ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

Архітектурна модель SDMS. SDMS розглядається як багаторівнева система, у якій функціонально розділено користувацький, прикладний, доказовий і контентний рівні. Така організація дає змогу відокремити функції взаємодії з користувачами, керування життєвим циклом документа, фіксації доказової інформації та зберігання контенту.

На користувацькому рівні забезпечуються ідентифікація та автентифікація суб'єктів, зокрема інтеграція з корпоративною інфраструктурою ідентичності, багатофакторна автентифікація та використання механізмів юридично значущого підписання, сумісних із КЕП і eIDAS.

На прикладному рівні реалізуються маршрути погодження, контроль версій і статусів, делегування повноважень, а також механізм політик доступу, який враховує роль користувача, контекст запиту та поточний стан документа.

На доказовому рівні, реалізованому на основі permissioned DLT, формується доказова частина історії документа. У розподіленому реєстрі фіксуються події життєвого циклу документа (Create, Update, Approve, Sign, Archive, Revoke), хеші станів, посилання на попередні стани, часові мітки, посилання на підписи та політики обробки. Смарт-контракти використовуються для валідації допустимих переходів між станами та формального контролю критичних операцій.

На контентному рівні зберігається виключно зашифрований вміст документа. DLT не використовується для зберігання самих файлів, а виконує функцію фіксації доказової інформації, необхідної для незалежної перевірки цілісності та історії обробки документа.

Отже, SDMS реалізує принцип архітектурного розділення, за якого вміст документа зберігається окремо та захищається криптографічними засобами, а доказова частина життєвого циклу фіксується в розподіленому незмінюваному реєстрі.

Таким чином, запропонована система SDMS реалізує принцип архітектурного розділення, за якого вміст документа зберігається у зовнішньому захищеному сховищі, тоді як доказова інформація про його стани фіксується в розподіленому незмінному реєстрі. Керування життєвим циклом документа здійснюється на прикладному рівні за допомогою формалізованих переходів між станами, які утворюють ланцюг Document State Chain. Архітектурну структуру системи SDMS та взаємодію її функціональних рівнів подано на рисунку 1.

ISSN 2663 – 4023

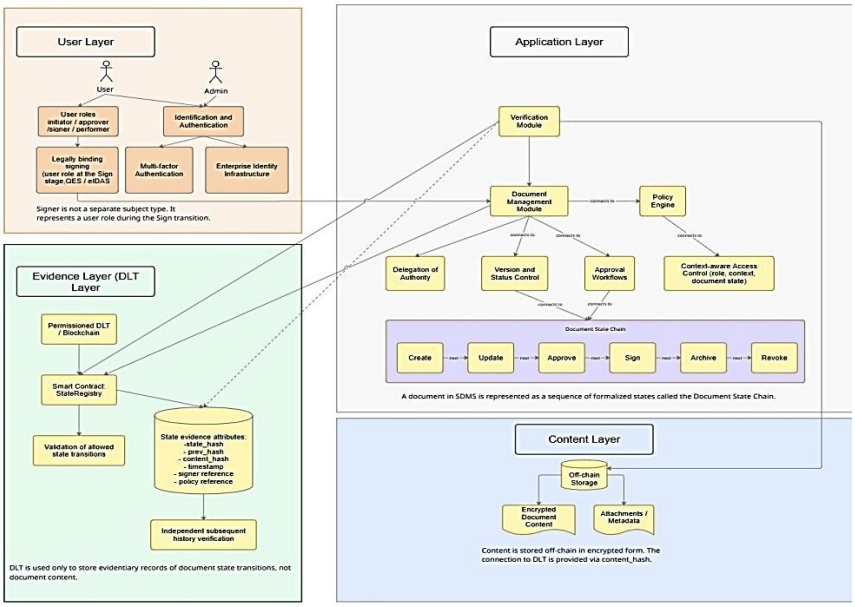

Рис. 1. Архітектурна модель захищеної системи документообігу SDMS (розроблено автором)

Формальна модель Document State Chain. Ключовим елементом запропонованої моделі є подання документа як ланцюга криптографічно пов'язаних станів. Кожна операція над документом формує новий стан, який описується сукупністю атрибутів:

$$D_i = H_i, M_i, S_i, T_i, C_i, P_i, \tag{1}$$

де H_i – хеш поточного стану; M_i – метадані стану; S_i – атрибути підписання або набір підписів; T_i – часова мітка; C_i – хеш вмісту документа або його зашифрованого представлення; P_i – посилання на попередній стан.

Для першого стану використовується нульове значення посилання на попередній стан, а для кожного наступного стану як P_iвикористовується хеш попереднього стану:

$$P_i = H_{i-1}, i > 1. \tag{2}$$

У такий спосіб формується лінійний ланцюг станів, у якому кожен новий елемент залежить від попереднього. Криптографічна зв'язність станів визначається співвідношенням:

$$H_i = \mathcal{H}(P_i \parallel M_i \parallel S_i \parallel T_i \parallel C_i), \tag{3}$$

де $\mathcal{H}(\cdot)$ – криптографічна хеш-функція, а символ $\parallel$ означає операцію конкатенації.

Співвідношення (3) означає, що хеш кожного нового стану залежить від посилання на попередній стан, поточних метаданих, атрибутів підписання, часової мітки та хешу вмісту документа. Такий підхід забезпечує ключову властивість моделі: зміна будь-якого значущого елемента стану призводить до зміни очікуваного хешу та порушення цілісності ланцюга.

Перевірка коректності стану зводиться до порівняння зафіксованого хешу з хешем, обчисленим на основі попереднього стану та поточних атрибутів:

$$Verif\ y(D_i) = \begin{cases} \text{True}, & \text{якщо } H_i = \mathcal{H}(P_i \parallel M_i \parallel S_i \parallel T_i \parallel C_i), \\ \text{False}, & \text{інакше.} \end{cases} \tag{4}$$

Для повної перевірки історії документа має виконуватися умова коректності для всіх станів ланцюга:

$$\forall i \in 1, \dots, n: \text{Verif y}(D_i) = \text{True}. \tag{5}$$

Таким чином, система або зовнішній аудитор можуть незалежно перевірити відповідність кожного стану його криптографічній моделі. У результаті історія документа набуває форми перевірюваної структури доказів, а не лише послідовності записів журналу.

Принципи захисту контенту та керування ключами. Одним із базових положень SDMS є розділення вмісту документа та доказової інформації про нього. Документ зберігається у зовнішньому сховищі лише в зашифрованому вигляді, тоді як у DLT фіксуються тільки ідентифікатори, хеші, посилання на стани, часові мітки та атрибути подій. Для кожного документа генерується унікальний симетричний ключ шифрування даних DEK, який використовується для захисту контенту, наприклад на основі AES-256-GCM. Сам ключ не зберігається разом із документом, а захищається засобами KMS/HSM і надається лише після успішної автентифікації та перевірки політик доступу. У загальному вигляді зв'язок між вмістом документа та його доказовим представленням може бути поданий як

$$C_i = \mathcal{H}\left(E_{DEK}(Content_i)\right), \tag{6}$$

де $E_{DEK}(\cdot)$ – операція шифрування вмісту документа із використанням ключа DEK.

Відкликання доступу реалізується не шляхом видалення документа, а через зміну політик доступу та керування ключовим матеріалом. У м'якому варіанті доступ припиняється блокуванням видачі DEK, а в більш жорсткому – повторним шифруванням із формуванням нового ключа. Це дає змогу забезпечити відгук доступу без порушення цілісності історії документа.

Принципи контролю доступу та аудиту. У SDMS доступ до документа визначається як функція ролі користувача, контексту звернення та поточного стану документа. У загальному вигляді правило прийняття рішення щодо доступу може бути подане як

$$Access = f(Role, Context, State, Policy), \tag{7}$$

де $Role$ – роль суб'єкта; $Context$ – контекст виконання запиту; $State$ – поточний стан документа; $Policy$ – сукупність застосовних правил доступу.

Наприклад, операція підписання може бути дозволена лише за умови, що документ перебуває у стані $approved$, а користувач має повноваження підписанта. Принципово важливою властивістю SDMS є повна трасованість змін прав доступу. Кожна зміна політики, делегування повноважень або зміна режиму доступності документа фіксується як окрема подія та включається до доказової історії.

Незмінюваний журнал дій містить відомості про те, хто виконав дію, до якого документа й версії вона належить, коли операцію було виконано, якими криптографічними атрибутами вона супроводжувалася та чи було успішно пройдено перевірку умов і політик. Завдяки фіксації подій у DLT журнал стає незалежним від довіри до централізованого адміністратора або прикладної бази даних, що робить можливою зовнішню перевірку коректності історії документа. Модель переходів станів документа. Окрім криптографічної зв'язності станів, життєвий цикл документа у SDMS описується як система допустимих переходів між станами. Кожна операція над документом переводить його з одного стану в інший відповідно до визначених правил. Формально життєвий цикл документа може бути поданий як орієнтований граф станів

$$G = (S, T), \tag{8}$$

де S – множина можливих станів документа, T – множина допустимих переходів між станами. Множина станів визначається як

$$S = draft, review, approved, signed, archived, revoked. \tag{9}$$

Кожен перехід між станами відповідає певній операції над документом:

$$T \subseteq S \times S. \tag{10}$$

Наприклад:

(draft,review), (review,approved), (approved,signed), (signed,archived) (11)

Допустимість переходу перевіряється смарт-контрактом системи відповідно до політик доступу, ролі користувача та поточного стану документа. Таким чином, смарт-контракт виступає формальним механізмом контролю коректності життєвого циклу документа. Поєднання моделі переходів станів із криптографічним ланцюгом станів Document State Chain забезпечує одночасно контроль допустимості операцій та перевірюваність історії документа.

Інтегрована формальна модель SDMS. Узагальнено модель SDMS може бути подана як сукупність компонентів, що описують стани документа, допустимі переходи між ними, правила доступу та механізм верифікації:

$$SDMS = \langle D, S, T, P, A, V \rangle, \tag{12}$$

де, D – множина документів;
S – множина можливих станів документа;
T – множина допустимих переходів між станами;
P – множина політик доступу та обробки;
A – множина суб'єктів і ролей доступу;
V – механізм криптографічної верифікації станів і ланцюга станів документа.

Для кожного документа $d \in D$його життєвий цикл визначається послідовністю станів

$$Chain(d) = \langle D_1, D_2, \dots, D_n \rangle, \tag{13}$$

де кожний наступний стан повинен одночасно задовольняти дві умови: бути криптографічно коректним та відповідати допустимому переходу життєвого циклу

$$\forall i \in \{2, \dots, n\}: \text{Verify}(D_i) = True \ \wedge \ (State_{i-1}, State_i) \in T. \tag{14}$$

Крім того, виконання операції над документом допускається лише за умови відповідності політикам доступу:

$$Allow(a, op, D_i) = True \ \Leftrightarrow \ f(Role_a, Context, State_i, P) = True, \tag{15}$$

де $a \in A$ – суб'єкт доступу, op – операція над документом, $Role_a$ – роль суб'єкта, $Context$ – контекст виконання запиту, $State_i$ – поточний стан документа, P – застосовні політики.

Таким чином, модель SDMS поєднує три взаємопов'язані властивості: допустимість переходів між станами документа, контроль доступу до операцій та криптографічно перевірювану цілісність усієї історії його обробки.

МЕТОДИКА ДОСЛІДЖЕННЯ

Методика дослідження була спрямована на експериментальну валідацію запропонованої моделі SDMS та перевірку її здатності забезпечувати верифікацію історії документа в умовах базового сценарію і контрольованого втручання. Експериментальна логіка ґрунтувалася на формуванні еталонного ланцюга станів документа, його фіксації в permissioned DLT-середовищі, зберіганні контенту у зовнішньому off-chain сховищі та подальшій перевірці узгодженості між on-chain записами, off-chain еталоном і фактичним станом контенту.

Об'єктом дослідження є процес формування, фіксації та верифікації історії документа в захищеній системі документообігу, побудованій на основі моделі Document State Chain. Для тестового документа було реалізовано типовий життєвий цикл Create → Update → Approve → Sign → Archive, у якому кожен етап формує новий криптографічно зв'язаний стан. Сформована історія використовувалася для проведення серії експериментів, спрямованих на перевірку цілісності ланцюга станів, масштабованості процедури повної верифікації та порівняльного оцінювання можливостей SDMS і baseline-моделі централізованого журналювання.

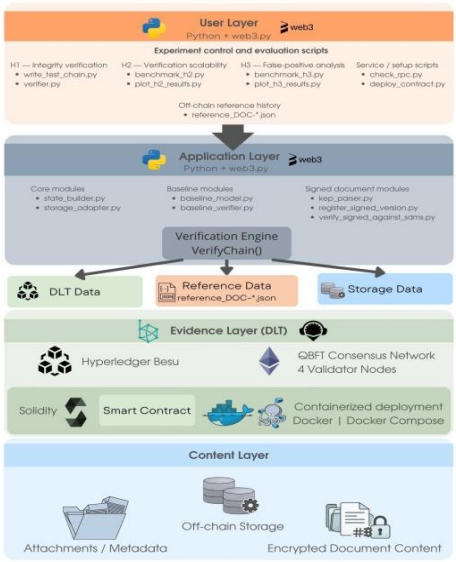

Рис. 2. Архітектуру експериментального стенда SDMS (розроблено автором)

Експериментальна база реалізована у вигляді відтворюваного локального стенда, який відображає чотирирівневу архітектуру SDMS та забезпечує керовану взаємодію між користувацьким, прикладним, доказовим і контентним рівнями. Архітектуру експериментального стенда SDMS та взаємодію його компонентів під час перевірки гіпотез H1-H3 наведено рисунку 2.

Експериментальний стенд поєднує приватну DLT-мережу Hyperledger Besu з консенсусом QBFT, смарт-контракт StateRegistry, зовнішнє mock storage, off-chain еталонне представлення історії документа та набір Python-модулів для запису і перевірки станів (табл.1).

Таблиця 1

Відповідність компонентів стенда рівням моделі SDMS

Рівень моделі SDMS	Компоненти експериментального стенда	Призначення
Користувацький рівень	write_test_chain.py, verifier.py, benchmark_h2.py, benchmark_h3.py, plot_h2_results.py, plot_h3_results.py, check_rpc.py, deploy_contract.py	Керування експериментами, запуск сценаріїв, діагностика та візуалізація результатів
Прикладний рівень	state_builder.py, storage_adapter.py, VerifyChain(), baseline_model.py, baseline_verifier.py, kep_parser.py, register_signed_version.py, verify_signed_against_sdms.py	Формування станів документа, взаємодія зі сховищем, верифікація історії, baseline-порівняння, перевірка підписаних версій
Доказовий рівень (DLT)	Hyperledger Besu, QBFT network, 4 validator nodes, StateRegistry, Solidity, Docker, Docker Compose	Зберігання on-chain доказів станів документа, забезпечення permissioned-консенсусу та контейнеризованого розгортання
Контентний рівень	Storage Data, document files, encrypted document content, attachments / metadata	Зберігання off-chain контенту документа, вкладень і супровідних метаданих

Для розгортання стенда використано Docker і Docker Compose, а взаємодію з DLT-мережею реалізовано засобами Python + web3.py через JSON-RPC інтерфейс. Така конфігурація забезпечує відтворюваність середовища, ізольованість виконання та можливість контрольованого моделювання експериментальних сценаріїв. На користувацькому рівні використовуються скрипти керування експериментами, що забезпечують формування еталонного ланцюга, запуск верифікації, виконання бенчмарків H1–H3 та побудову графіків результатів.

Прикладний рівень реалізує логіку SDMS-прототипу й включає модулі формування станів документа, взаємодії зі сховищем контенту, порівняльної baseline-моделі та обробки підписаних версій документа. Центральним елементом цього рівня є модуль VerifyChain(), який виконує перевірку

криптографічної зв'язності історії документа. Доказовий рівень реалізовано у вигляді permissioned DLT-мережі Hyperledger Besu, де в смарт-контракті StateRegistry зберігаються докази станів документа. Контентний рівень представлено зовнішнім off-chain сховищем, у якому розміщуються зашифрований вміст документа, вкладення та супровідні метадані.

Важливою особливістю стенда є те, що модуль VerifyChain() використовує три незалежні джерела вхідних даних: DLT Data, Reference Data та Storage Data. Джерелом DLT Data є on-chain представлення станів документа у смарт-контракті StateRegistry; джерелом Reference Data є off-chain еталонний файл reference_DOC-*.json; джерелом Storage Data є фактичний вміст документа та його хеші у зовнішньому сховищі. Така схема відповідає принципу розділення контентного та доказового рівнів у SDMS і забезпечує незалежну верифікацію історії документа (табл.2).

Таблиця 2

Загальна послідовність підготовки експериментального дослідження

Етап	Зміст етапу	Результат
Підготовка середовища	Налаштування Docker, Python та структури проєкту	Готове локальне середовище
Розгортання DLT-мережі	Запуск 4-вузлової мережі Hyperledger Besu QBFT	Працююча permissioned-мережа
Перевірка мережевої взаємодії	Контроль RPC-доступу та взаємодії з вузлами	Підтверджена доступність мережі
Деплой смарт-контракту	Розгортання StateRegistry	Готовий on-chain реєстр станів
Налаштування off-chain модулів	Підключення state_builder.py, storage_adapter.py, verifier.py	Готова логіка запису та перевірки
Формування еталонного ланцюга	Створення базової історії документа	Еталонний on-chain / off-chain ланцюг
Проведення серій експериментів	Окремі перевірки для H1, H2, H3	Дані для аналізу гіпотез

У межах дослідження було сформульовано три робочі гіпотези, спрямовані на перевірку ключових властивостей запропонованої моделі SDMS. Зокрема, оцінювалася здатність системи виявляти ретроактивне втручання в історію документа та локалізувати перший порушений стан, поведінка процедури повної верифікації зі збільшенням довжини ланцюга станів документа, а також порівняльна ефективність моделі SDMS відносно baseline-підходу централізованого журналювання за властивостями tamper-evidence та auditability. Для кожної гіпотези було визначено окрему процедуру підготовки та проведення експерименту, а також набір показників і критеріїв оцінювання. Загальна послідовність підготовки експериментального дослідження наведена в табл. 2, а методики перевірки гіпотез H1-H3 – у табл. 3-5.

Гіпотеза H1. Ретроактивна модифікація будь-якого значущого атрибута попереднього стану документа (вмісту, метаданих, підписів або часових атрибутів) повинна порушувати криптографічну зв'язність Document State Chain і детектуватися модулем верифікації.

Таблиця 3

Методика перевірки гіпотези H1

Параметр	Опис
Мета перевірки	Оцінити здатність SDMS виявляти ретроактивну модифікацію значущих атрибутів стану документа
Експериментальна база	Еталонний ланцюг станів документа в StateRegistry, зовнішнє сховище, off-chain еталон
Підготовка	Формування коректного ланцюга Create → Update → Approve → Sign → Archive
Проведення	Послідовне моделювання контрольованих втручань у вміст, метадані, підписи та часові атрибути з повторним запуском VerifyChain()
Показники	Result, тип невідповідності, локалізація першого порушеного стану
Критерій підтвердження	Для кожного сценарію втручання очікується False та фіксація порушення цілісності

Гіпотеза H2. Зі збільшенням довжини ланцюга станів документа процедура VerifyChain() демонструє передбачуване зростання часу повної верифікації, яке не супроводжується втратою коректності перевірки та зберігає практичну придатність механізму для документів із довшою історією станів.

Таблиця 4

Методика перевірки гіпотези H2

Параметр	Опис
Мета перевірки	Оцінити залежність часу повної верифікації від довжини ланцюга станів документа
Експериментальна база	Серії коректних ланцюгів із різною кількістю станів K
Підготовка	Формування наборів ланцюгів довжиною $K = 5{,}10{,}20{,}50{,}100{,}200$
Проведення	Повторний запуск процедури повної верифікації для кожного значення Kз фіксацією часу виконання
Показники	Характер залежності $t_{verify\_chain}(K)$, середній час перевірки для каждого K
Критерій підтвердження	Зі збільшенням Kчас повної верифікації має зростати монотонно, без втрати коректності результату перевірки, з характером залежності, близьким до лінійного

Гіпотеза H3. Для сценаріїв порушення історії документа модель SDMS забезпечує вищу здатність до виявлення втручання, доказову локалізацію порушеного стану та відтворюваність аудиту порівняно з традиційним підходом централізованого журналювання.

Таблиця 5

Методика перевірки гіпотези H3

Параметр	Опис
Мета перевірки	Порівняльно оцінити можливості SDMS та baseline-моделі централізованого журналювання щодо виявлення втручання в історію документа, локалізації першого порушеного стану та забезпечення незалежної перевірюваності результатів аудиту
Експериментальна база	Набір тестових сценаріїв B1-B5, що моделюють часткове, ретроактивне та узгоджене втручання в історію документа
Підготовка	Формування еталонного ланцюга станів документа, побудова відповідного baseline-подання історії та підготовка модифікованих сценаріїв порушення цілісності
Проведення	Послідовний запуск процедур перевірки для SDMS і baseline-моделі в кожному зі сценаріїв B1-B5 з фіксацією результатів виявлення, локалізації та аудиторської перевірки
Показники	Detection rate, auditability, composite capability
Критерій підтвердження	SDMS має демонструвати вищі значення detection rate, auditability та composite capability порівняно з baseline-моделлю

Оцінювання результатів експериментів здійснювалося за системою показників, узгоджених із відповідними гіпотезами. Для H1 основним критерієм була здатність системи детектувати будь-яку ретроактивну модифікацію значущих атрибутів стану документа та локалізувати перший порушений стан. Для H2 оцінювалася часова складність повної верифікації залежно від довжини ланцюга станів документа. Для H3 визначальним критерієм було порівняльне оцінювання властивостей tamper-evidence та auditability у моделі SDMS і baseline-моделі централізованого журналювання, зокрема здатності виявляти втручання, локалізувати перший порушений стан і забезпечувати незалежну перевірюваність результатів аудиту. Узагальнення критеріїв, показників і параметрів оцінювання (табл. 6).

Таблиця 6

Критерії, показники та параметри оцінювання

Гіпотеза	Основний показник	Додаткові показники	Очікуваний результат
H1	Result	Тип невідповідності, локалізація порушення	Детекція будь-якого ретроактивного втручання
H2	Час повної верифікації (verify_time_ms)	Середній час для кожного K, характер залежності	Монотонне зростання часу зі збільшенням довжини ланцюга, близьке до лінійного
H3	Detection rate	Auditability, composite capability	Перевага SDMS над baseline-моделлю за здатністю виявлення, локалізації та незалежної перевірки

Таким чином, запропонована методика поєднує відтворюваний експериментальний стенд, формально визначену процедуру формування ланцюга станів документа та систему критеріїв оцінювання, орієнтовану на перевірку здатності моделі виявляти порушення цілісності історії, забезпечувати практично придатну повну верифікацію та перевищувати baseline-модель за властивостями tamper-evidence й auditability. Це створює підґрунтя для безпосереднього зіставлення теоретичної моделі SDMS з результатами її реалізації у мінімальному функціональному прототипі.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

З урахуванням наведеного, для експериментального дослідження було розгорнуто локальний стенд на базі чотирьох вузлів-валідаторів Hyperledger Besu.

Такий вибір ґрунтується на класичному співвідношенні n = 3f + 1, де n – кількість валідаторів, а f – максимально допустима кількість вузлів із відмовою або некоректною поведінкою. Для n = 4 отримуємо f = 1, тобто мережа зберігає коректність досягнення консенсусу навіть за умови відмови або компрометації одного валідатора.

Така конфігурація забезпечує можливість моделювання QBFT-консенсусу в мінімально стійкому BFT-сценарії та водночас залишається придатною для локального відтворення без надмірного навантаження на обчислювальне середовище.

```
PS C:\work\sdms-besu-qbft> docker compose ps
NAME      IMAGE                    COMMAND                  SERVICE   CREATED      STATUS         PORTS
besu1     hyperledger/besu:latest  "besu-entry.sh --dat…"   besu1     11 days ago  Up 7 minutes   8546-8547/tcp, 8550-8551/tcp, 30303/tcp, 0.0.0.0:18545->8545/tcp
besu2     hyperledger/besu:latest  "besu-entry.sh --dat…"   besu2     11 days ago  Up 7 minutes   8545-8547/tcp, 8550-8551/tcp, 30303/tcp
besu3     hyperledger/besu:latest  "besu-entry.sh --dat…"   besu3     11 days ago  Up 7 minutes   8545-8547/tcp, 8550-8551/tcp, 30303/tcp
besu4     hyperledger/besu:latest  "besu-entry.sh --dat…"   besu4     11 days ago  Up 7 minutes   8545-8547/tcp, 8550-8551/tcp, 30303/tcp
PS C:\work\sdms-besu-qbft>
```

Рис. 3. Стан контейнерів Docker після запуску локального стенда Hyperledger Besu з чотирма вузлами-валідаторами QBFT

На рис. 3 подано результат запуску контейнерів Docker, який підтверджує коректне розгортання та одночасну роботу всіх чотирьох вузлів мережі.

Для підтвердження готовності стенда до виконання експериментів було перевірено доступність RPC-інтерфейсу вузла та мережеву доступність порту, через який здійснюється взаємодія з блокчейн-мережею. На рис. 4 показано результат цієї перевірки, що підтверджує успішне підключення до клієнта Besu та коректну роботу локального мережевого інтерфейсу.

```
(.venv) PS C:\work\sdms-besu-qbft> python .\python\check_rpc.py
Connected: True
Client version: besu/v26.1.0/linux-x86_64/openjdk-java-25
Current block: 114074
(.venv) PS C:\work\sdms-besu-qbft> Test-NetConnection localhost -Port 18545


ComputerName     : localhost
RemoteAddress    : ::1
RemotePort       : 18545
InterfaceAlias   : Loopback Pseudo-Interface 1
SourceAddress    : ::1
TcpTestSucceeded : True
```

Рис. 4. Перевірка доступності JSON-RPC-інтерфейсу вузла Hyperledger Besu та мережевої доступності порту 18545

Перед початком експериментів смартконтракт StateRegistry.sol компілювали та розгортали в локальній мережі Hyperledger Besu. Компіляцію виконували в контейнеризованому середовищі Docker з використанням параметра -evm-version=berlin, оскільки саме така конфігурація забезпечувала сумісність з параметрами розгорнутої мережі. За відсутності цього параметра можливе формування несумісного байткоду, що призводить до неуспішного виконання транзакції розгортання.

У результаті формувалися файли StateRegistry.abi і StateRegistry.bin, які далі використовувалися для розгортання контракту в мережі. Розгортання виконували за допомогою Python-скрипта (рис. 5)

```
(.venv) PS C:\work\sdms-besu-qbft> python .\python\deploy_contract.py
Connected: True
Sender: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Balance: 1000 ETH
Deployment tx hash: afeec2470c48c96c87b82ed03b2c546ae159c021f4b4f2fe9b40187248 8f8acb
Receipt status: 1
Contract address: 0xc8Cfa1a751829803e9dd39f860f9Fbe0330bA1c5
Block number: 115454
Code size: 3670
Deployment info saved to: C:\work\sdms-besu-qbft\python\deployment.json
(.venv) PS C:\work\sdms-besu-qbft>
```

Рис. 5. Результат компіляції та розгортання смартконтракту

Успішність розгортання визначали за такими ознаками: встановлення з'єднання з вузлом (Connected: True), отримання хешу транзакції, значення Receipt status: 1, фіксація адреси контракту та збереження параметрів розгортання у файл deployment.json. На рис. X наведено результат успішного розгортання смартконтракту в локальній мережі.

Після розгортання смартконтракту, налаштування взаємодії з мережею та перевірки працездатності експериментального стенда було виконано серію експериментів, спрямованих на оцінювання коректності запропонованої моделі документа, стійкості механізму верифікації до модифікації атрибутів стану та практичної придатності повної перевірки ланцюга станів документа.

1. Результати експерименту перевірки детекції ретроактивних змін. Перший експеримент був спрямований на перевірку гіпотези H1, відповідно до якої будь-яка ретроактивна модифікація значущих атрибутів попереднього стану документа повинна порушувати криптографічну зв'язність Document State Chain і детектуватися модулем верифікації. Для цього було сформовано еталонний ланцюг станів документа, що відображає типовий життєвий цикл *Create → Update → Approve → Sign → Archive*. Кожен стан містив метадані, атрибути підписання, часову мітку, хеш вмісту документа, посилання на попередній стан і поточний хеш стану.

Після формування базового ланцюга було виконано серію сценаріїв контрольованого втручання. У сценарії A1 моделювалася підміна вмісту документа у зовнішньому сховищі, у сценарії A2 – ретроактивна модифікація метаданих стану, у сценарії A3 – зміна набору підписів, а у сценарії A4 – маніпуляція часовою міткою. Для кожного сценарію після внесення змін повторно запускалася процедура повної верифікації за допомогою модуля VerifyChain(), який порівнює on-chain записи у StateRegistry, off-chain еталонну історію та фактичний стан контенту у зовнішньому сховищі.

Формування еталонного ланцюга станів виконується скриптом write_test_chain.py. Репрезентативний фрагмент реалізації наведено в лістингу 1.

Лістинг 1. Формування еталонного ланцюга станів документа та його фіксація в StateRegistry

```
workflow = ["Create", "Update", "Approve", "Sign", "Archive"]

doc_id = generate_doc_id()
prev_hash = ZERO_HASH
states = []

for event_type in workflow:
    state = build_state(
        doc_id=doc_id,
        event_type=event_type,
        prev_hash=prev_hash,
        metadata=make_metadata(event_type),
        signatures=make_signatures(event_type),
        timestamp=current_timestamp(),
        content_hash=compute_content_hash(doc_path),
    )

    tx_hash = registry.functions.registerState(
        state["doc_id"],
        state["state_id"],
        state["state_hash"],
        state["prev_hash"],
        state["event_type"],
        state["timestamp"]
    ).transact()

    states.append(state)
    prev_hash = state["state_hash"]

save_reference_json(doc_id, states)
```

Лістинг 1 демонструє, що кожен новий етап життєвого циклу документа формує окремий стан, який одночасно включається до off-chain еталонної історії та фіксується on-chain у смарт-контракті StateRegistry. Саме така подвійна репрезентація станів забезпечує основу для подальшої незалежної перевірки історії документа. Ключова логіка верифікації узагальнено наведена в лістингу 2.

Лістинг 2. Узагальнена логіка процедури VerifyChain() для перевірки узгодженості on-chain, off-chain та storage-представлень документа

```
def verify_chain(onchain_states, reference_states, storage_adapter):
    for i, (onchain, ref_state) in enumerate(zip(onchain_states, reference_states), start=1):
        actual_content_hash = storage_adapter.get_content_hash(ref_state["content_ref"])

        if actual_content_hash != ref_state["content_hash"]:
            return False, i, "content_hash mismatch"

        expected_state_hash = compute_state_hash(
            prev_hash=ref_state["prev_hash"],
            metadata=ref_state["metadata"],
            signatures=ref_state["signatures"],
            timestamp=ref_state["timestamp"],
            content_hash=ref_state["content_hash"],
        )

        if onchain["timestamp"] != ref_state["timestamp"]:
            return False, i, "timestamp mismatch"

        if onchain["state_hash"] != expected_state_hash:
            return False, i, "state_hash mismatch"

    return True, None, None
```

Як видно з лістингу 2, процедура VerifyChain() використовує три незалежні джерела даних: on-chain стани, off-chain reference-історію та фактичні хеші контенту зі сховища. Це дозволяє не лише фіксувати факт порушення, а й класифікувати його за типом невідповідності.

У базовому сценарії Baseline Verify отримано позитивний результат True, що підтверджує коректність еталонної історії документа та узгодженість між on-chain реєстром, off-chain еталоном і фактичним вмістом документа. Для всіх сценаріїв контрольованого втручання (A1-A4) отримано результат False, що свідчить про успішну детекцію порушення цілісності. Зведені результати наведено в табл. 7.

Таблиця 7

Результати верифікації для сценаріїв Baseline та A1-A4

№	Сценарій	Result	Detection locality	Зафіксовані невідповідності	Інтерпретація
1	Baseline Verify	True	–	Відсутні	Еталонна історія документа узгоджується з on-chain реєстром і зовнішнім сховищем
2	A1: Підміна вмісту файлу	False	2	content_hash mismatch; state_hash mismatch	Підміна фактичного файлу змінює content_hash і порушує доказову цілісність стану
3	A2: Підміна метаданих	False	2	state_hash mismatch	Ретроактивна зміна метаданих другого стану руйнує криптографічну зв'язність історії
4	A3: Підміна підписів	False	3	state_hash mismatch	Заміна набору підписів третього стану змінює хеш стану та детектується верифікатором
5	A4: Маніпуляція часовою міткою	False	2	timestamp mismatch; state_hash mismatch	Зміна часової мітки виявляється безпосередньо та додатково порушує хеш стану

Типовий приклад консольного виводу процедури верифікації може бути наведено в лістингу 3.

Лістинг 3. Консольний вивод результатів верифікації

```
(.venv) PS C:\work\sdms-besu-qbft> python .\python\verifier.py
Verifying doc_id: DOC-1773146471

=== BASELINE VERIFY ===
Result: True
Chain integrity is valid.

=== A1 TAMPER TEST ===
Result: False
Detection locality (first failed index): 2
Issues:
- content_hash mismatch:
actual=f910a41f85764916c3dbc84be17d95e6394bace0910889d99cb60970656c58c5,
expected=3cf40d18861b9026780ed50dff731088bfc4410b0f5cb88a10caf9defc53986c
- state_hash mismatch: on-
chain=0xb410dc8f5a9f926497323062d175c59de69ab0895e52f9aec13b290ce8d2975b,
expected=0x47f981278d4b93cbd1db39e8a6b98fc97c75fa92361aa77fdee3090329ed8c73

=== A2 TAMPER TEST ===
Result: False
Detection locality (first failed index): 2
Issues:
- state_hash mismatch: on-
chain=0xb410dc8f5a9f926497323062d175c59de69ab0895e52f9aec13b290ce8d2975b,
expected=0xa735514de8cc4cf2baed4bd8a51ddd1420661d73955a6d434145d8b71a6c400b

=== A3 TAMPER TEST ===
Result: False
Detection locality (first failed index): 3
Issues:
- state_hash mismatch: on-
chain=0x89895a3013c2cc560bb12f746d2a5303b003dc8eb187af18509152517a0307d1,
expected=0xc255fea32795bc344c224d7194c428e19ccdd762c4041577b89019b4a57f248a

=== A4 TAMPER TEST ===
Result: False
Detection locality (first failed index): 2
Issues:
- timestamp mismatch: on-chain=1773146474, expected=1773150074
- state_hash mismatch: on-
chain=0xb410dc8f5a9f926497323062d175c59de69ab0895e52f9aec13b290ce8d2975b,
expected=0xfbfaf33093c809b923df6ac0a564ae9b00171165eb365b7b801df782f7ab43f3
(.venv) PS C:\work\sdms-besu-qbft>
```

Отримані результати демонструють, що запропонована модель SDMS формує криптографічно зв’язану доказову структуру історії документа, а не просто послідовність журнальних записів. Будь-яке ретроактивне втручання в значущі атрибути стану призводить до порушення очікуваної структури доказів і виявляється процедурою VerifyChain().

Особливо показовим є те, що для сценарію A1 система фіксує пряму невідповідність content_hash, а для сценаріїв A2 та A3 основним індикатором виступає state_hash mismatch, що відображає руйнування криптографічної зв’язності ланцюга. У сценарії A4 зміна часової мітки детектується як безпосередньо через timestamp mismatch, так і опосередковано через порушення хешу стану. Таким чином, результати першого експерименту підтверджують гіпотезу H1.

2. Результати експерименту оцінювання масштабованості повної верифікації. Другий експеримент було спрямовано на перевірку гіпотези H2, відповідно до якої час повної верифікації історії документа зростає зі збільшенням довжини ланцюга станів і для коректно побудованого ланцюга демонструє передбачувану, близьку до лінійної залежність. На відміну від першого експерименту, де оцінювалася здатність системи виявляти ретроактивне втручання, у цьому випадку основним об’єктом аналізу була

обчислювальна поведінка процедури VerifyChain() при збільшенні кількості послідовно перевірюваних станів документа.

У межах експерименту формувалися коректні ланцюги станів довжиною $K = 5,10,20,50,100,200$, де K – кількість послідовних станів одного документа, що входять до процедури повної перевірки. Для кожного значення Kпроцедура верифікації виконувалася 7 разів, після чого обчислювався середній час виконання. В усіх серіях вимірювань для всіх значень Kотримано позитивний результат верифікації (result = True), що підтверджує коректність сформованих тестових ланцюгів і придатність отриманих даних для аналізу часової залежності.

Логіку бенчмарка H2 відображено в лістингу 4. Для кожного значення Kспочатку будувався коректний ланцюг заданої довжини, після чого запускалася процедура verify_chain_full(doc_id=current_doc_id()), а час її виконання фіксувався у мілісекундах як показник verify_time_ms. Серійний характер вимірювань давав змогу зменшити вплив випадкових коливань і отримати стійкі середні значення часу повної верифікації.

Лістинг 4. Фрагмент бенчмарка H2 для вимірювання часу повної верифікації залежно від довжини ланцюга станів

```
k_values = [5, 10, 20, 50, 100, 200]
repeats = 7
results = []

for k in k_values:
    for r in range(repeats):
        build_clean_chain(length=k)

        start = perf_counter()
        result = verify_chain_full(doc_id=current_doc_id())
        elapsed_ms = (perf_counter() - start) * 1000.0

        results.append({
            "K": k,
            "repeat": r + 1,
            "result": result,
            "verify_time_ms": elapsed_ms,
        })

save_csv("h2_results_raw.csv", results)
save_summary("h2_results_summary.csv", results)
```

Виконуються не для одного запуску, а для серії повторів при різних значеннях K, що дає змогу зменшити вплив випадкових коливань та отримати стійкі середні значення часу повної верифікації.

Зведені результати вимірювань наведено в табл. 8. Як видно з таблиці, зі збільшенням довжини ланцюга станів середній час повної верифікації зростав монотонно: від 67.821 мс для $K = 5$до 2202.643 мс для $K = 200$. При цьому для всіх досліджених значень процедура завершувалася успішно, що свідчить про збереження коректності перевірки незалежно від довжини історії документа.

Таблиця 8

Час повної верифікації для різної довжини ланцюга станів

K	**Середній час перевірки, мс**	**Результат верифікації**
5	67.821	True
10	125.104	True
20	247.386	True
50	602.917	True
100	1167.553	True
200	2202.643	True

Для графічного подання результатів використано скрипт plot_h2_results.py. Як показано в табл. 8, для всіх досліджених значень Kповна верифікація завершувалася успішно, а середній час перевірки монотонно зростав зі збільшенням довжини ланцюга станів документа.

На рис. 6 наведено залежність середнього часу повної верифікації від довжини ланцюга станів документа разом із лінійною апроксимацією. Отримана залежність є близькою до лінійної, що підтверджується значенням $R^2 \approx 0.969$. Це свідчить про відсутність різкого погіршення часових характеристик процедури VerifyChain() зі зростанням K.

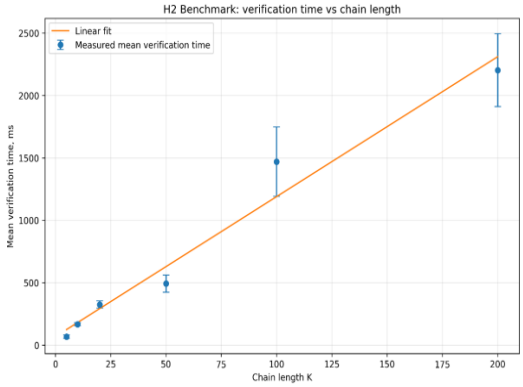

Рис. 6. Лінійна апроксимація залежності $t_{verify_chain}(K)$від довжини ланцюга станів

Для кількісної інтерпретації результатів було виконано лінійну апроксимацію залежності часу повної верифікації від довжини ланцюга:

$$t_{verify\_chain}(K) \approx \alpha + \beta K,$$

де $\alpha \approx 68.25$мс – оцінка постійної складової витрат, $\beta \approx 11.21$мс/стан – середній приріст часу на один додатковий стан, $R^2 \approx 0.969$ – коефіцієнт детермінації.

Значення R^2свідчить про добру відповідність лінійної моделі експериментальним даним. Отже, процедура повної верифікації характеризується передбачуваною часовою поведінкою: зі збільшенням довжини історії документа час перевірки закономірно зростає, але без вибухового ускладнення. Це дає підстави вважати механізм повної верифікації практично придатним для систем, у яких кількість послідовних станів документа є помірною або може бути організована з використанням архівного розбиття. Отримані результати підтверджують гіпотезу H2.

3. Результати експерименту порівняльної оцінки доказової стійкості SDMS і централізованого журналювання. Третій експеримент був спрямований на порівняльне оцінювання доказової стійкості запропонованої моделі SDMS та baseline-моделі централізованого журналювання.

Метою експерименту було перевірити, чи забезпечує SDMS вищий рівень детекції ретроактивного втручання, локалізації першого порушеного стану та незалежної перевірюваності результату порівняно з централізованим підходом без окремого доказового шару.

У межах runbook цей експеримент реалізовано окремими скриптами baseline_model.py, baseline_verifier.py, benchmark_h3.py та plot_h3_results.py.

У порівняльному дослідженні використовувалися однакові reference-історії документа для двох моделей:

1. SDMS, у якій історія документа подана як ланцюг криптографічно зв’язаних станів і фіксується в DLT через StateRegistry;
2. baseline-модель, у якій використовується централізований журнал подій без незалежного on-chain доказового представлення.

Для кожного сценарію втручання оцінювалися три критерії: detected, localization та independent verifiability. Очікувані результати передбачали, що SDMS повинна демонструвати стабільно високі значення цих показників для всіх сценаріїв B1-B5, тоді як baseline-модель – помітно нижчі, особливо в разі узгодженого ретроактивного редагування журналу. Фрагмент коду порівняльного бенчмарка наведено в лістингу 5.

Лістинг 5. Узагальнена логіка порівняльного бенчмарка H3

```
reference = load_reference(args.reference)

sdms_model = build_sdms_from_reference(reference)
baseline_log = build_baseline_from_reference(reference)

results = []

for scenario in [“B1”, “B2”, “B3”, “B4”, “B5”]:
   for r in range(args.repeats):
      sdms_case = apply_scenario_sdms(sdms_model, scenario)
      baseline_case = apply_scenario_baseline(baseline_log, scenario)

      sdms_verdict = verify_sdms_case(sdms_case)
      baseline_verdict = verify_baseline_case(baseline_case)

      results.append(compare_models(
         scenario=scenario,
         sdms_verdict=sdms_verdict,
         baseline_verdict=baseline_verdict
      ))

save_raw_results(“h3_results_raw.csv”, results)
save_summary_results(“h3_results_summary.csv”, results)
```

Порівняння виконується на одних і тих самих reference-даних і для однакових сценаріїв втручання, а відмінності в результатах зумовлені саме архітектурою SDMS і baseline-моделі, а не різницею у вхідних даних.

На рис. 7 представлено фрагмент консольного виводу, що відображає агреговані показники detection rate для моделі SDMS та baseline-підходу централізованого журналювання. Якщо модель SDMS демонструє повну детекцію втручання у всіх сценаріях (detection rate = 1.0), то baseline-модель централізованого журналювання виявляє порушення лише в окремих сценаріях (detection rate = 0.2). Це безпосередньо впливає на значення інтегрального показника, оскільки відсутність надійної детекції робить неможливим подальшу локалізацію порушеного стану та незалежну перевірку результатів. У результаті модель SDMS у всіх сценаріях B1-B5 досягає максимального значення composite capability, тоді як baseline-модель демонструє значно нижчі результати, що підтверджує системну перевагу запропонованої моделі за властивостями tamper-evidence та auditability.

На рис. 8 наведено підсумкові результати бенчмарка, зокрема значення інтегрального показника composite capability та посценарні значення цього показника для тестових сценаріїв B1-B5.

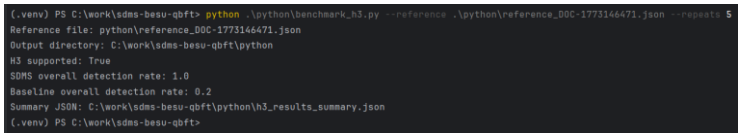

Рис. 7. Консольний вивід виконання порівняльного бенчмарка H3 (показники detection rate)

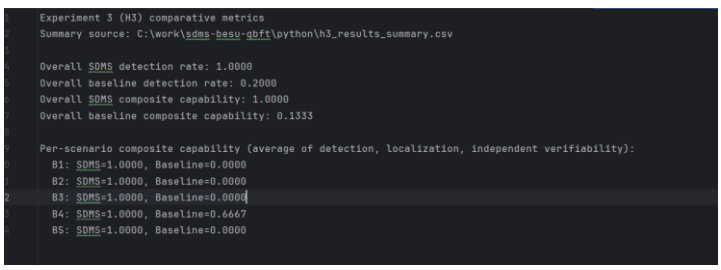

Рис. 8. Консольний вивід результатів порівняльного бенчмарка H3 (показник composite capability та посценарні значення для B1-B5)

Агреговані метрики демонструють, що SDMS забезпечує повну детекцію та повну доказову придатність результату (1.0), тоді як централізоване журналювання виявляє втручання лише епізодично (0.2) і не забезпечує повноцінної незалежної перевірюваності результатів (0.1333). Це узгоджується з гіпотезою H3 про системну перевагу SDMS у стійкості до прихованої ретроактивної модифікації історії документа.

Зведені результати порівняльного експерименту H3 наведено в табл. 9. У таблиці представлено посценарні значення інтегрального показника composite capability для моделі SDMS та baseline-підходу централізованого журналювання. Композитний показник визначається як середнє значення трьох властивостей: здатності виявляти втручання (detection), локалізувати перший порушений стан (localization) та забезпечувати незалежну перевірюваність результату (independent verifiability).

Таблиця 9 відображає результати для тестових сценаріїв B1-B5, що моделюють різні типи порушення історії документа, включаючи ретроактивні зміни станів, модифікацію атрибутів історії та узгоджене редагування журналу подій.

Таблиця 9

Порівняльні результати експерименту 3 для моделі SDMS та baseline-підходу централізованого журналювання

Сценарій	Тип перевірки	SDMS (composite capability)	Baseline SDMS (composite capability)	Інтерпретація	Порівняльна оцінка
B1	Детекція ретроактивної модифікації	1.0000	0.0000	SDMS: 3/3 (детекція+локалізація +незалежна перевірка); Baseline: 0/3	Максимальна перевага SDMS
B2	Локалізація першого порушеного стану	1.0000	0.0000	SDMS: 3/3; Baseline: 0/3	Максимальна перевага SDMS
B3	Незалежна перевірюваність результату	1.0000	0.0000	SDMS: 3/3; Baseline: 0/3	Максимальна перевага SDMS
B4	Стійкість до модифікації атрибутів історії	1.0000	0.6667	SDMS: 3/3; Baseline: 2/3 (часткове виконання критеріїв без повної доказової перевірюваност)	Перевага SDMS (baseline частково спрацьовує)
B5	Стійкість до узгодженого ретроактивного редагування журналу	1.0000	0.0000	SDMS: 3/3; Baseline: 0/3	Максимальна перевага SDMS

Результати порівняльного експерименту H3 демонструють принципову відмінність між моделлю SDMS та baseline-підходом централізованого журналювання з погляду доказовості, цілісності та незалежної перевірюваності історії документа. На рис. 9 наведено інтегральний показник композитної здатності до забезпечення цілісності та аудиту (composite integrity/audit capability) для сценаріїв B1-B5.

Для SDMS значення показника є максимальним і становить 1.0 у всіх сценаріях, що означає одночасне виконання трьох ключових властивостей: детекції втручання, локалізації першого порушеного стану та незалежної перевірюваності результату. Натомість baseline-модель демонструє нульові значення для сценаріїв B1, B2, B3, B5 (композит = 0.0), а частковий результат спостерігається лише для сценарію B4 (композит ≈ 0.67). Таким чином, централізоване журналювання виявляється нездатним забезпечити повноцінну доказову придатність результатів у більшості типових сценаріїв ретроактивного втручання, тоді як SDMS зберігає повний набір властивостей у всіх випадках (рис. 9).

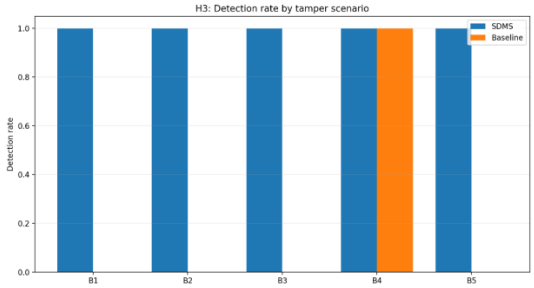

Рис. 9. Порівняння detection rate для SDMS і baseline-моделі за сценаріями B1-B5

Окремо на рис. 9 показано частоту детекції (detection rate) за кожним сценарієм. Для SDMS детекція дорівнює 1.0 у всіх сценаріях B1-B5, що підтверджує чутливість моделі Document State Chain до ретроактивної модифікації будь-якого значущого атрибута стану (контенту, метаданих, підписних атрибутів, часових параметрів або зв'язності станів). Для baseline-моделі детекція спостерігається лише у сценарії B4 (detection = 1.0), тоді як у сценаріях B1, B2, B3, B5 детекція відсутня (detection = 0.0). Це означає, що централізований журнал здатний "помітити" лише ті порушення, які безпосередньо руйнують очевидну часову узгодженість або формальну логіку записів, тоді як "узгоджені" ретроактивні зміни залишаються для нього недетектованими (рис. 9).

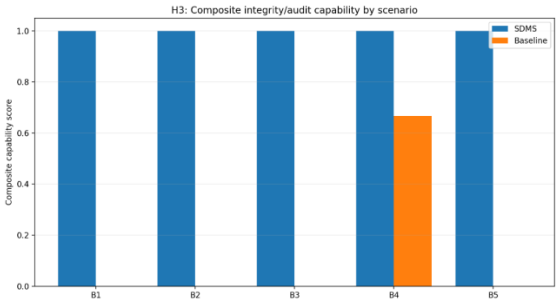

Рис. 10. Порівняння інтегрального capability score для SDMS і baseline-моделі

SDMS демонструє не лише факт детекції, а й повну доказову придатність результату для аудиту, що проявляється у композитному показнику 1.0 (рис. 10). По-друге, baseline-модель навіть у випадку, коли здатний детектувати втручання (B4), не забезпечує повної доказової сили, оскільки композитний показник залишається нижчим за 1.0 (рис. 10), тобто відсутній принаймні один із необхідних елементів: або формальна локалізація першого порушення, або незалежна перевірюваність висновку. Особливо показовим є сценарій B5 ("узгоджена" ретроактивна правка централізованого журналу), де baseline має нульову детекцію та нульову композитну здатність, тоді як SDMS зберігає максимальні значення. Саме цей результат підтверджує системну перевагу запропонованої моделі: фіксація доказових атрибутів у DLT та криптографічна зв'язність станів перетворюють ретроактивне переписування історії на відтворювано виявлюване порушення.

Прикладний інтерес становить сценарій B6, який спрямований на демонстрацію прикладного аспекту доказової моделі SDMS у сценарії, коли два документи підписані однією й тією самою особою кваліфікованим електронним підписом (КЕП), але мають різний вміст. Така ситуація є типовою для практики: перевірка валідності підпису підтверджує коректність підпису для конкретного файла, однак не відповідає на питання, чи є поданий файл саме тією версією, яка була раніше зафіксована в процесі керованого життєвого циклу документа.

У межах сценарія перша підписана версія (контейнер 1.pdf.p7s) була заякорена як еталонна для заданого doc_id шляхом фіксації її доказових атрибутів (зокрема content_hash та похідного evidence_hash) з формуванням JSON-анкера. Далі виконувалась перевірка другої підписаної версії (2.pdf.p7s) відносно першої заякореної версії. Результат верифікації показав, що підписувач для обох файлів збігається (Same signer: True), а час підписання другої версії не суперечить часовій послідовності (Candidate is later than or equal to anchor: True).

Водночас встановлено, що вміст другої версії не відповідає заякореній еталонній версії (Content matches anchored version: False), тобто має місце відмінність контенту відносно першого зафіксованого стану (Changes relative to first anchored version: True). За класифікатором системи результат віднесено до

класу DIFFERENT_SIGNED_VERSION_SAME_SIGNER, що інтерпретується як «інша підписана версія того самого документа тим самим підписувачем» (рис.11).

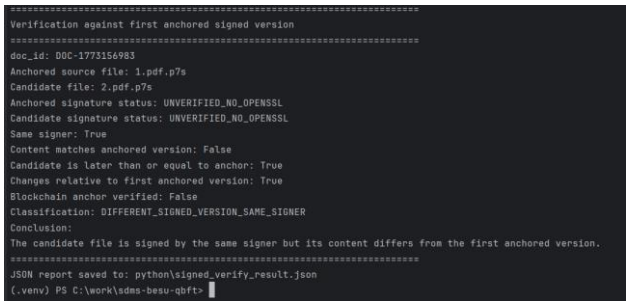

Рис.11. Консольний вивід перевірки B6

Таким чином, сценарій B6 демонструє, що SDMS уміє розрізняти два випадки, які у традиційних підходах часто змішуються: валідність підпису як криптографічної операції та тотожність поданого файла раніше зафіксованій версії документа.

У запропонованій моделі доказовість забезпечується не лише фактом підписання, а й зв'язком підписаної версії з доказовим представленням (якорем) першого стану, що дозволяє виявляти повторно підписані модифіковані версії як відмінні від первинно зафіксованої версії навіть за умови однакового підписувача. Вихідний JSON-звіт перевірки збережено у файлі python\signed_verify_result.json.

Експеримент 3 продемонстрував системну перевагу SDMS над baseline-моделлю централізованого журналювання за критеріями виявлення втручання та аудитопридатності: для SDMS у сценаріях B1–B5 отримано detection rate = 1.0 і composite capability = 1.0, тоді як baseline показав лише detection rate = 0.2 і composite capability = 0.1333, а в сценарії B5 мав нульові показники. Додатково сценарій B6 підтвердив прикладну цінність підходу, оскільки система коректно класифікувала другу КЕП-підписану версію як DIFFERENT_SIGNED_VERSION_SAME_SIGNER, тобто як відмінну від заякореної первинної версії за незмінного підписувача.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У роботі розроблено модель SDMS як багаторівневої системи захищеного документообігу та формальну модель документа у вигляді Document State Chain, де кожна операція формує новий криптографічно зв'язаний стан із доказовими атрибутами. Експериментальні результати підтвердили tamper-evidence властивості моделі та здатність локалізувати перший порушений стан при ретроактивному втручанні (H1). Встановлено монотонне зростання часу повної верифікації зі збільшенням довжини ланцюга станів і узгодженість залежності $t_{verify}(K)$з лінійною апроксимацією в досліджуваному діапазоні (H2). Порівняльний експеримент засвідчив системну перевагу SDMS над централізованим журналюванням: SDMS забезпечує детекцію, локалізацію та незалежну перевірюваність результатів, тоді як baseline-модель є вразливою до "узгоджених" ретроактивних правок (H3). Додатковий прикладний сценарій із двома КЕП-підписаними версіями показав, що SDMS відрізняє тотожність версії документа від самого факту валідного підпису, що підсилює аудитопридатність у практичних процесах.

Подальші дослідження доцільно спрямувати на проєктування референсної архітектури SDMS для впровадження та інтеграції з корпоративними платформами ERP/CRM/ECM. Перспективними є розроблення стандартних інтеграційних адаптерів і подійних механізмів, які відображають бізнес-операції у переходи Document State Chain, а також інтеграція SDMS з корпоративними механізмами управління доступом і сервісами довіри. Окремим напрямом є масштабування експериментів у багатокористувацьких сценаріях і формалізація правил канонізації контенту та доказових атрибутів для забезпечення відтворюваної верифікації у міжорганізаційному документообігу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Han, J., Kim, H., Eom, H., & Son, Y. (2021). A decentralized document management system using blockchain and secret sharing. In *Proceedings of the ACM Symposium on Applied Computing (SAC '21)*. ACM. https://doi.org/10.1145/3412841.3442077

2. Dolhova, N., Shapovalova, O., & Solodovnyk, H. (2025). Systema pidtrymky rishen u zavdanniakh proektuvannia kiberbezpeky krytychnoi infrastruktury [Decision support system in cybersecurity design of critical infrastructure]. *Kiberbezpeka: osvita, nauka, tekhnika*, *1*(29), 348-372. https://doi.org/10.28925/2663-4023.2025.29.884
3. Precht, H., Hüllmann, J. A., & Marx Gómez, J. (2024). Paperless everything: A systematic literature review for the design of blockchain-based document management systems. *Distributed Ledger Technologies: Research and Practice*, *5*(2), Article 19, 1-48. https://doi.org/10.1145/3737296
4. Azzam, F., Jaber, M., Saies, A., Kirresh, T., Awadallah, R., Karakra, A., Barghouthi, H., & Amarneh, S. (2023). The use of blockchain technology and OCR in e-government for document management: Inbound invoice management as an example. *Applied Sciences*, *13*(14), 8463. https://doi.org/10.3390/app13148463
5. Akula, V. S. S. R., Kavarakuntla, T., Kanipakam, S., et al. (2023). Blockchain-backed verification systems for enhanced interoperability and trust in managing legal documents across multi-cloud environments. *Journal of Electrical Systems*, *19*(4), 254-269. https://doi.org/10.52783/jes.637
6. Das, M., Tao, X., Liu, Y., & Cheng, J. C. P. (2022). A blockchain-based integrated document management framework for construction applications. *Automation in Construction*, *133*, 104001. https://doi.org/10.1016/j.autcon.2021.104001
7. Muravskyi, V., Khoma, N., Khokhlova, L., & Liu, C. (2021). Open document flow based on blockchain technology for cybersecurity of the accounting system. *Herald of Economics*, *4*, 156-170. https://doi.org/10.35774/visnyk2021.04.156
8. Hwang, H.-C., & Kim, W.-J. (2022). Design of chained document HTML generation technique based on blockchain for trusted document communication. *Electronics*, *11*(7), 1006. https://doi.org/10.3390/electronics11071006
9. Shekhtman, L., & Waisbard, E. (2021). EngraveChain: A blockchain-based tamper-proof distributed log system. *Future Internet*, *13*(6), 143. https://doi.org/10.3390/fi13060143
10. Chen, C., Wang, L., Long, Y., Luo, Y., & Chen, K. (2022). A blockchain-based dynamic and traceable data integrity verification scheme for smart homes. *Journal of Systems Architecture*, *130*, 102677. https://doi.org/10.1016/j.sysarc.2022.102677
11. Zhang, C., Xuan, H., Wu, T., Liu, X., Yang, G., & Zhu, L. (2024). Blockchain-based dynamic time-encapsulated data auditing for outsourcing storage. *IEEE Transactions on Information Forensics and Security*, *19*, 1979-1993. https://doi.org/10.1109/TIFS.2023.3338485
12. Lu, Z., & Wu, H. (2025). A framework for dynamic blockchain-based data auditing. *International Journal of Accounting Information Systems*, *56*, 100737. https://doi.org/10.1016/j.accinf.2025.100737
13. Yang, D., Wang, Y., Huang, W., & Zhao, Y. (2025). A novel cryptography-based architecture for secure data asset sharing and circulation systems. *Applied Sciences*, *15*(12), 6877. https://doi.org/10.3390/app15126877
14. Kapeliushna, T., Muzhanova, T., Berestiana, T., Holoborodko, S., & Prymachenko, D. (2025). Mekhanizm upravlinnia dostupom do informatsiinykh resursiv na pidpryiemstvi [Access control mechanism for enterprise information resources]. *Kiberbezpeka: osvita, nauka, tekhnika*, *3*(27), 205-219. https://doi.org/10.28925/2663-4023.2025.27.743
15. Poberezhnyk, V., & Nakonechnyi, T. (2025). Doslidzhennia hibrydnoi systemy keruvannia oblikovymy danymy na osnovi poiednannia tekhnolohii DID, IPFS, blockchain [Study of a hybrid identity management system based on DID, IPFS, and blockchain]. *Kiberbezpeka: osvita, nauka, tekhnika*, *3*(31), 441-459. https://doi.org/10.28925/2663-4023.2025.31.1033

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 1 (33), 2026
ISSN 2663 – 4023

Nataliya Dolgova
Ph.D. (Tech.), Associate Professor,
Associate Professor of the Department of Cybersecurity and Information Technology,
Simon Kuznets Kharkiv National University of Economics, Kharkiv, Ukraine
ORCID: 0000-0002-8950-8200
natalya.dolgova@hneu.net

MODEL OF A DOCUMENT MANAGEMENT SYSTEM WITH A CRYPTOGRAPHICALLY VERIFIABLE DOCUMENT STATE CHAIN

Abstract. This paper develops a document management system model intended for environments in which the integrity of document history, control of the document lifecycle, and the possibility of independent verification of performed operations are critically important. The relevance of the study is determined by the fact that traditional electronic document management systems mainly rely on centralized event logs and application logic, which does not eliminate the risks of retrospective modification of document history and a reduction in its evidential value. The aim of the work is to construct a document management system model in which the integrity of document history is ensured through a cryptographically verifiable chain of document states and the recording of evidential event attributes in a permissioned distributed ledger. The proposed model combines architectural and formal levels of system representation. At the architectural level, the user, application, evidential, and content layers are distinguished. At the formal level, a document is represented as a sequence of cryptographically linked states, in which each new state contains the state hash, metadata, timestamp, content hash, and a reference to the previous state, thus ensuring the integrity and traceability of the entire document history. To implement the evidential layer, a smart contract for registering document states and a permissioned distributed ledger based on Hyperledger Besu are used. Experimental validation of the model was carried out on a local testbed using the QBFT consensus mechanism, external storage, and software modules for generating and fully verifying document history. The experimental results confirmed the ability of the model to detect retrospective changes in content, metadata, signatures, and temporal attributes, to localize the first compromised state, and to provide near-linear growth in full verification time as the length of the state chain increases. Comparative evaluation against centralized logging demonstrated the advantage of the proposed approach in terms of tamper detection, localization of violations, and independent verifiability of results. The practical significance of the work lies in the possibility of using the proposed model as a basis for building corporate document management systems.

Keywords: secure document management; blockchain; smart contract; permissioned distributed ledger; Hyperledger Besu; document state chain; cryptographic verification; document integrity verification.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Han, J., Kim, H., Eom, H., & Son, Y. (2021). A decentralized document management system using blockchain and secret sharing. In *Proceedings of the ACM Symposium on Applied Computing (SAC '21)*. ACM. https://doi.org/10.1145/3412841.3442077
2. Dolhova, N., Shapovalova, O., & Solodovnyk, H. (2025). Systema pidtrymky rishen u zavdanniakh proektuvannia kiberbezpeky krytychnoi infrastruktury [Decision support system in cybersecurity design of critical infrastructure]. *Kiberbezpeka: osvita, nauka, tekhnika, 1*(29), 348-372. https://doi.org/10.28925/2663-4023.2025.29.884
3. Precht, H., Hüllmann, J. A., & Marx Gómez, J. (2024). Paperless everything: A systematic literature review for the design of blockchain-based document management systems. *Distributed Ledger Technologies: Research and Practice, 5*(2), Article 19, 1-48. https://doi.org/10.1145/3737296
4. Azzam, F., Jaber, M., Saies, A., Kirresh, T., Awadallah, R., Karakra, A., Barghouthi, H., & Amarneh, S. (2023). The use of blockchain technology and OCR in e-government for document management: Inbound invoice management as an example. *Applied Sciences, 13*(14), 8463. https://doi.org/10.3390/app13148463

5. Akula, V. S. S. R., Kavarakuntla, T., Kanipakam, S., et al. (2023). Blockchain-backed verification systems for enhanced interoperability and trust in managing legal documents across multi-cloud environments. *Journal of Electrical Systems*, *19*(4), 254-269. https://doi.org/10.52783/jes.637
6. Das, M., Tao, X., Liu, Y., & Cheng, J. C. P. (2022). A blockchain-based integrated document management framework for construction applications. *Automation in Construction*, *133*, 104001. https://doi.org/10.1016/j.autcon.2021.104001
7. Muravskyi, V., Khoma, N., Khokhlova, L., & Liu, C. (2021). Open document flow based on blockchain technology for cybersecurity of the accounting system. *Herald of Economics*, *4*, 156-170. https://doi.org/10.35774/visnyk2021.04.156
8. Hwang, H.-C., & Kim, W.-J. (2022). Design of chained document HTML generation technique based on blockchain for trusted document communication. *Electronics*, *11*(7), 1006. https://doi.org/10.3390/electronics11071006
9. Shekhtman, L., & Waisbard, E. (2021). EngraveChain: A blockchain-based tamper-proof distributed log system. *Future Internet*, *13*(6), 143. https://doi.org/10.3390/fi13060143
10. Chen, C., Wang, L., Long, Y., Luo, Y., & Chen, K. (2022). A blockchain-based dynamic and traceable data integrity verification scheme for smart homes. *Journal of Systems Architecture*, *130*, 102677. https://doi.org/10.1016/j.sysarc.2022.102677
11. Zhang, C., Xuan, H., Wu, T., Liu, X., Yang, G., & Zhu, L. (2024). Blockchain-based dynamic time-encapsulated data auditing for outsourcing storage. *IEEE Transactions on Information Forensics and Security*, *19*, 1979-1993. https://doi.org/10.1109/TIFS.2023.3338485
12. Lu, Z., & Wu, H. (2025). A framework for dynamic blockchain-based data auditing. *International Journal of Accounting Information Systems*, *56*, 100737. https://doi.org/10.1016/j.accinf.2025.100737
13. Yang, D., Wang, Y., Huang, W., & Zhao, Y. (2025). A novel cryptography-based architecture for secure data asset sharing and circulation systems. *Applied Sciences*, *15*(12), 6877. https://doi.org/10.3390/app15126877
14. Kapeliushna, T., Muzhanova, T., Berestiana, T., Holoborodko, S., & Prymachenko, D. (2025). Mekhanizm upravlinnia dostupom do informatsiinykh resursiv na pidpryiemstvi [Access control mechanism for enterprise information resources]. *Kiberbezpeka: osvita, nauka, tekhnika*, *3*(27), 205-219. https://doi.org/10.28925/2663-4023.2025.27.743
15. Poberezhnyk, V., & Nakonechnyi, T. (2025). Doslidzhennia hibrydnoi systemy keruvannia oblikovymy danymy na osnovi poiednannia tekhnolohii DID, IPFS, blockchain [Study of a hybrid identity management system based on DID, IPFS, and blockchain]. *Kiberbezpeka: osvita, nauka, tekhnika*, *3*(31), 441-459. https://doi.org/10.28925/2663-4023.2025.31.1033

Отримано редакцією журналу / Received: 04.02.26
Прорецензовано / Revised: 16.02.26
Схвалено до друку / Accepted: 25.06.26