DOI: 10.35681/1560-9189.2025.27.2.345550

УДК 004.9

Ю. Є. Боярінова[1,2], О. П. Мартинова[2], В. Р. Селетков[2]
[1]Інститут проблем реєстрації інформації НАН України
вул. М. Шпака, 2, 03113 Київ, Україна
[2]Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Проспект Берестейський, 37, 03056 Київ, Україна

Спосіб захищеного зберігання приватного ключа акаунту мережі блокчейн

Досліджено способи захищеного зберігання приватних ключів акаунтів у блокчейн-мережах, зокрема, методи симетричного (AES) і асиметричного (RSA) шифрування. Безпечне зберігання приватного ключа є критично важливою задачею, оскільки компрометація ключа може призвести до втрати активів користувача і значних збитків. Акцентовано увагу на вдосконалених підходах, таких як фрагментація ключа, паралельне шифрування та комбіноване використання алгоритмів AES і RSA.

Ключові слова: *блокчейн, приватний ключ, RSA, AES*

Вступ

У сучасному цифровому середовищі безпека даних є одним із ключових викликів, з якими стикаються як окремі користувачі, так і великі організації. Особливо це стосується захисту криптографічних ключів, які використовуються для аутентифікації, шифрування даних і управління цифровими активами. Приватний ключ у блокчейн-мережах є унікальним ідентифікатором користувача, що дозволяє здійснювати операції, підписувати транзакції й отримувати доступ до криптовалютних активів. Втрата або компрометація приватного ключа призводить до незворотних наслідків, оскільки у блокчейн-мережах не існує механізмів відновлення або скасування транзакцій [1–4].

Мета

Метою дослідження є розробка ефективних методів захищеного зберігання приватного ключа в блокчейн-мережах, що забезпечують високий рівень безпеки, стійкість до атак і компрометації, а також збереження продуктивності криптографічних операцій.

© Ю. Є. Боярінова, О. П. Мартинова, В. Р. Селетков

Криптографічні алгоритми для приватного ключа

Розглядаючи можливості захищеного зберігання приватного ключа, необхідно обрати криптографічні алгоритми, які забезпечують високу безпеку, ефективність і відповідність сучасним стандартам. У рамках запропонованого підходу визначаємо, що найкраще для досягнення поставлених цілей підходять два алгоритми: AES для швидкого й ефективного шифрування і RSA для безпечного обміну та додаткового рівня захисту [5].

AES є одним із найбільш популярних алгоритмів симетричного шифрування завдяки високій швидкості роботи, криптографічній стійкості та широкому використанню в індустрії. Симетричне шифрування з фіксованою довжиною ключа (128, 192, 256 біт) забезпечує баланс між продуктивністю та безпекою. Висока швидкість шифрування та дешифрування, що особливо важливо для частого використання в криптографічних операціях. Захищеність від відомих атак, AES стійкий до атак brute-force та диференційного криптоаналізу. Звісно, більшість сучасних процесорів мають апаратні розширення для AES, що підвищує продуктивність і знижує навантаження на систему [6].

У дослідженні AES використовується для основного захисту приватного ключа, шифруючи його перед збереженням у локальному чи віддаленому сховищі. Застосування AES-256 гарантує найвищий рівень криптографічної стійкості, що унеможливлює його злам навіть при використанні потужних обчислювальних ресурсів.

Алгоритм RSA є одним із найбільш поширених алгоритмів асиметричного шифрування, який використовується для безпечного зберігання та передачі секретної інформації [7].

Асиметричний принцип (пара відкритого та закритого ключа) дозволяє реалізувати безпечний механізм доступу. Стійкість до квантових атак поки що залишається високою (при використанні ключів 2048 біт і більше). RSA дозволяє забезпечити автентичність і невід'ємність даних. Алгоритм має стійкість до багатьох видів атак за умови використання ключів великої довжини. У нашій системі RSA використовується як другий рівень захисту, доповнюючи AES.

Зокрема, RSA застосовується для шифрування самого AES-ключа, що дозволяє зберігати ключові дані в захищеній формі та безпечно передавати їх між різними компонентами системи.

Застосування лише симетричного або асиметричного шифрування має свої недоліки. Наприклад, AES ефективний для шифрування даних, але потребує безпечного способу зберігання ключа, тоді як RSA, хоч і забезпечує асиметричний захист, значно повільніший при обробці великих обсягів даних.

Поєднання AES та RSA дозволяє створити багаторівневу систему захисту для зберігання приватного ключа. В цьому випадку AES використовується для основного шифрування ключа, гарантуючи швидке шифрування та дешифрування. RSA шифрує AES-ключ, запобігаючи його компрометації. Навіть у випадку компрометації збережених даних без закритого ключа RSA зловмисник не зможе розшифрувати AES-ключ. Дана комбінація дозволяє інтегрувати різні рівні безпеки залежно від конкретної реалізації системи.

Аналізуючи криптографічні алгоритми, можна прийти до висновку, що найкращим варіантом для захищеного зберігання приватного ключа є комбіноване

використання AES та RSA. AES-256 забезпечує швидке та стійке до зламування шифрування ключа, а RSA-2048/4096 використовується для захисту ключа AES, додаючи додатковий рівень безпеки [8]. Такий підхід поєднує продуктивність і криптографічну надійність, що робить його оптимальним для практичного застосування у блокчейн-екосистемах.

Способи фрагментації приватного ключа

Фрагментація приватного ключа є одним із методів підвищення безпеки його зберігання, що полягає в розподілі ключа на кілька частин з подальшим їхнім окремим збереженням у різних сховищах, пристроях або у різних власників. Це дозволяє мінімізувати ризик компрометації, оскільки навіть у випадку витоку окремої частини відновити повний ключ буде неможливо. Даний підхід однозначно являється особливо актуальним у контексті захисту блокчейн-гаманців, децентралізованих фінансів і корпоративних систем, що оперують криптовалютними активами [9].

Метод простого поділу на частини (Split Key Storage) передбачає механічний поділ приватного ключа на декілька рівнозначних частин, які зберігаються окремо. Відновлення можливе лише за наявності всіх частин. Такий підхід простий у реалізації і має мінімальні обчислювальні витрати. В той же час присутній високий ризик втрати доступу (якщо хоча б одна частина буде втрачена, ключ неможливо відновити). Також він вразливий перед атакою на всі місця зберігання.

Схема порогового розподілу Shamir`s Secret Sharing (SSS) являє собою один із найпоширеніших криптографічних методів, який дозволяє поділити ключ на n частин, причому для його відновлення достатньо отримати лише k із n. Тобто, можна задати рівень необхідного кворуму (наприклад, схема (3,5) означає, що потрібно зібрати три частини з п’яти. Такий спосіб стійкий до втрати частини ключів і має захист від атаки з підбором частин (фрагменти окремо не несуть корисної інформації). Але це вимагає високих обчислювальних витрат на розподіл і відновлення, присутня необхідність узгодження серед власників частин.

Метод гібридної фрагментації (поєднання SKS та SSS) має на увазі методику, що поєднує поділ із додатковим рівнем захисту, таким як AES або RSA-шифрування перед поділом. У цьому випадку, додатковий рівень безпеки, навіть якщо частина фрагментів буде викрадена. Доступна захищеність від атак на відкритому каналі передачі даних. Даний підхід потребує управління криптографічними ключами для шифрування і вимагає значних ресурсів.

Метод динамічної фрагментації з періодичною ротацією використовує регулярну зміну фрагментів і місць їхнього зберігання, що ускладнює компрометацію. В результаті можна отримати зменшення ризику атаки «відкладеного збору» (зловмисник не може поступово зібрати фрагменти). Дана система є досить гнучкою у керуванні безпекою. Такий спосіб має підвищену складність адміністрування й додаткові витрати на обчислення та синхронізацію.

Різні підходи до фрагментації застосовуються залежно від вимог безпеки системи. Гаманці з мультипідписом (Multisig Wallets) використовують SSS або інші методи для розподілу ключів між кількома учасниками. Великі фінансові установи комбінують шифрування та пороговий розподіл для безпечного зберігання ключів. DeFi використовує динамічне фрагментоване зберігання з оновленням ключів для зниження ризиків атак.

Окрім самого процесу поділу, важливим є захист фрагментів від компрометації. Необхідно зосередитися на захищеному зберіганні, а саме: використання HSM або захищених елементів у мобільних пристроях, хеш-функцій для перевірки цілісності кожного фрагмента перед об'єднанням, двофакторної аутентифікації, де перед об'єднанням фрагментів користувач має пройти додаткову перевірку особи. Вибір конкретного методу залежить від вимог до безпеки та рівня зручності, що необхідний користувачам.

Методологія поділу ключа на фрагменти

Поділ приватного ключа на фрагменти базується на концепції підвищення безпеки за рахунок розділення єдиного секретного значення на частини, які окремо не містять корисної інформації, але можуть бути відновлені в потрібній комбінації. Основна мета такого підходу — запобігання несанкціонованому доступу до приватного ключа шляхом мінімізації ризику його компрометації у разі злому чи втрати даних.

Виконання поділу приватного ключа повинно відповідати наведеним нижче критеріям: неможливість відновлення за частковим доступом, гарантоване відновлення при заданій кількості, криптографічна стійкість, гнучкість і масштабованість.

Метод лінійного поділу (простий розріз ключа) передбачає механічний поділ бітової послідовності приватного ключа на рівні частини.

1. Якщо K — це приватний ключ довжиною L біт.

Поділяємо його на n рівних частин, кожна з яких має розмір L/n біт.

Тоді, якщо всі частини не зібрані разом, ключ неможливо відновити. Втрата хоча б одного фрагмента робить ключ недоступним. Недостатня криптографічна стійкість, оскільки фрагменти містять частини реального ключа.

2. Використання XOR (бітовий поділ із секретом).

Один із простих способів, але ефективних підходів, що базується на операції XOR (бітовий поділ із секретом).

Генеруємо випадковий ключ S того ж розміру, що і K.

$$F_1 = K \oplus S. \tag{1}$$

Зберігаємо S окремо від F_1. Використовуємо наступний вираз для відновлення ключа:

$$K = F_1 \oplus S. \tag{2}$$

Коли один із фрагментів загублений, відновлення неможливе. Отримуємо мінімальні обчислювальні витрати. Але маємо лише дві частини, що обмежує використання в складних системах, і збереження S має бути максимально безпечним.

Якщо застосовувати гібридний спосіб AES + SSS, який поєднує переваги порогового розподілу та симетричного шифрування, то можна отримати наступний алгоритм.

1. Шифруємо приватний ключ за допомогою AES-256.
2. Отриманий шифр текст поділяємо за схемою SSS.
3. Кожен фрагмент передається окремо різним сторонам.
4. Для відновлення потрібно зібрати достатню кількість фрагментів, відновити за SSS, а потім розшифрувати AES.

У такому випадку, можна отримати додатковий рівень безпеки через використання симетричного шифрування і стійкість до атак при витоку значної частини фрагментів. Як завжди, мінусом є висока обчислювальна складність і ускладнене управління ключами для AES.

При виборі конкретного методу слід враховувати наведені нижче аспекти. Порогові методи забезпечують найкращий баланс між стійкістю та гнучкістю. XOR-метод або простий поділ підходять для невеликих систем. Схеми SSS дозволяють часткову втрату фрагментів без втрати ключа. Алгоритм AES + SSS вимагає значних ресурсів, тоді як XOR-метод швидший.

Найбільш ефективним підходом у більшості випадків є використання SSS, оскільки він дозволяє налаштовувати порогові параметри та запобігає компрометації навіть при частковій втраті даних. Використання AES у комбінації із SSS забезпечує додатковий рівень криптографічного захисту. Вибір конкретного методу залежить від вимог до рівня безпеки, продуктивності та можливостей управління ключами.

Використання секретного розподілу

Секретний розподіл є фундаментальним методом захисту конфіденційних даних, зокрема приватних ключів у блокчейні. Метод SSS ґрунтується на принципах поліноміальної інтерполяції і дозволяє розподіляти секретні дані між кількома учасниками таким чином, щоб їхнє відновлення стало можливим лише при зборі певної кількості часток. Цей підхід широко застосовується у сфері захисту блокчейн-гаманців, децентралізованих фінансів, корпоративного управління криптографічними секретами та інших критичних галузях.

Алгоритм SSS дозволяє поділити секретний ключ на n фрагментів, причому для його відновлення необхідно володіти мінімум k частками (де $k \leq n$).

Такий механізм забезпечує стійкість до атак і компрометації окремих фрагментів, оскільки частини, кількість яких менша за порогове значення k, не дають жодної інформації про початковий секрет.

Криптографічна основа SSS базується на поліноміальній інтерполяції Лагранжа. Нехай маємо секретне число K (наприклад, приватний ключ). Для його поділу обирається випадковий поліном $k-1$ ступеня.

Опишемо наступне обчислення SSS таким чином, що є одним із найбільш безпечних і поширених методів. Генеруємо поліном ступеня $k-1$ вигляду:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}, \quad (3)$$

де $a_0 = K$ (приватний ключ), а коефіцієнти $a_1, a_2, \ldots a_{k-1}$ вибираються випадковим чином.

Визначаємо n унікальних точок полінома $(x_i, f(x_i))$, де x_i — унікальні значення, а $f(x_i))$ — їхні відповідні значення. Кожна із цих точок виступає окремим фрагментом секрету.

Відновлення секрету відбувається за допомогою інтерполяції Лагранжа, що дозволяє відновити поліном, а відповідно і його вільний член $a_0 = K$, знаючи хоча би k точок. Таким чином, навіть якщо частина часток буде втрачена, ключ можна відновити, що критично важливо для безпеки.

Алгоритм SSS надає захист приватних ключів у криптогаманцях шляхом поділу ключа між різними пристроями або довіреними особами. Застосовується він у децентралізованому управлінні ключами в інституційних криптосистемах, що використовують мультипідписні технології. Також забезпечує безпечне відновлення доступу до ключів у випадку втрати пароля або зламу сховища. Метод застосовується у системах голосування на блокчейні, де необхідно розподіляти довіру між різними вузлами для підпису транзакцій. Наприклад, криптогаманці MetaMask і Trust Wallet не підтримують метод SSS на рівні користувача, однак існують сторонні рішення, які дозволяють реалізовувати цей підхід для покращення безпеки. Тож даний алгоритм має високий криптографічний захист, у ньому відсутні єдині точки компрометації, гнучкий в управлінні, є можливість використання в децентралізованих середовищах. Але SSS складний у розподілі та зберіганні часток, існує ризик незворотної втрати і він обмежений у масштабованості.

Криптографічні порогові схеми фрагментації

Фрагментація приватного ключа є ефективним методом підвищення безпеки зберігання конфіденційних даних у блокчейн-екосистемах [10]. Одним із найбільш потужних підходів у цій сфері є криптографічні порогові схеми, які дозволяють поділити секрет між декількома учасниками таким чином, щоби його можна було відновити лише за умови наявності певної кількості частин. Порогові криптографічні методи забезпечують стійкість до атак, втрати частини ключа та підвищують рівень безпеки блокчейн-гаманців і децентралізованих систем [11, 12].

У загальному вигляді схема Threshold Signature Scheme (TSS) передбачає, що секрет S поділяється на n частин, причому для його відновлення необхідно мати принаймні k з них, де $k \leq n$.

Якщо всі n частин необхідні для відновлення секрету, схема є максимально захищеною, але вразливою до втрати навіть однієї частини. Якщо $k < n$, то система стає більш гнучкою: навіть якщо деякі частини будуть втрачені, ключ залишиться відновлюваним. Однією із найбільш відомих схем у цій категорії є SSS, яка була детально розглянута в попередньому розділі.

Проте, крім SSS, існують й інші порогові методи, які застосовуються у сфері блокчейн-безпеки.

Одним із важливих удосконалень класичної схеми SSS, є схема із верифікацією часток Verifiable Secret Sharing (VSS), яка додає механізм перевірки цілісності кожного фрагмента. Так можна захищатися від зловмисних учасників, які можуть надавати неправильні частки, запобігати компрометації секрету через підміну фрагментів і використовувати порогову схему в середовищах з обмеженою довірою.

У традиційних блокчейн-гаманцях приватний ключ підписує транзакції самостійно, що створює ризик компрометації [13]. Порогові цифрові підписи (TSS) дозволяють розподілити процес підпису між кількома сторонами без розкриття приватного ключа. Це працює наступним описаним чином. Приватний ключ ділиться між n учасниками. Для підпису транзакції необхідно об'єднати щонайменше k частин. Жоден із учасників не може відновити ключ самостійно, що унеможливлює злам. TSS використовується в таких продуктах як Binance Custody, Fireblocks і ZenGo, дозволяючи реалізувати децентралізоване управління ключами без потреби в мультисигнатурних транзакціях (MultiSig).

На відміну від класичного шифрування, де для розшифрування потрібен повний приватний ключ, Threshold Encryption (TE) дозволяє розшифрувати повідомлення лише за наявності певної кількості часток. Тобто, метод корисний для безпечного зберігання приватних ключів на розподілених серверах, що запобігає єдиній точці компрометації, для захисту даних у децентралізованих сховищах, таких як IPFS або Arweave, і для обмеженого доступу до критичних даних, наприклад, у системах управління цифровими активами.

Наведемо кілька прикладів реального застосування порогових схем.

1. MetaMask — підтримує використання порогових цифрових підписів (TSS) для спільного управління криптоактивами. Приватний ключ ніколи не існує повністю. Його частки генеруються та зберігаються на пристроях різних сторін (наприклад, користувач — бекенд, MetaMask — провайдер безпеки). Підпис формується колективно, без відновлення ключа. Децентралізоване управління доступом без єдиної точки компрометації.

2. Fireblocks і Coinbase — застосовують порогові технології для корпоративного зберігання приватних ключів. У Fireblocks частки ключа розподілені між серверними HSM Fireblocks, користувацькими девайсами та внутрішніми сервісами. Для створення підпису виконується MPC-протокол, без збирання ключа в одне місце. Корпоративне рішення для фінансових інституцій, підтримує багаторівневу авторизацію. Для Coinbase ключ генерується спільно між Coinbase і клієнтом; обидві сторони зберігають свої частки. Транзакція можлива лише за взаємної участі. Централізована кастодіальна модель, але без єдиного ключового файлу.

3. SSS у Polkadot та Ethereum — використовується для децентралізованого управління стейкінгом. У Polkadot ключ валідатора ділиться на n частин, які зберігаються у різних вузлах або учасників. Для відновлення ключа потрібні k часток. Зручно для відновлення доступу або децентралізованого стейкінгу, але ключ з'являється при відновленні. У Ethereum ключі розділені між кількома стейкерами або керівниками DAO. Використовується SSS або звичайна багатопідписна модель. Забезпечує колективне управління, але з ризиком компрометації при об'єднанні часток.

4. Chainlink — інтегрує порогові підписи у свою систему оракулів. Ключ оракулів для підпису даних розподілений між вузлами. Кожен вузол обчислює частковий підпис, який агрегується у спільний результат. Децентралізована модель без єдиного центру довіри; стійка до компрометації одного вузла.

Отже, на основі проведеного аналізу можна зробити висновок, що найбільш ефективним підходом до захищеного зберігання приватного ключа є комбіноване використання симетричного і асиметричного шифрування, зокрема алгоритмів AES-256 і RSA-2048/4096. AES забезпечує швидке та стійке до зламування шифрування ключа, тоді як RSA додає додатковий рівень захисту при передачі або зберіганні шифрувального ключа. Для підвищення надійності доцільно застосовувати метод SSS, який дозволяє розділити ключ на декілька частин і зберігати їх у різних захищених сховищах. Це суттєво знижує ризик компрометації при втраті одного із фрагментів. Такий багаторівневий підхід поєднує продуктивність, криптографічну стійкість і гнучкість, а також відповідає сучасним вимогам до безпеки у блокчейн-системах, включаючи перспективи впровадження постквантових алгоритмів.

Для тестування продуктивності часу використано 2048-бітовий RSA-ключ у поєднанні із симетричним алгоритмом AES-256 для шифрування, а також RSA-

4096 для асиметричного захисту, результати наведено в таблиці. Дослідження проводилося на ПК з параметрами: процесор AMD Ryzen 9 7950X (16C/32T, Zen 4 з AVX-512, AES-NI/VAES), RAM 64 GB DDR5-6000, диск NVMe SSD PCIe 4.0 (1 TB), операційна система Ubuntu 24.04 LTS. Аналіз показав, що використання паралельної обробки дозволяє досягти найкращих результатів. Зокрема, час шифрування та розшифрування зменшується на 58 % порівняно з комбінованим методом і на 39 % порівняно із методом фрагментації.

Час шифрування та розшифрування для різних методів

Метод	Шифрування час (мс)	Розшифрування час (мс)
Фрагментація ключа (6 частин, AES-256)	1,38	1,56
Комбінований метод (AES-256 + RSA-4096)	2,01	2,04
Паралельна обробка фрагментів (8 потоків, AES-256 + RSA-4096)	0,84	0,92

Висновки

У межах дослідження реалізовано схему подвійного шифрування приватного ключа, що поєднує переваги симетричного і асиметричного шифрування. Такий підхід значно підвищує рівень захисту конфіденційної інформації навіть у разі компрометації одного із компонентів. На першому етапі приватний ключ користувача шифрується симетричним AES-алгоритмом із випадково згенерованим ключем. На другому етапі він шифрується відкритим ключем RSA. Це забезпечує розділення логіки доступу: без розшифрування AES-ключа неможливо отримати сам приватний ключ, а розшифрування можливе лише за наявності приватного RSA-ключа.

Симетричне шифрування AES забезпечує високу швидкодію та мінімальні затримки при роботі з великими обсягами даних. Водночас асиметричний алгоритм RSA дозволяє безпечно передавати ключі, навіть через незахищені канали зв'язку, оскільки для розшифрування потрібен лише приватний ключ одержувача. Для прикладу, результат подвійного шифрування можна зберігати у форматі JSON. Такий підхід дозволяє зберігати метаінформацію разом із ключами, а також легко інтегрувати дані у системи безпечного зберігання.

Основна практична цінність дослідження полягає у можливості створення безпечних криптогаманців і систем керування цифровими активами, у яких приватні ключі користувачів залишаються захищеними навіть у разі компрометації одного із компонентів системи. Застосування фрагментації ключа, паралельного шифрування та комбінованого використання AES і RSA дозволяє мінімізувати ризики несанкціонованого доступу, а також підвищити швидкодію операцій шифрування та розшифрування.

Запропонований спосіб може бути впроваджений у корпоративних блокчейн-системах (DeFi, DAO, біржових платформах) для забезпечення спільного управління криптоактивами за допомогою порогових підписів або розподілених моделей зберігання ключів. Крім того він може бути застосований для захищеного резервного копіювання та відновлення ключів у хмарних або гібридних інфраструктурах.

Отже, цей спосіб може бути використаний як базова технологія для створення нових поколінь безпечних децентралізованих застосунків (DApp), систем елект-

ронної ідентифікації і зберігання цифрових активів, що відповідають сучасним вимогам до конфіденційності та цілісності інформації.

1. Bashir I. Mastering Blockchain: distributed ledger technology, decentralization, and smart contracts explained. 2-nd ed. Birmingham: Packt Publishing, 2018.

2. Тищенко О.С., Гумен Т.Ф., Трапезон К.О. Дослідження особливостей технології Blockchain в інформаційних системах передавання даних. *Вчені записки Таврійського національного університету*. 2019. Т. 30(69), № 2. Ч. 1. С. 77–81.

3. Метєлєв Д.І. Основи технології блокчейн. Київ: Вид-во Ліра-К, 2020. 236 с.

4. Fang X., Misra S., Xue G. Smart grid — the new and improved power grid: a survey. IEEE Communications Surveys & Tutorials. 2012. Vol. 14, No. 4. P. 944–980.

5. Mmasmoudi M. An Overview of MPC, Threshold Signatures, TSS, and Wallet Security. 2022. Medium. URL: https://mmasmoudi.medium.com/an-overview-of-multi-party-computation-mpc-threshold-signatures-tss-and-mpc-tss-wallets-4253adacd1b2

6. Mollin R. RSA and Public-Key Cryptography (Discrete Mathematics and Its Applications). United States, Boca Raton, Fla.: Chapman and Hall/CRC; 1st edition (November 12, 2002), 2002. 304 p.

7. AES vs RSA Encryption: What Are the Differences? Precisely. 2022. URL: https://www.precisely.com/blog/data-security/aes-vs-rsa-encryption-differences

8. Алгоритм шифрування RSA, види атак на нього. Реалізація мовою Python: URL: https://dou.ua/forums/topic/43026/

9. Narayanan Arvind, Bonneau Joseph, Felten Edward, Miller Andrew, Goldfeder Steven. Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction. Princeton University Press, July 19, 2016. 336 p.

10. Ferguson N., Schneier B., Kohno T. Cryptography Engineering: design principles and practical applications. New York: Wiley, 2010. ISBN 978-0470474242.

11. Shamir's Secret Sharing: Explanation and Visualization / Evervault. 2022. URL: https://evervault.com/blog/shamir-secret-sharing

12. A Beginner's Guide to Shamir's Secret Sharing. Medium. 2020. URL: https://medium.com/@keylesstech/a-beginners-guide-to-shamir-s-secret-sharing-e864efbf3648

13. Kravitz D.W., Cooper J. Securing user identity and transactions symbiotically: IoT meets blockchain. 2017 Global Internet of Things Summit (GIoTS). 2017.

Надійшла до редакції 14.05.2025