



DOI 10.28925/2663-4023.2026.35.1310

УДК 004.056:004.436

**Жебка Вікторія Вікторівна**

доктор технічних наук, професор, завідувач кафедри технологій цифрового розвитку  
Державний університет інформаційно-комунікаційних технологій, Київ, Україна  
ORCID: 0000-0003-4051-1190  
[v.zhebka@duikt.edu.ua](mailto:v.zhebka@duikt.edu.ua)

**Аронов Андрій Олексійович**

кандидат технічних наук, доцент кафедри технологій цифрового розвитку  
Державний університет інформаційно-комунікаційних технологій, Київ, Україна  
ORCID: 0009-0000-7868-8341  
[a.aronov@duikt.edu.ua](mailto:a.aronov@duikt.edu.ua)

**Герцюк Микола Модестович**

доктор філософії, доцент кафедри технологій цифрового розвитку  
Державний університет інформаційно-комунікаційних технологій, Київ, Україна  
ORCID: 0000-0003-2946-9673  
[m.gertsuik@duikt.edu.ua](mailto:m.gertsuik@duikt.edu.ua)

**Ганенко Людмила Дмитрівна**

доктор філософії, старший викладач кафедри технологій цифрового розвитку  
Державний університет інформаційно-комунікаційних технологій, Київ, Україна  
ORCID: 0000-0003-2219-8196  
[hanenkoliudmyla@gmail.com](mailto:hanenkoliudmyla@gmail.com)

**МЕТОД БЕЗПЕЧНОГО ПРОГРАМУВАННЯ ДЕЦЕНТРАЛІЗОВАНИХ ДОДАТКІВ НА ОСНОВІ  
КОМПОЗИЦІЙНОЇ ВАЛІДАЦІЇ ПІДПИСАНИХ ОПЕРАЦІЙ**

**Анотація.** У статті досліджено авторизацію підписаних операцій у децентралізованих додатках на основі смартконтрактів. Коректний цифровий підпис засвідчує походження повідомлення, однак не визначає допустимості його виконання після зміни стану, повноважень або контексту. Мета роботи - формалізувати композиційний метод допуску операцій та оцінити співвідношення блокування атак, помилкових відмов і обчислювальних витрат. Дослідження поєднує аналіз наукових публікацій, формальне моделювання переходів стану та описове оцінювання агрегованих модельних даних. Запропоновано специфікацію, яка об'єднує перевірку підпису, домену, попсе, строку дії, актуальної ролі, предметних лімітів та інваріантів. Встановлено умови одноразового успішного виконання дозволу в межах одного домену й узгодженої історії стану. Визначено порядок перевірок та оновлення стану до зовнішньої взаємодії. Модельний набір охоплює сім класів атак і легітимні запити; внесок компонентів проаналізовано через їх почергове вилучення. Повна конфігурація забезпечує частку блокування 99,26 % проти 25,57 % для простого підпису. Частка помилкових відмов зростає з 0,70 % до 1,25 %, а середні модельні витрати - на 31,02 %. Вилучення домену, попсе та актуальної ролі супроводжується вираженими втратами блокування відповідних класів атак. Показано, що загальна ефективність залежить від складу набору, а високий середній показник не усуває потреби аналізувати залишкові помилки. Результати є описовими: відсутність коду генератора й первинних журналів обмежує перевірку причинних зв'язків та статистичної значущості. Окрему увагу приділено доступності системи для легітимних користувачів. Практична цінність полягає в узгодженій специфікації авторизації та сценаріях її тестування. Подальша робота передбачає реалізацію в EVM, аналіз пропущених атак і публікацію відтворених випробувань.

**Ключові слова:** децентралізований додаток; смартконтракт; цифровий підпис; авторизація; попсе; повторний вхід; інваріант стану; криптографія; інформаційні технології.

**ВСТУП****Постановка проблеми**

Децентралізовані додатки поєднують користувацький інтерфейс, засоби підписання повідомлень і програмну логіку смартконтрактів. У схемах із передаванням підписаного дозволу через посередника особа, яка надсилає транзакцію, може відрізнятись від власника повноважень. Тому перевірка `msg.sender` не є універсальним способом встановлення суб'єкта авторизації. Водночас відновлення адреси з цифрового підпису визначає лише походження повідомлення, а не правомірність операції в момент її виконання.

Дозвіл може бути повторно використаний, перенесений до іншого контракту або мережі, виконаний після завершення строку дії чи відкликання ролі. Окремий ризик створює зовнішній виклик, під час якого атакуювальний контракт повторно звертається до функцій системи до завершення узгодженого оновлення стану. Відтак необхідна явна специфікація зв'язку між автентифікацією, авторизацією та переходом стану. Це завдання має практичне значення для делегованих платежів, керування цифровими активами й операцій із доступом за ролями.

**Аналіз останніх досліджень і публікацій**

Atzei, Bartoletti та Cimoli систематизували типові помилки програмування Ethereum і пов'язані з ними атаки [1]. Їхня таксономія допомагає відокремити помилки логіки контракту від особливостей платформи. Liu та співавтори представили Oyente і показали застосовність символічного виконання до пошуку небезпечних шляхів виконання [2]. Для цього дослідження ці роботи є підставою розглядати безпеку як властивість поведінки програми, а не лише застосованого криптографічного алгоритму.

Tsankov та співавтори у Securify використали семантичні факти й шаблони відповідності та порушення для автоматизованого аналізу контрактів [3]. Feist, Grieco та Groce описали Slither із проміжним представленням, яке підтримує аналіз потоків даних і перевірку програмного коду [4]. Обидва підходи придатні для перевірки реалізації, однак коректність прикладної політики залежить від того, чи сформулював розробник необхідні властивості доступу та стану.

Іншим поширеним підходом є динамічний аналіз поведінки смартконтрактів. У [5] запропоновано ContractFuzzer, який використовує генерацію вхідних даних відповідно до ABI та спеціалізовані оракули для виявлення вразливостей під час виконання контракту. Подальший розвиток цього напрямку представлено у [6], де символічне виконання поєднано з навчанням політики фазингу. Такі методи дозволяють досліджувати поведінку контракту для різних послідовностей вхідних даних та виявляти небезпечні сценарії виконання. Проте для операцій, авторизованих цифровим підписом, перевірки окремого виклику недостатньо, оскільки допустимість операції може змінитися в проміжку між формуванням підпису та фактичним виконанням транзакції. Зокрема, можливими є повторне використання раніше підписаного дозволу, його застосування в іншому контексті або виконання після зміни повноважень підписанта.

Результати порівняльного дослідження автоматизованих засобів аналізу Ethereum-контрактів [7] засвідчили, що ефективність виявлення вразливостей залежить як від застосованого інструмента, так і від класу досліджуваних дефектів. У роботах [8, 9] систематизовано основні платформні та інженерні проблеми смартконтрактів, класи атак і відповідні механізми протидії. Отримані результати формують методологічну основу для комплексного аналізу захищеності децентралізованих застосунків, однак не вирішують окремого завдання визначення допустимості конкретної підписаної операції з урахуванням поточного стану смартконтракту.

Важливим для вирішення зазначеної проблеми є підхід, заснований на представленні поведінки смартконтракту через стани та допустимі переходи між ними. У [10] запропоновано методологію FSolidM, у межах якої Ethereum-контракти моделюються як скінченні автомати із застосуванням захисних шаблонів. Такий підхід дає можливість пов'язати виконання операції з поточним станом системи та визначити умови допустимості переходу. Водночас у випадку підписаних операцій виникає додаткова необхідність узгодження параметрів дозволу, криптографічної автентифікації, контексту виконання та актуального стану смартконтракту.

Суміжні аспекти захисту розподілених середовищ розглянуто у працях авторів цієї статті. В. Жебка [11] запропонувала архітектуру моніторингу з використанням `permissioned blockchain` для перевірки достовірності телеметрії та простежуваності керуючих рішень. А. Аронов [12] поєднав графове подання блокчейн-мережі, алгоритм Isolation Forest і адаптивне оцінювання довіри для виявлення аномальної поведінки вузлів. Ці підходи стосуються довіри до даних і моніторингу загроз на рівні мережі. У цій статті увагу зосереджено на іншому рівні захисту: перевірці допустимості конкретної підписаної операції до зміни стану смартконтракту.

Проведений аналіз показав, що існуючі дослідження переважно зосереджені на виявленні



вразливостей програмної реалізації, статичному та динамічному аналізі смартконтрактів, а також моделюванні їх поведінки, тоді як питання формування комплексної умови допуску підписаної операції потребує подальшого розвитку. Коректність цифрового підпису підтверджує автентичність підписаного повідомлення, однак сама по собі не гарантує допустимості його виконання в поточному стані системи. Залишаються актуальними задачі запобігання повторному використанню дозволу, його перенесенню до іншого контракту або мережі, виконанню після завершення строку дії чи відкликання повноважень, а також забезпечення коректності переходу стану під час зовнішніх взаємодій. Саме тому актуальним є розроблення методу, в якому рішення про виконання підписаної операції формується на основі композиційної перевірки автентичності, контексту виконання, одноразовості дозволу, актуальних повноважень, предметних обмежень та інваріантів стану.

#### Мета статті.

Мета статті - формалізувати композиційний метод допуску підписаних операцій і оцінити співвідношення блокування атак, помилкових відмов та обчислювальних витрат за модельними даними. Для цього визначено модель загроз, побудовано функцію допуску й умови переходу стану, сформовано сценарії перевірки та проаналізовано повну й скорочені конфігурації захисту. Внесок роботи полягає в поєднанні цих елементів у специфікацію, придатну для подальшої реалізації та перевірки; окремі криптографічні механізми не розглядаються як нові.

### ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

Об'єктом дослідження є виконання підписаних дозволів у смартконтракті EVM. Предметом є умови допуску й порядок оновлення стану. Автентифікацією називаємо перевірку зв'язку повідомлення з підписувачем; авторизацією - встановлення права виконати саме цю операцію в актуальному стані. Роль - збережена в контракті характеристика повноважень, а попси - лічильник одноразових дозволів для визначеного підписувача.

Модель передбачає надійність криптографічних примітивів, коректне типізоване кодування повідомлень і атомарне виконання транзакцій. Атакувальник спостерігає дозволи, повторно надсилає їх, змінює параметри, використовує інший контекст або реалізує зворотний виклик. Компрометація приватного ключа, зловмисне адміністрування, підміна оракула, зміна консенсусу та економічні атаки не покриваються цією моделлю.

Для структурованого підпису використовується схема EIP-712. Вона визначає типізоване хешування та розділення доменів, але не забезпечує одноразовість дозволу автоматично. Обмеження строку дії також не замінює попси: один дозвіл інакше може бути виконаний багато разів у межах допустимого часового інтервалу. Поля домену повинні формуватися відповідно до фактичної мережі та контракту, а не довільно задаватися неперевіраним викликачем.

Для зовнішнього облікового запису застосовується перевірка ECDSA з відхиленням некоректних і неканонічних підписів. Для контрактного гаманця необхідний окремий шлях валідації на кшталт ERC-1271. Його результат може залежати від поточного стану гаманця, тому не можна вважати колись перевірений підпис назавжди чинним. У цій статті основна формальна властивість розглядається для однозначно визначеного суб'єкта и.

### МЕТОДИКА ДОСЛІДЖЕННЯ

Дослідження має два складники: формальну специфікацію авторизації та описове оцінювання модельного набору. Числовою основою є агрегати супровідного матеріалу «Результати дослідження безпечне програмування»: по 1000 атакуювальних випробувань для семи класів і 2000 легітимних запитів на конфігурацію. Код контрактів, генератор, початкове значення генератора випадкових чисел і журнали випробувань недоступні. Тому аналіз охоплює перерахунок показників та їхню інтерпретацію, а не незалежне відтворення експерименту в EVM.

Порівнюються конфігурації A - контроль викликача msg.sender; B - простий підпис; C - повна композиція. Конфігурації C-N, C-D, C-R та C-I передбачають вилучення перевірок попси, домену, актуальної ролі та інваріантів відповідно. Такий дизайн дає змогу зіставити повну систему з варіантами, у яких відсутня одна захисна умова. Оскільки реалізації A і B не описані на рівні коду, висновки про них обмежено цим модельним набором.

Протокол оцінювання розділяє легітимні та атакуювальні запити за чинною політикою доступу. Для перевірки окремого механізму інші передумови операції мають виконуватися; інакше причина відмови неоднозначна. Таблиця 1 задає сценарії та очікувані властивості майбутньої інструментальної перевірки. Відповідність наявної симуляції цим умовам потребує перевірки за первинними журналами.



Таблиця 1

Сценарії перевірки підписаних дозволів

| Клас               | Зміна або послідовність                   | Очікувана властивість             |
|--------------------|-------------------------------------------|-----------------------------------|
| Replay             | Повторний запит після успішного виконання | Nonce уже спожито                 |
| Міжконтрактний     | Той самий дозвіл для іншого контракту     | Інший домен підпису               |
| Міжмережевий       | Той самий дозвіл за іншого chainId        | Інший домен підпису               |
| Підміна параметрів | Зміна суми, одержувача або дії            | Підпис не відповідає повідомленню |
| Прострочення       | Час блока перевищує deadline              | Дозвіл відхилено                  |
| Відкликання ролі   | Підписання – відкликання – виконання      | Перевіряється актуальна роль      |
| Повторний вхід     | Зовнішній виклик – повторне звернення     | Захищено весь критичний шлях      |

Набір запитів  $\Omega$  поділено на неперетинні множини легітимних  $\Omega_L$  та атакуювальних  $\Omega_A$  запитів. Атакуювальна частина охоплює сім класів:

$$\Omega = \Omega_L \cup \Omega_A, \quad \Omega_L \cap \Omega_A = \emptyset \quad (1)$$

$$\Omega_A = \{\Omega_{rep}, \Omega_{cc}, \Omega_{ch}, \Omega_{par}, \Omega_{exp}, \Omega_{role}, \Omega_{re}\} \quad (2)$$

Індекси rep, cc, ch, par, exp, role та re позначають replay, міжконтрактне й міжмережеве відтворення, підміну параметрів, прострочення, відкликання ролі та повторний вхід. Позитивним класом є атака: TP - заблоковані атаки, FN - пропущені; FP - помилково відхилені легітимні запити, TN - правильно прийняті. Для оцінювання використовуються частка блокування TPR, частка помилкових відмов FPR, точність позитивного рішення Precision та  $F_1$ :

$$TPR = \frac{TP}{TP + FN} \quad (3)$$

$$FPR = \frac{FP}{FP + TN} \quad (4)$$

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

$$F_1 = \frac{2 \cdot Precision \cdot TPR}{Precision + TPR} \quad (6)$$

$$TP + FN = 7000, \quad FP + TN = 2000 \quad (7)$$

TPR характеризує блокування атак, а FPR - вплив перевірок на легітимні операції. Precision і  $F_1$  залежать також від частоти атак: співвідношення 7000:2000 у модельному наборі не обов'язково відповідає робочому трафіку. Для оцінки невизначеності TPR визначимо частку успіхів  $\hat{p}$  та інтервал Вілсона:

$$\hat{p} = \frac{TP}{N}, \quad N = TP + FN \quad (8)$$

$$CI_{Wilson} = \frac{\hat{p} + \frac{z^2}{2N} \pm z \cdot \sqrt{\frac{\hat{p}(1-\hat{p})}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}} \quad (9)$$

Для повного протоколу порівняння конфігурацій задаються гіпотези про рівність часток і міра зв'язку для таблиці спряженості:

$$H_0: p_1 = p_2; \quad H_1: p_1 \neq p_2 \quad (10)$$



$$V = \sqrt{\frac{\chi^2}{N \cdot \min(r-1, c-1)}} \quad (11)$$

У (9)  $z = 1,96$  для рівня 95 %,  $N$  - кількість атакувальних випробувань. Біноміальна інтерпретація передбачає незалежність спостережень; для фіксованої суми класів цей інтервал є умовним. У (10)  $p_1$  і  $p_2$  - порівнювані частки; у (11)  $V$  - коефіцієнт Крамера,  $N$  - загальна кількість спостережень таблиці спряженості,  $r$  і  $c$  - кількість її рядків і стовпців. Позначення  $V$  тут відрізняється від вектора перевірок у розділі 4. Формули (10)–(11) задають інструменти для незалежних вибірок; для парних випробувань потрібен відповідний парний критерій. Через невідомий дизайн залежностей перевірку значущості за наявними агрегатами не виконано.

## РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

### Формалізація композиційного методу

Метод розділяє опис дозволу та рішення щодо його виконання. Підпис фіксує повідомлення  $M$  і домен  $D$ ; функція допуску  $A$  додатково враховує актуальний стан  $S$ . Модель містить підпис  $\sigma$ , вектор перевірок  $V$ , оператор переходу  $T$  та множину інваріантів  $I$ . Суб'єкта, чий підпис перевіряється, позначено  $u$ :

$$M_{SP} = \langle M, D, \sigma, S, V, A, T, I \rangle \quad (12)$$

$$M = (a, v, r, n, t, c) \quad (13)$$

Тут  $a$  - ідентифікатор дії;  $v$  - сума або структуровані предметні параметри;  $r$  - одержувач;  $n$  - попсе;  $t$  - deadline;  $c$  - додатковий контекст, зокрема ідентифікатор ресурсу. Усі параметри, що впливають на результат операції, мають входити до підписаного повідомлення. Інакше правильний підпис може супроводжувати непідписану зміну поведінки.

Контекст дозволу задає домен  $D$ . Разом із повідомленням він входить до хешу, який перевіряється під час валідації підпису:

$$D = (\text{name}, \text{version}, \text{chainId}, \text{verifyingContract}) \quad (14)$$

$$H(X) = \text{keccak256}(\text{typeHash}(X) \parallel \text{encodeData}(X)) \quad (15)$$

$$d = H(0x1901 \parallel H(D) \parallel H(M)) \quad (16)$$

Для структури  $X$  функція  $H(X)$  у (15) задає типізоване `hashStruct`; зовнішнє  $H$  у (16) означає `keccak256` байтової послідовності. `encodeData` враховує правила кодування динамічних полів і вкладених структур. Домен зв'язує дозвіл із мережею та контрактом: `chainId_D` і `this_D` - значення з домену, `address` - фактична адреса контракту. У проксі-схемі зміна семантики дозволу потребує визначеної політики `version`. На цій основі формується вектор умов допуску:

$$V(M, S) = [v_{\text{sig}}, v_{\text{ctx}}, v_{\text{nonce}}, v_{\text{time}}, v_{\text{role}}, v_{\text{lim}}, v_{\text{inv}}] \quad (17)$$

$$v_{\text{sig}} = I(\text{Verify}(u, d, \sigma) = \text{true}) \quad (18)$$

$$v_{\text{ctx}} = I(\text{chainId} = \text{chainId}_D) \cdot I(\text{address} = \text{this}_D) \quad (19)$$

$$v_{\text{nonce}} = I(n = N_u(S)) \quad (20)$$

$$v_{\text{time}} = I(\text{now} \leq t) \quad (21)$$

$$v_{\text{role}} = I(\text{Role}(u, a, S) = \text{true}) \quad (22)$$

$$v_{\text{lim}} = I(\text{Limit}(M, S) = \text{true}) \quad (23)$$

$$v_{\text{inv}} = I(I(S, M) = \text{true}) \quad (24)$$



Індикатор  $I(P)$  дорівнює одиниці, коли предикат  $P$  істинний;  $I(S, M)$  задає збереження інваріантів операції.  $N_u$  - лічильник суб'єкта,  $pow$  - час блока,  $Role$  - актуальні повноваження,  $Limit$  - предметні обмеження. Для контрактного гаманця  $Verify$  залежить від поточного стану механізму валідації. Перевірка  $v_{inv}$  є специфікацією вимог до переходу: реалізація має забезпечувати їх перевірку та порядком оновлення стану. Оскільки кожна умова необхідна, допуск задається добутком бінарних перевірок, еквівалентним їх кон'юнкції:

$$A(M, S) = \prod_{j=1}^7 v_j \quad (25)$$

$$A(M, S) = 1 \Leftrightarrow \bigwedge_{j=1}^7 (v_j = 1) \quad (26)$$

Умова допуску визначає можливість переходу. Відхилення залишає стан контракту незмінним, а успішна операція застосовує оператор  $T$ :

$$A(M, S) = 0 \Rightarrow S' = S \quad (27)$$

$$A(M, S) = 1 \Rightarrow S' = T(S, M) \quad (28)$$

Незмінність у формулі (27) стосується стану захищеного контракту. Вона не означає відсутності комісії: невдала транзакція витрачає  $gas$ . Формула (28) описує допустимий успішний перехід; якщо зовнішній виклик або завершальна перевірка дає помилку, транзакція атомарно відкочується. Отже, істинність  $A$  є умовою допуску, а фактичне завершення операції також залежить від успішності її виконання.

#### Одноразовість та інваріанти переходу

Одноразовість потребує не лише перевірки  $n$ , а й зміни лічильника після успішної операції. Для кожного суб'єкта  $u$  та значення  $n$  задаються такі умови:

$$N_u(S') = N_u(S) + 1 \quad (29)$$

$$\forall u, n: \sum_k \text{Successful}_k(u, n) \leq 1 \quad (30)$$

Розглянемо успішне виконання з  $n$  по  $n$ . За (20) до переходу  $N_u(S) = n$ , а за (29) після переходу лічильник дорівнює  $n + 1$ . Повторне подання того самого дозволу не задовольняє (20). Отже, (30) виконується за умов незменшення лічильника, відсутності його скидання чи переповнення та атомарного оновлення до зовнішньої взаємодії.  $\text{Successful}_k(u, n)$  - індикатор успішного виконання у спробі  $k$ . Відкат транзакції скасовує і зміну  $n$ , тому невдалий виклик не споживає дозвіл. Властивість стосується одного домену та узгодженої історії стану.

Для операції переказу внутрішнього балансу прикладні інваріанти включають невід'ємність залишків, обмеження суми та збереження сумарного обліку:

$$0 < v \leq L_a(S), \quad v \leq B_u(S), \quad r \neq 0 \quad (31)$$

$$B_u' = B_u - v, \quad B_r' = B_r + v \quad (32)$$

У (32) передбачається  $u \neq r$  і відсутність комісії або емісії; сума двох внутрішніх балансів зберігається. Для однакових адрес потрібно визначити окрему гілку. Для зовнішніх токенів із комісіями фактична зміна балансу може відрізнятись від номінальної, тому інваріант має враховувати спостережуваний результат.

Захист від повторного входу не впливає з  $n$  самостійно: атакувальник може мати інший чинний дозвіл або викликати суміжну незахищену функцію. Потрібні узгоджений порядок  $checks-effects-interactions$  і спільний захисний механізм для всіх функцій, які змінюють той самий критичний стан. Для складних взаємодій також перевіряється інваріант після зовнішнього виклику.

Алгоритм нижче задає необхідний порядок дій; він є псевдокодом, а не готовою реалізацією Solidity. Валідація контрактного підпису також є зовнішньою взаємодією, тому  $guard$  активується до неї.

```
execute(M, signature):
    enter shared reentrancy guard
    derive typed digest from trusted current domain
    identify signer and validate signature
    check nonce, deadline, current role and limits
    check preconditions and internal post-state
    increment signer nonce; update internal state
    perform external interaction; require valid result
    check final invariants; emit event
    leave guard; otherwise revert atomically
```

Предметні ліміти мають перевірятися з урахуванням стану, що змінюється в операції. Наприклад, денний ліміт вимагає не лише порівняння суми з константою, а й атомарного оновлення вже використаного обсягу. Роль перевіряється на момент виконання, тому підписання дозволу до відкликання не створює постійного права виконувати його після відкликання.

Послідовність обробки підписаної операції узагальнено на рис. 1. Схема відображає попередню обробку, композиційну валідацію, оновлення стану та виконання операції.

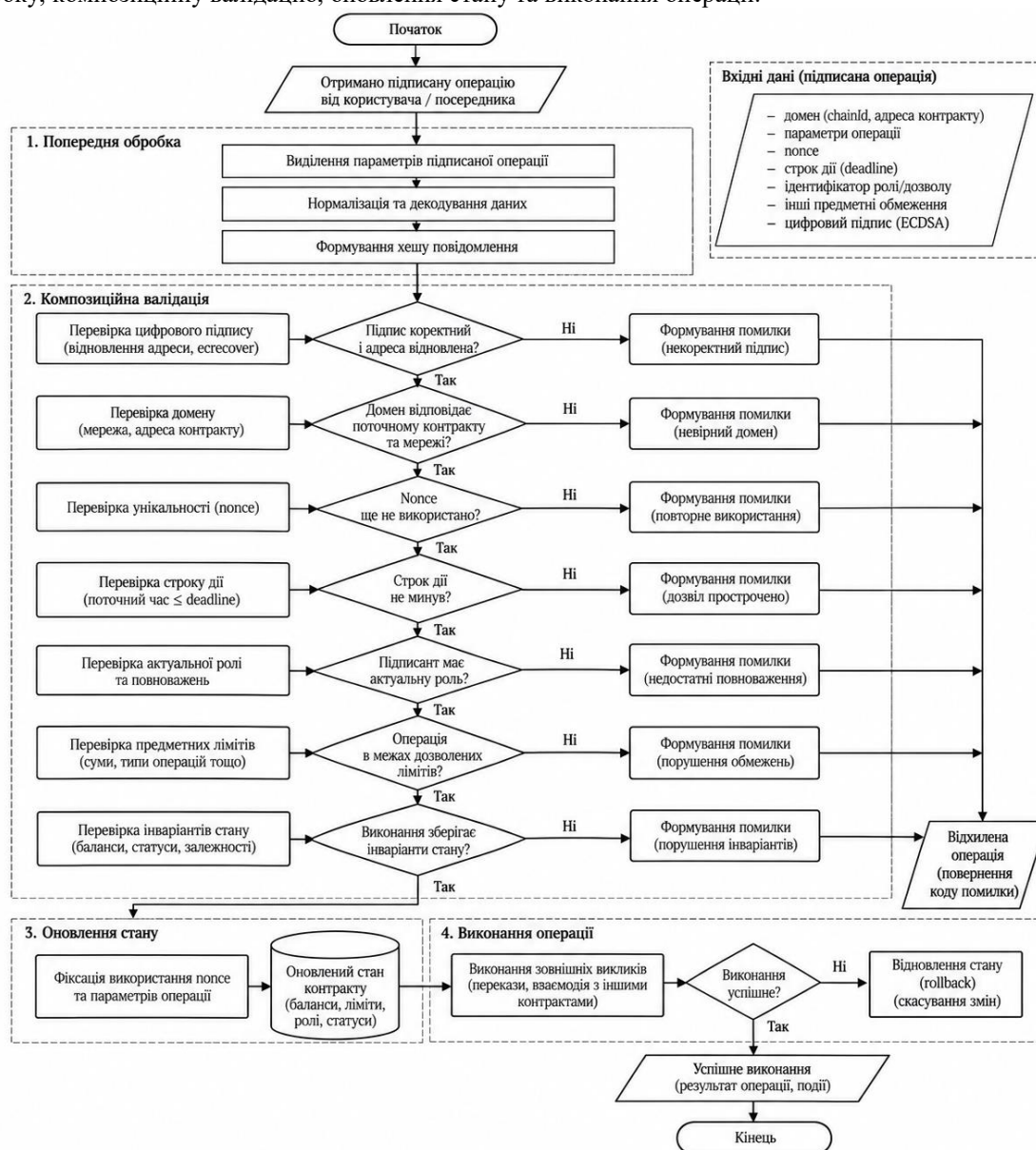


Рис. 1. Блок-схема композиційної валідації підписаних операцій



На рис. 1 представлено блок-схему запропонованого методу безпечного програмування децентралізованих додатків на основі композиційної валідації підписаних операцій. На відміну від підходу, за якого рішення про виконання операції приймається лише за результатом перевірки цифрового підпису, запропонований метод формує рішення про допуск як результат послідовної перевірки сукупності взаємопов'язаних умов, що характеризують автентичність запиту, контекст його виконання та актуальний стан смартконтракту.

Виконання алгоритму починається з отримання підписаної операції та виділення її параметрів. До складу повідомлення включаються параметри, здатні впливати на результат операції: ідентифікатор дії, предметні параметри, одержувач, попсе, строк дії та додатковий контекст. Принциповою умовою є включення таких параметрів саме до підписаного повідомлення, оскільки в іншому випадку коректний цифровий підпис не виключає можливості зміни непідписаних параметрів операції.

Після попередньої обробки виконується композиційна валідація. Спочатку перевіряється цифровий підпис та визначається суб'єкт, від імені якого сформовано дозвіл. Наступним рівнем є перевірка домену, що зв'язує підписаний дозвіл із конкретною мережею та конкретним смартконтрактом. Таким чином обмежується можливість перенесення коректно підписаної операції в інший контекст виконання. Домен безпосередньо пов'язується з мережею та контрактом через chainId і фактичну адресу контракту.

Наступні перевірки встановлюють актуальну допустимість операції. Перевірка попсе визначає, чи не був відповідний дозвіл використаний раніше; перевірка строку дії виключає виконання прострочених дозволів; перевірка ролі та повноважень встановлює, чи зберігає підписант необхідні права саме на момент виконання операції; предметні обмеження визначають відповідність суми, типу операції та інших параметрів установленим політиці. Завершальним елементом композиційної валідації є перевірка інваріантів стану, тобто умов, які повинні залишатися істинними після допустимого переходу системи.

Кожна перевірка є необхідною умовою допуску. Тому негативний результат на будь-якому етапі припиняє подальше виконання та переводить операцію на гілку відхилення із формуванням відповідної причини помилки. Така структура блок-схеми безпосередньо відображає математичну модель статті, у якій загальна умова допуску задається кон'юнкцією бінарних перевірок (25)–(26).

Якщо всі умови композиційної валідації виконані, алгоритм переходить до принципово важливого етапу - оновлення внутрішнього стану до здійснення зовнішньої взаємодії. Фіксується використання попсе та виконуються необхідні зміни балансів, лімітів, ролей, статусів або інших елементів критичного стану. Лише після цього дозволяється зовнішня взаємодія з іншими контрактами. Такий порядок відповідає закладеному в методі принципу checks–effects–interactions та зменшує ризик повторного входу. Водночас попсе не є самодостатнім захистом від reentrancy: необхідний узгоджений порядок оновлення стану та спільний захисний механізм для функцій, що працюють з одним критичним станом.

Завершальна частина алгоритму контролює результат виконання. У разі успішної зовнішньої взаємодії та виконання завершальних перевірок інваріантів формується результат операції та відповідна подія. Якщо виконання завершується помилкою, застосовується атомарне скасування внесених змін. Таким чином, блок-схема відображає не тільки процедуру прийняття рішення «дозволити/відхилити», а повний життєвий цикл підписаної операції - від її отримання до безпечної зміни стану або відкату.

#### **Переваги отриманого результату**

Основною перевагою запропонованого результату є перехід від ізольованої криптографічної перевірки до композиційної моделі авторизації. Коректний цифровий підпис підтверджує походження повідомлення, але сам по собі не встановлює, чи залишається операція допустимою після зміни стану, повноважень або контексту. Саме цю проблему вирішує запропонована схема.

Важливою перевагою є також невзаємозамінність захисних компонентів. Домен захищає від перенесення дозволу між контрактами та мережами, попсе - від повторного використання дозволу, перевірка актуальної ролі - від виконання операції після відкликання повноважень, а інваріанти та правильний порядок зміни стану забезпечують коректність переходу та посилюють захист критичних операцій. Модельний абляційний аналіз ілюструє цей зв'язок: після вилучення попсе блокування replay-атак зменшується до 2,5 %, після вилучення домену блокування міжконтрактних і міжмережових атак - до 3,4 % та 4,5 %, після вилучення перевірки актуальної ролі блокування відповідного класу атак - до 5,9 %.

Блок-схема показує основний шлях для підпису ECDSA. Спільний захист від повторного входу та завершальна перевірка інваріантів деталізовані в наведеному псевдокоді; їх слід враховувати під час реалізації, хоча на схемі вони не виділені окремими блоками. Для контрактного гаманця перевірка підпису потребує окремої гілки валідації. Повернення причини відхилення не означає збереження часткових змін: у разі помилки виконання вони атомарно скасовуються.

#### **Аналіз агрегованих модельних показників**

Таблиця 2 поєднує ефективність, помилкові відмови та модельні витрати. Значення TPR, FPR і gas взято з модельного набору; Precision та F<sub>1</sub> перераховано за цілими кількостями, сумісними з розмірами



вибірок та округленими частками. Для С зазначено 6948 заблокованих атак. Для решти конфігурацій кількості відновлено з агрегатів; вони не замінюють первинних спостережень.

Таблиця 2

Показники конфігурацій у модельному наборі

| Конфігурація | TPR, % | FPR, % | Precision, % | F <sub>1</sub> | Gas, mean $\pm$ SD |
|--------------|--------|--------|--------------|----------------|--------------------|
| A            | 22.74  | 0.45   | 99.44        | 0.3702         | 46211 $\pm$ 785    |
| B            | 25.57  | 0.70   | 99.22        | 0.4066         | 68155 $\pm$ 1219   |
| C            | 99.26  | 1.25   | 99.64        | 0.9945         | 89298 $\pm$ 1503   |
| C–N          | 85.41  | 1.05   | 99.65        | 0.9198         | 70819 $\pm$ 1331   |
| C–D          | 72.00  | 1.15   | 99.55        | 0.8356         | 83947 $\pm$ 1503   |
| C–R          | 85.94  | 0.85   | 99.72        | 0.9232         | 85581 $\pm$ 1510   |
| C–I          | 93.57  | 1.15   | 99.65        | 0.9652         | 83264 $\pm$ 1538   |

Повна конфігурація С має найбільший TPR, проте не усуває всі помилки: пропущено 52 атаки й відхилено 25 легітимних запитів. Матриця в табл. 3 розмежовує ці результати. Наявність пропусків не дозволяє ототожнювати високий середній TPR з виконанням усіх формальних властивостей; кожен пропуск потребує перевірки сценарію та передумов.

Таблиця 3

Матриця помилок повної конфігурації С

| Фактичний клас   | Відхилено | Прийнято | Усього |
|------------------|-----------|----------|--------|
| Атака            | 6948      | 52       | 7000   |
| Легітимний запит | 25        | 1975     | 2000   |

На рис. 2 зіставлено блокування окремих класів атак для В, С, С–N і С–D. Загальний TPR конфігурації С перевищує В на 73,69 відсоткового пункту. Розподіл за класами показує, чому одного загального показника недостатньо: висока ефективність для одних сценаріїв може приховувати систематичний пропуск інших.

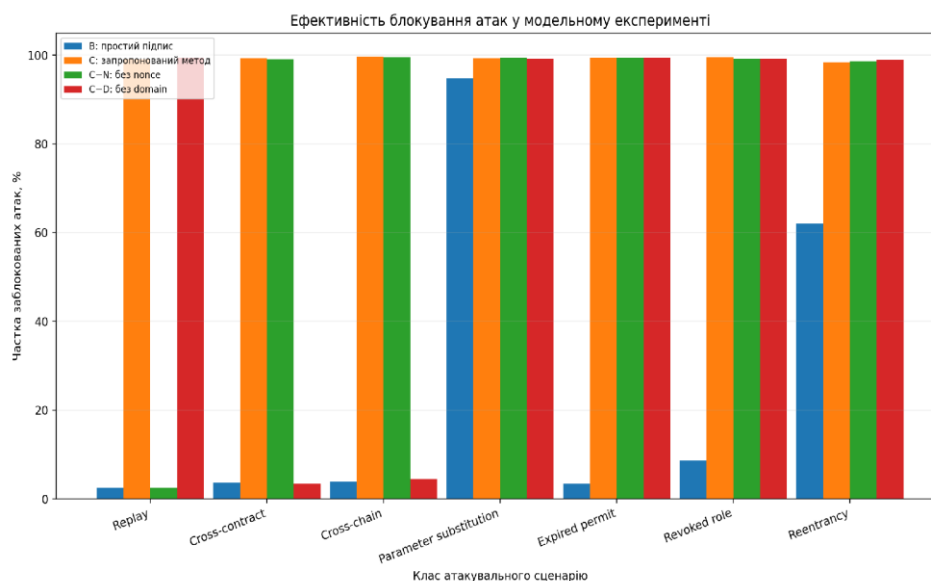


Рис. 2. Ефективність блокування окремих класів атак у модельному експерименті



Умовний 95 % інтервал Вілсона для 6948 успіхів із 7000 дорівнює 99,027–99,433 %. Він описує біноміальну невизначеність за передумови незалежності, але не враховує зміщення генератора, залежність сценаріїв і повноту моделі загроз. Твердження про статистичну значущість різниці конфігурацій не наведено, оскільки немає парних результатів або підтвердження незалежного дизайну.

## Абляційний аналіз

Абляційний аналіз пов'язує окрему перевірку зі змінами ефективності. У (33)  $C_0$  позначає повну конфігурацію  $C$ , а  $\setminus$  - вимкнення компонента за незмінності інших умов. Для вилученого компонента  $j$  величина  $\Delta_j$  у (34) є різницею загального TPR. Матриця на рис. 3 деталізує результат за класами атак:

$$C_{-N} = C_0 \setminus v_{\text{nonce}}; C_{-D} = C_0 \setminus v_{\text{ctx}}; C_{-R} = C_0 \setminus v_{\text{role}}; C_{-I} = C_0 \setminus v_{\text{inv}} \quad (33)$$

$$\Delta_j = \text{TPR}_C - \text{TPR}_{C-j} \quad (34)$$

Найпомітніші локальні зміни відповідають призначенню вилучених перевірок: без nonce блокування replay становить 2,5 %, без домену міжконтрактних і міжмережових атак - 3,4 % і 4,5 %, без актуальної ролі атак після її відкликання - 5,9 %. Без інваріантів частка блокування повторного входу дорівнює 58,7 %. Матриця вказує на сценарії, для яких перевірки не є взаємозамінними; причинний внесок кожного компонента потребує контрольованого відтворення.

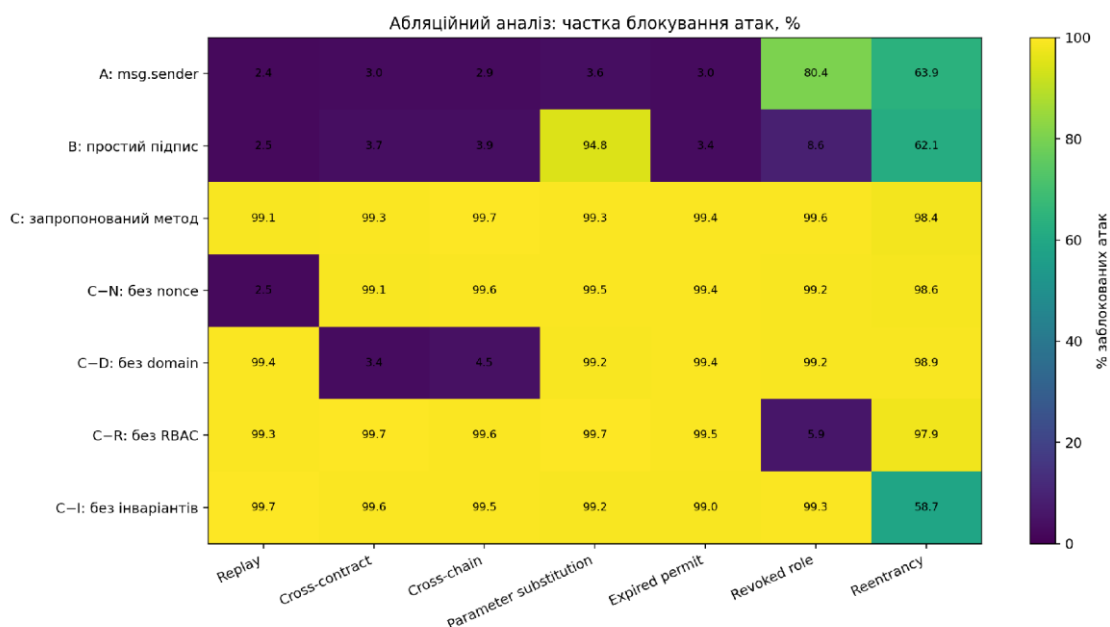


Рис. 3. Абляційний аналіз внеску окремих компонентів методу

Результат для домену слід тлумачити з урахуванням складу набору: його вилучення безпосередньо зачіпає два класи перенесення дозволу. Тому найбільше загальне падіння TPR не означає універсально найбільшої важливості цього компонента. Водночас показник повторного входу не можна приписувати лише nonce або одному предикату інваріантів: він залежить від порядку оновлення стану та захисту всіх пов'язаних функцій.

## Обчислювальні витрати

Нехай  $G_{ki}$  - витрати у випробуванні і конфігурації  $k$ , а  $N$  - кількість спостережень витрат. Середнє, вибіркове стандартне відхилення та приріст витрат  $C$  відносно  $B$  визначаються так:

$$\bar{G}_k = \frac{1}{N} \sum_{i=1}^N G_{ki} \quad (35)$$

$$s_{G,k} = \sqrt{\frac{\sum_{i=1}^N (G_{ki} - \bar{G}_k)^2}{N-1}} \quad (36)$$

$$\Delta G_C = \frac{\bar{G}_C - \bar{G}_B}{\bar{G}_B} \cdot 100\% \quad (37)$$

На рис. 4 показано розподіли модельних витрат для A, B, C, C-N і C-D. Медіани та міжквартильні діапазони описують центр і розкид, тоді як табл. 2 містить середні та стандартні відхилення за (35)–(36). Правило побудови вусів і первинні вимірювання недоступні, тому діаграма використовується описово. Відносно збільшення середніх витрат C проти B визначено за (37).

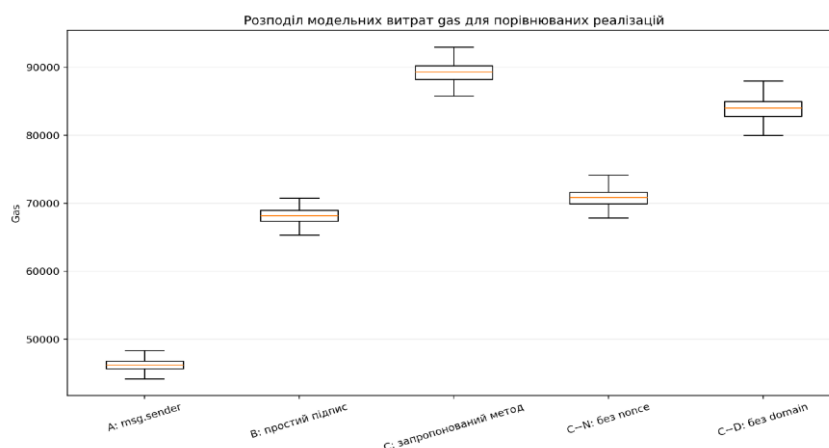


Рис. 4. Розподіл модельних витрат gas для порівнюваних реалізацій

Перехід від B до C збільшує середню модельну вартість на 21143 gas, або 31,02 %. Зростання супроводжується підвищенням TPR, але агрегати не дають змоги розкласти витрати за перевірками чи відокремити успішні операції від відхилених. Для практичного оцінювання потрібні вимірювання за фіксованих версій компілятора та EVM, налаштувань оптимізації й умов доступу до сховища.

На рис. 5 конфігурації зіставлено за двома вимірами - TPR і середніми витратами. Дешевший варіант не є прийнятною заміною, якщо він втрачає обов'язкову властивість, наприклад одноразовість дозволу. Крім того, такий графік не відображає FPR, тому підсумкове оцінювання має бути багатокритеріальним:

$$E = \langle E_{\text{sec}}, \text{FPR}, G \rangle \quad (38)$$

У (38)  $E_{\text{sec}}$  характеризує блокування атак і оцінюється через TPR; FPR характеризує помилкові відмови,  $G$  - модельні обчислювальні витрати. Спільний розгляд компонентів вектора дає змогу оцінювати захист разом із доступністю та вартістю виконання. Допустимі межі кожного показника визначаються вимогами прикладної системи.

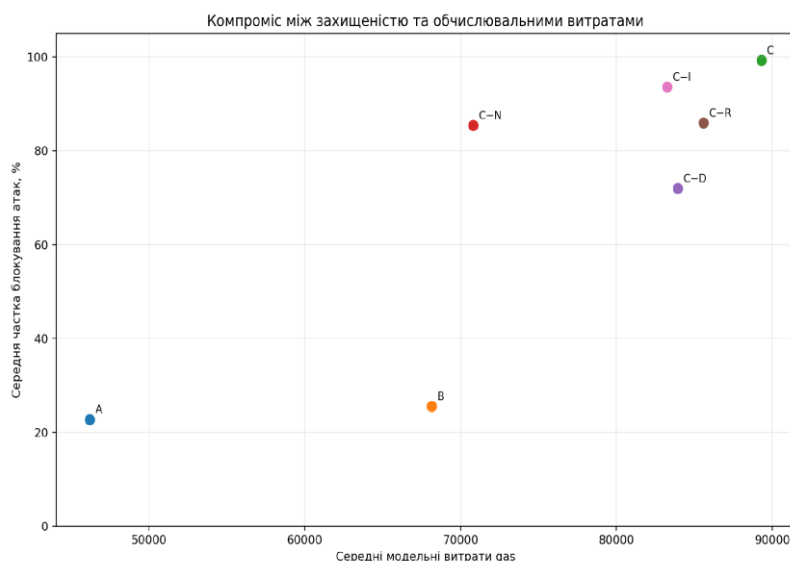


Рис. 5. Співвідношення модельної ефективності та середніх витрат

**Помилкові рішення та чутливість до складу захисту**

Для інтерпретації загального TPR необхідно окремо оцінити залишкові атаки й відмови легітимним користувачам. Рисунки 6–8 деталізують агрегати табл. 2 в цих аспектах та узагальнюють результати абляції. Усі три графіки є похідними від того самого набору, тому не становлять незалежного експериментального підтвердження.

Частка пропущених атак FNR доповнює TPR, а специфічність показує частку правильно прийнятих легітимних запитів. Збалансована точність BA надає однакову вагу атакувальному й легітимному класам незалежно від їхньої кількості в наборі:

$$FNR = \frac{FN}{TP + FN} = 1 - TPR \quad (39)$$

$$Specificity = \frac{TN}{TN + FP} = 1 - FPR \quad (40)$$

$$BA = \frac{TPR + Specificity}{2} \quad (41)$$

$$TPR_{macro} = \frac{1}{7} \sum_{j=1}^7 TPR_j \quad (42)$$

Макроусереднення TPR за сімома класами збігається із загальним TPR лише за однакової кількості випробувань у кожному класі, як у наданому наборі. Формули FNR, Specificity та BA обчислюються за частками від 0 до 1; відображення у відсотках потребує множення на 100.

Рис. 6 показує ціну посилення перевірок для легітимних запитів. У С частка помилкових відмов дорівнює 1,25 % проти 0,70 % у В: це 25 і 14 відмов із 2000 відповідно. Додаткові 11 відмов потребують аналізу причин - прострочення, порядку попсо або інших умов допуску; самі агрегати не дозволяють визначити, яка перевірка їх спричинила.

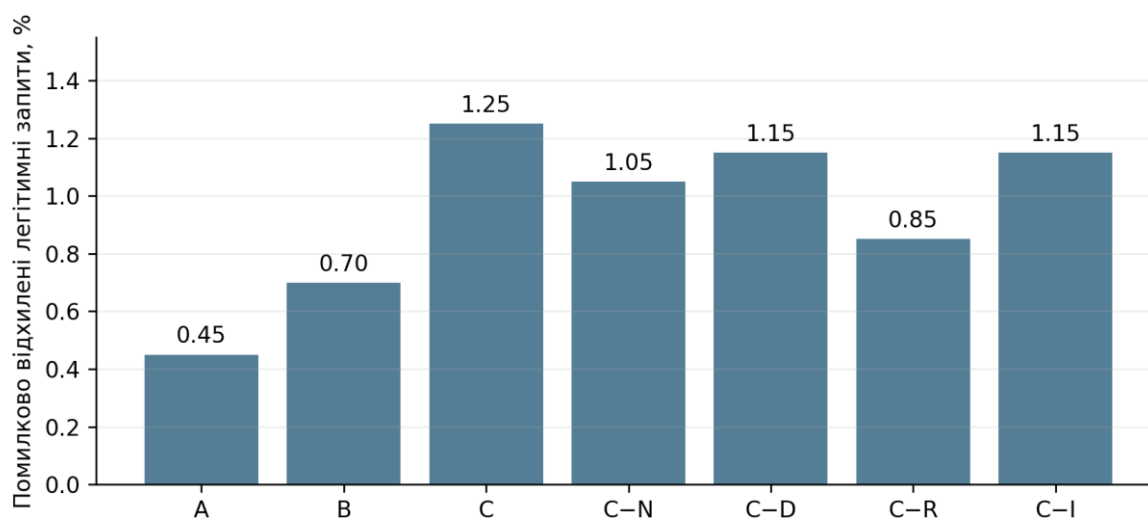


Рис. 6. Помилкові відмови легітимним запитам за модельними даними

На рис. 7 наведено кількість пропущених атак  $FN = 7000 - TP$ . Для С вона дорівнює 52, для В - 5210. Кількості для конфігурацій, крім С, відновлено з розміру набору й округлених TPR у табл. 2; це похідні агрегати, а не нові спостереження. Графік показує абсолютний масштаб залишкових помилок, який менш помітний при порівнянні високих відсотків блокування.

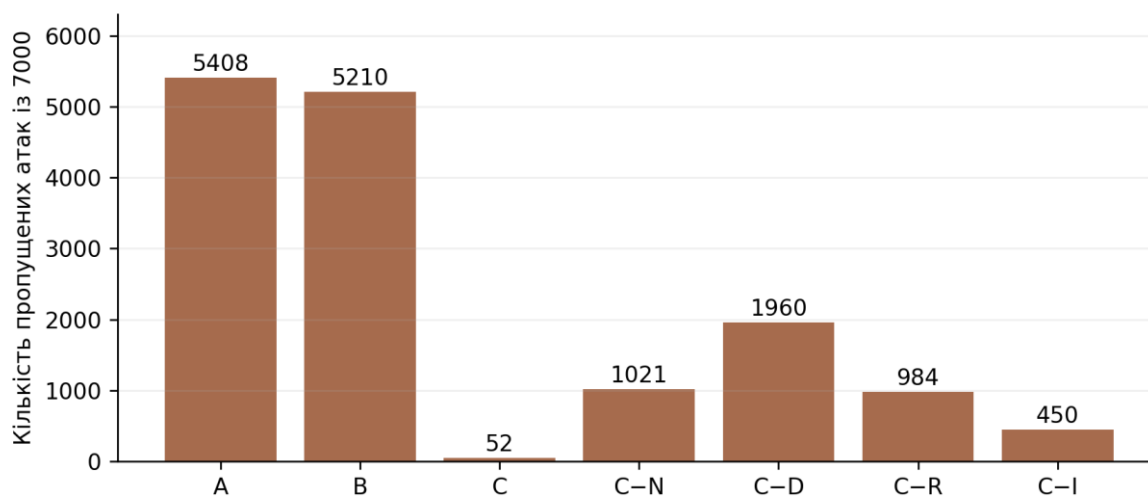


Рис. 7. Кількість пропущених атак у модельному наборі

Рис. 8 узагальнює втрати TPR після вилучення компонентів: домену - 27,26, попсе - 13,84, актуальної ролі - 13,31, інваріантів - 5,69 відсоткового пункту. Різниці обчислено за відновленими цілими кількостями ТР. Ці значення залежать від ваг класів атак; вони характеризують чутливість цього набору, а не універсальний порядок важливості перевірок.

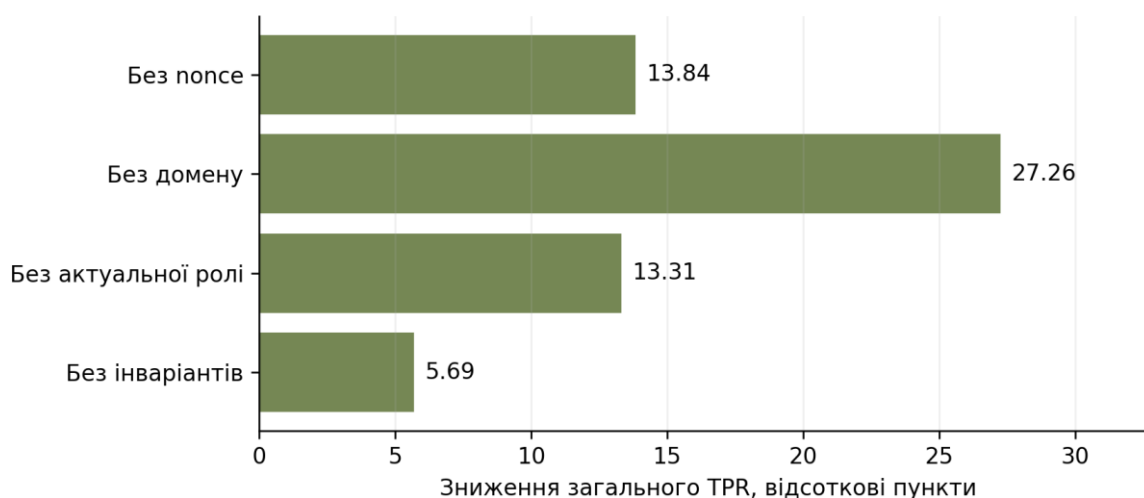


Рис. 8. Зниження загальної ефективності після вилучення компонентів

Для С специфічність становить 98,75 %, FNR - 0,743 %, BA - 99,004 %; для В - 99,30 %, 74,429 % і 62,436 % відповідно. Збалансована точність враховує обидва класи, але надає їм однакову вагу. Вона не замінює оцінювання збитків: пропущена атака й помилкова відмова можуть мати суттєво різні наслідки. Тому вибір конфігурації має спиратися також на політику ризику системи.

#### Обговорення результатів та меж застосування

Формальний результат роботи - специфікація, яка пов'язує підписаний дозвіл з актуальним станом і порядком його зміни. Її практичний зміст полягає в перевірюваних вимогах: кожне поле операції зв'язане з підписом; домен відповідає контексту; попсе споживається один раз; права перевіряються на момент виконання; перехід зберігає предметні інваріанти. Модельний аналіз ілюструє наслідки вилучення цих умов, але не є доведенням коректності конкретного смартконтракту.

Перевірка підпису не захищає користувача, який свідомо або через фішинг підписав небезпечну операцію. Домен не розрізняє форки з однаковим chainId, адресою й станом. Відкликання ролі не скасовує підпис автоматично, якщо реалізація не перевіряє її актуальний стан. Послідовний попсе може спричинити відмови під час надсилання дозволів не за порядком. Ці обмеження потрібно враховувати під час



проектування інтерфейсу та політики повторного подання.

Достовірність кількісних висновків обмежують три чинники: невідомий механізм генерації сценаріїв, відсутність журналів для перевірки залежностей між випробуваннями та модельний характер gas. Фіксовані ваги семи класів обмежують перенесення TPR на інший профіль загроз. Для відтворюваної перевірки необхідні контракти й тести, параметри середовища, генератор із фіксованим початковим значенням та результати кожного запиту. Окремо слід аналізувати 52 пропущені атаки конфігурації C.

## ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

Запропоновано композиційну специфікацію допуску підписаних операцій, яка поєднує перевірку автентичності з доменом, попсе, строком дії, поточними повноваженнями, лімітами та інваріантами. Встановлено умови одноразового успішного виконання в межах одного домену й узгодженої історії стану. Визначено порядок перевірок, споживання попсе та оновлення стану перед зовнішньою взаємодією.

У модельному наборі повна композиція має TPR 99,26 % проти 25,57 % для простого підпису, зі зростанням FPR з 0,70 % до 1,25 % та середніх витрат на 31,02 %. Абляційний аналіз виявляє локальні втрати захисту в класах атак, пов'язаних із вилученою перевіркою. Водночас 52 пропущені атаки показують необхідність аналізу залишкових помилок; наведені частки не підтверджують повноти захисту або статистичної значущості відмінностей.

Практичним результатом є набір умов допуску, властивостей переходу та сценаріїв тестування для реалізації авторизації. Наступний етап - реалізація специфікації в EVM, публікація відтворюваних тестів і перевірка причин помилок. Оцінювання має охоплювати міжфункціональний повторний вхід, контрактні гаманці, відкликання прав та оновлення проксі, а вимірювання gas - розділяти успішні й відхилені операції.

Використання інструментів штучного інтелекту. Під час підготовки рукопису використано ChatGPT (OpenAI) для редагування тексту, структурування формальної моделі та пошуку бібліографічних відомостей. Чотири оригінальні графіки збережено без зміни вмісту; код трьох додаткових графіків підготовлено за допомогою ChatGPT, значення обчислено з агрегованої таблиці. Автори мають перевірити твердження, джерела й результати перед поданням; остаточною відповідальністю за рукопис належить авторам.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Atzei, N., Bartoletti, M., & Cimoli, T. (2017). A survey of attacks on Ethereum smart contracts (SoK). In *Principles of Security and Trust* (pp. 164–186). Springer. [https://doi.org/10.1007/978-3-662-54455-6\\_8](https://doi.org/10.1007/978-3-662-54455-6_8)
2. Luu, L., Chu, D.-H., Olickel, H., Saxena, P., & Hobor, A. (2016). Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 254–269). ACM. <https://doi.org/10.1145/2976749.2978309>
3. Tsankov, P., Dan, A., Drachsler-Cohen, D., Gervais, A., Bünzli, F., & Vechev, M. (2018). Securify: Practical security analysis of smart contracts. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (pp. 67–82). ACM. <https://doi.org/10.1145/3243734.3243780>
4. Feist, J., Grieco, G., & Groce, A. (2019). Slither: A static analysis framework for smart contracts. In *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)* (pp. 8–15). IEEE. <https://doi.org/10.1109/WETSEB.2019.00008>
5. Jiang, B., Liu, Y., & Chan, W. K. (2018). ContractFuzzer: Fuzzing smart contracts for vulnerability detection. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering* (pp. 259–269). ACM. <https://doi.org/10.1145/3238147.3238177>
6. He, J., Balunović, M., Ambroladze, N., Tsankov, P., & Vechev, M. (2019). Learning to fuzz from symbolic execution with application to smart contracts. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (pp. 531–548). ACM. <https://doi.org/10.1145/3319535.3363230>
7. Durieux, T., Ferreira, J. F., Abreu, R., & Cruz, P. (2020). Empirical review of automated analysis tools on 47,587 Ethereum smart contracts. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (pp. 530–541). ACM. <https://doi.org/10.1145/3377811.3380364>
8. Zheng, Z., Xie, S., Dai, H.-N., Chen, W., Chen, X., Weng, J., & Imran, M. (2020). An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, 105, 475–491. <https://doi.org/10.1016/j.future.2019.12.019>
9. Sayeed, S., Marco-Gisbert, H., & Caira, T. (2020). Smart contract: Attacks and protections. *IEEE Access*, 8, 24416–24427. <https://doi.org/10.1109/ACCESS.2020.2970495>
10. Mavridou, A., & Laszka, A. (2018). Designing secure Ethereum smart contracts: A finite state machine based approach. In *Financial Cryptography and Data Security* (pp. 523–540). Springer.



- [https://doi.org/10.1007/978-3-662-58387-6\\_28](https://doi.org/10.1007/978-3-662-58387-6_28)
11. Жебка, В. В. (2026). Блокчейн-орієнтована архітектура моніторингу та адаптивного управління функціонально стійкою гетерогенною телекомунікаційною мережею. Кібербезпека: освіта, наука, техніка, 1(33). <https://doi.org/10.28925/2663-4023.2026.33.1287>
  12. Аронов, А. О. (2026). Метод підвищення безпеки децентралізованих систем шляхом адаптивного виявлення аномалій у блокчейн-мережах із використанням машинного навчання. Кібербезпека: освіта, наука, техніка, 1(33), 657-666. <https://doi.org/10.28925/2663-4023.2026.33.1141>

**Zhebka Viktoriia**

Doctor of Technical Sciences, Professor, Head of the Department of Digital Development Technologies  
State University of Information and Communication Technologies, Kyiv, Ukraine  
ORCID: 0000-0003-4051-1190  
[v.zhebka@duikt.edu.ua](mailto:v.zhebka@duikt.edu.ua)

**Aronov Andrii**

Candidate of Technical Sciences, Associate Professor, Department of Digital Development Technologies  
State University of Information and Communication Technologies, Kyiv, Ukraine  
ORCID: 0009-0000-7868-8341  
[a.aronov@duikt.edu.ua](mailto:a.aronov@duikt.edu.ua)

**Gertsyuk Mykola**

PhD, Associate Professor, Department of Digital Development Technologies  
State University of Information and Communication Technologies, Kyiv, Ukraine  
ORCID: 0000-0003-2946-9673  
[m.gertsyuk@duikt.edu.ua](mailto:m.gertsyuk@duikt.edu.ua)

**Hanenko Liudmyla**

PhD, Senior Lecturer in the Department of Digital Development Technologies State University of Information and Communication Technologies, Kyiv, Ukraine  
ORCID: 0000-0003-2219-8196  
[hanenkoliudmyla@gmail.com](mailto:hanenkoliudmyla@gmail.com)

**A METHOD FOR SECURE PROGRAMMING OF DECENTRALIZED APPLICATIONS THROUGH COMPOSITIONAL VALIDATION OF SIGNED OPERATIONS**

**Abstract.** This article studies authorization of signed operations in decentralized applications based on smart contracts. A valid signature authenticates a message but does not establish whether it remains executable after changes in state, privileges or context. The objective is to formalize a compositional admission method and assess attack blocking, false rejections and computational costs. The study combines a literature review, state transition modeling and descriptive analysis of aggregate simulation data. The specification integrates signature, domain, nonce, expiry, current role, application limits and invariant checks. Conditions for at most one successful execution are established within one domain and a consistent state history. Validation and state updates are ordered before external interactions. The model dataset includes seven attack classes and legitimate requests; component contributions are examined through ablation. The full configuration blocks 99.26% of attacks, compared with 25.57% for basic signature verification. False rejections increase from 0.70% to 1.25%, while mean modeled costs rise by 31.02%. Removing the domain, nonce or current role check is associated with pronounced losses for the corresponding attack classes. Aggregate effectiveness depends on the class distribution, and a high average rate does not eliminate the need to investigate missed attacks. These results are descriptive: missing generator code and trial records prevent independent reproduction and limit causal interpretation and significance testing. Modeled costs are not measurements of a deployed EVM contract. Practical value lies in a coherent authorization specification and test scenarios linking each check to a transition property. Further work should implement the method in EVM, investigate residual errors and publish reproducible trials, including contract wallets and cross-function reentrancy.

**Keywords:** decentralized application; smart contract; digital signature; authorization; nonce; reentrancy; state invariant; cryptography; information technologies.

**REFERENCES**

1. Atzei, N., Bartoletti, M., & Cimoli, T. (2017). A survey of attacks on Ethereum smart contracts (SoK). In *Principles of Security and Trust* (pp. 164–186). Springer. [https://doi.org/10.1007/978-3-662-54455-6\\_8](https://doi.org/10.1007/978-3-662-54455-6_8)
2. Luu, L., Chu, D.-H., Olickel, H., Saxena, P., & Hobor, A. (2016). Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 254–269). ACM. <https://doi.org/10.1145/2976749.2978309>



3. Tsankov, P., Dan, A., Drachsler-Cohen, D., Gervais, A., Bünzli, F., & Vechev, M. (2018). Securify: Practical security analysis of smart contracts. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (pp. 67–82). ACM. <https://doi.org/10.1145/3243734.3243780>
4. Feist, J., Grieco, G., & Groce, A. (2019). Slither: A static analysis framework for smart contracts. In *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)* (pp. 8–15). IEEE. <https://doi.org/10.1109/WETSEB.2019.00008>
5. Jiang, B., Liu, Y., & Chan, W. K. (2018). ContractFuzzer: Fuzzing smart contracts for vulnerability detection. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering* (pp. 259–269). ACM. <https://doi.org/10.1145/3238147.3238177>
6. He, J., Balunović, M., Ambroladze, N., Tsankov, P., & Vechev, M. (2019). Learning to fuzz from symbolic execution with application to smart contracts. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (pp. 531–548). ACM. <https://doi.org/10.1145/3319535.3363230>
7. Durieux, T., Ferreira, J. F., Abreu, R., & Cruz, P. (2020). Empirical review of automated analysis tools on 47,587 Ethereum smart contracts. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (pp. 530–541). ACM. <https://doi.org/10.1145/3377811.3380364>
8. Zheng, Z., Xie, S., Dai, H.-N., Chen, W., Chen, X., Weng, J., & Imran, M. (2020). An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, 105, 475–491. <https://doi.org/10.1016/j.future.2019.12.019>
9. Sayeed, S., Marco-Gisbert, H., & Caira, T. (2020). Smart contract: Attacks and protections. *IEEE Access*, 8, 24416–24427. <https://doi.org/10.1109/ACCESS.2020.2970495>
10. Mavridou, A., & Laszka, A. (2018). Designing secure Ethereum smart contracts: A finite state machine based approach. In *Financial Cryptography and Data Security* (pp. 523–540). Springer. [https://doi.org/10.1007/978-3-662-58387-6\\_28](https://doi.org/10.1007/978-3-662-58387-6_28)
11. Zhebka, V. V. (2026). Blockchain-oriented architecture for monitoring and adaptive management of a functionally resilient heterogeneous telecommunication network. *Cybersecurity: Education, Science, Technique*, 1(33). <https://doi.org/10.28925/2663-4023.2026.33.1287> [in Ukrainian].
12. Aronov, A. O. (2026). Method for enhancing the security of decentralized systems through adaptive anomaly detection in blockchain networks using machine learning. *Cybersecurity: Education, Science, Technique*, 1(33), 657–666. <https://doi.org/10.28925/2663-4023.2026.33.1141> [in Ukrainian].

Отримано редакцією журналу / Received: 04.09.26

Прорецензовано / Revised: 18.09.26

Схвалено до друку / Accepted: 26.10.26

