ISSN 0485-8972

УДК 004.05 **DOI:10.30837/rt.2026.1.224.02**

В.І. ЄСІН, д-р техн. наук, В.В. ВИЛИГУРА, PhD

ОСНОВНІ АСПЕКТИ КОНЦЕПЦІЇ НУЛЬОВОГО РОЗГОЛОШЕННЯ: ТЕОРЕТИЧНІ ОСНОВИ, СУЧАСНІ ZKP-СИСТЕМИ ТА КОНЦЕПТУАЛЬНЕ ПРЕДСТАВЛЕННЯ ТЕХНОЛОГІЇ ZK-SNARK

Вступ

В умовах стрімкої повсюдної диджиталізації суспільства, активного впровадження нових інноваційних технологій та зростання кіберзагроз, питання організації ефективної кібербезпеки підприємств, організацій та установ набуває особливої значущості. Традиційне поняття периметра кібербезпеки, яке колись (коли офіси розміщували комп'ютери та сервери за мережевими екранами та VPN (*Virtual Private Network*)) було адекватним, більше не відповідає реальній дійсності [1]. Багато підприємств, організацій, установ більше не мають чітко визначеного периметра. Периметр втрачає свою актуальність через кілька факторів, включаючи зростання хмарних обчислень, мобільність та зміни у сучасному штаті співробітників (використання віддалених працівників) [2]. Крім того, слід враховувати, що загрозу становлять і внутрішні зловмисники – інсайдери. Таким чином, ідея про те, що жодна мережа (ні внутрішня, ні зовнішня) не заслуговує на довіру, просувається як у наукових колах, так і на практиці [3]. Це змушує переглянути підходи до побудови систем безпеки та визнати недосконаліть класичних стратегій захисту на основі периметру. Щоб сьогодні захистити сучасне цифрове підприємство, необхідна комплексна стратегія для безпечного доступу у будь-який час і в будь-якому місці до власних корпоративних ресурсів (застосунків, успадкованих систем, даних, пристроїв тощо) незалежно від того, де вони розташовані [4]. Тому підприємства стали переосмислювати традиційний периметр безпеки мережі, схиляючись до нової концепції та архітектури захисту [5, 6]. У цьому контексті особливого значення набуває концепція нульової довіри (*Zero Trust* – ZT), що формує нову парадигму захисту інформації, орієнтовану на користувача, ресурси та дані. Дотримуючись принципів архітектури нульової довіри (*Zero Trust Architecture* – ZTA), що передбачають доступ з найменшими привілеями та безперервну перевірку, підприємства можуть ефективно мінімізувати площу атаки та обмежити потенційні збитки від скомпрометованих облікових записів. Однак одних існуючих механізмів контролю доступу та автентифікації самих по собі не завжди достатньо для забезпечення повного захисту критично важливих даних, особливо у сценаріях, де потрібний доказ прав доступу або дій без розкриття вмісту. У таких випадках ефективним доповненням до ZTA може стати використання концепції «нульового знання/розголошення» (*Zero Knowledge* – ZK), що дозволяє підтверджувати права доступу або факт володіння (знання) певною інформацією без необхідності її розкриття, що суттєво знижує ризики витоків та несанкціонованого доступу. Стрімкий розвиток розподілених обчислювальних середовищ, блокчейн-технологій та децентралізованих систем обробки даних призвів до підвищеного інтересу до криптографічних механізмів, що забезпечують одночасно конфіденційність та верифікованість інформації. Одним із найбільш значущих напрямків сучасної криптографії в цьому сенсі є докази з нульовим розголошенням (*Zero-Knowledge Proofs* – ZKP), що дозволяють переконатися у правильності обчислень або тверджень, не розкриваючи при цьому жодних додаткових даних, крім факту їхньої істинності.

ZKP, спочатку розроблені академічною спільнотою у 1980-х роках [7, 8], сьогодні набули значного розвитку. Вони стали придатними для практичного застосування в різних галузях, що становлять інтерес для промисловості, а також для великої спільноти розробників та дослідників [9]. ZKP успішно використовуються у складних реальних сценаріях, де завдання полягає у доведенні знань без їх розкриття, зокрема:

– *криптовалюти* (*cryptocurrencies*) *та блокчейн* (*blockchain*) – для забезпечення конфіденційності транзакцій. Користувачі можуть здійснювати перекази, приховуючи інформацію про відправника, одержувача та суму транзакції, при цьому зберігаючи можливість перевірити легітимність транзакції;

– *системи автентифікації* (*authentication systems*). Користувач може довести, що він володіє певним секретом (паролем, біометричними даними тощо), не розкриваючи сам секрет стороні, що перевіряє;

– *цифрові підписи* (*digital signatures*). Докази з нульовим розголошенням можуть використовуватися в схемах цифрового підпису для автентифікації особистості сторони, що підписує, без розкриття її закритого ключа (наприклад, протокол Шнорра [10, 11] є основним методом доказу з нульовим розголошенням, що лежить в основі багатьох сучасних схем цифрового підпису);

– *безпечні багатосторонні обчислення* (*Secure Multiparty Computation* – MPC). ZKP активно та ефективно використовуються у безпечних багатосторонніх обчисленнях, допомагаючи учасникам обчислювального протоколу переконатися у коректності дій інших сторін без розкриття їх секретних даних. Більше того, гібридні конструкції ZKP і MPC вважаються основним напрямом для промислових систем, що вимагають одночасно високої приватності та коректності [12–14]. Це підвищує довіру до процесу, знижує ризики шахрайства та підвищує безпеку спільних обчислень;

– *безпечна ідентифікація* (*secure identification*). У системах цифрової ідентифікації ZKP можуть бути використані для захисту конфіденційності користувачів. Традиційні методи цифрової ідентифікації часто вимагають від учасників інформаційного обміну надання більшої кількості інформації ніж необхідно, розкриваючи небажані ідентифікаційні та криптографічні дані. За допомогою ZKP учасники інформаційного обміну можуть підтверджувати тільки ту інформацію, яка необхідна верифікатору, і мінімізувати розкриття даних. Цифрові підписи BBS, BBS+ [15–17] є прикладами рішень на основі ZKP, які дозволяють здійснювати ідентифікацію із збереженням конфіденційності,

тощо.

Різні системи ZK-доказів ґрунтуються на різних криптографічних припущеннях та розроблені для вирішення різних обчислювальних завдань. Їх реалізації пишуться різними мовами програмування та тестуються на різному обладнанні [18]. Але незважаючи на різні напрямки застосування ZKP, виявлені в ході численних досліджень та реалізацій, сьогодні вважається, що процес переходу від теорії до широко застосовуваних практичних рішень для ZKP все ще не завершено. У цьому аспекті фахівці наголошують на необхідності подальших досліджень, особливо в галузі оптимізації їх впровадження та розширення сфери застосування [19]. Крім того, практичне застосування вже відомих можливостей ZKP у різних актуальних напрямках, що забезпечують безпеку, гальмується, в тому числі, через непоінформованість та недостатню теоретичну підготовку в даному аспекті фахівців, які відповідають за безпеку та доводять ці можливості (їх потенціал) до керівників відповідних IT-підприємств, організацій та установ.

Мета статті полягає в систематизації теоретичних основ та практичного застосування на простих і очевидних прикладах концепції нульового розголошення для розуміння потенціалу ZKP у вирішенні завдань конфіденційності/приватності та верифікованості даних. Це, на нашу думку, дозволить краще розібратися і використовувати надалі непросту і динамічно еволюціонуючу концепцію як одного з ключових механізмів сучасної криптографії, що забезпечують можливість доказу правильності обчислень без розкриття самого обчислювального процесу або вихідних даних.

1. Концепція нульового розголошення: важливі аспекти та визначення

У широкому розумінні *Zero Knowledge* (нульове знання/розголошення) можна визначити як криптографічну концепцію, яка використовується в багатьох криптографічних застосун-

ках (ідентифікації, шифрування, доказу), що забезпечують конфіденційність [7, 8, 20–26]. Ця концепція має вирішальне значення у технологіях збереження конфіденційності, оскільки вона дозволяє перевіряти транзакції або дії, не розкриваючи жодних фактичних /конфіденційних даних.

З іншого боку, поняття Zero Knowledge розглядається як загальна властивість, що описує мінімальне розкриття інформації, що застосовується до широкого класу криптографічних застосунків, у тому числі в контексті доказів з нульовим розголошенням (ZKP), при якому одна сторона (що доказує – *prover*) може довести іншій стороні (що перевіряє – верифікатору, перевіряльнику – *verifier*), що певне твердження є істинним, не передаючи жодної інформації, крім того факту, що твердження дійсно є істинним.

Далі нас більшою мірою цікавитиме аспекти ZK саме у контексті доказів із нульовим розголошенням. Докази з нульовим розголошенням/знанням вперше зустрічаються у статті [7], згодом доопрацьованій та розширеній [8]. У них автори визначають докази з нульовим розголошенням як докази, які не передають жодних додаткових знань, крім самого факту істинності/правильності розглянутого (висунутого) твердження (пропозиції). Докази з нульовим розголошенням [7]) – це елегантний метод, що дозволяє обмежити обсяг інформації, що передається у криптографічному протоколі від сторони *P* (*prover*), що доводить, стороні *V* (*verifier*), що перевіряє, відзначають автори роботи [22]. Автор роботи [23] також говорить про ZKP як про методи перевірки тверджень без розкриття самої інформації, в процесі якого той, хто доводить (*prover*), ділиться доказом свого твердження з «верифікатором» (*verifier*), який потім перевіряє точність доказу, не дізнаючись ніякої додаткової інформації. Тобто одна сторона (*prover*) переконує іншу сторону (верифікатора) у істинності деякого твердження, не передаючи верифікатору жодної інформації, крім самого факту істинності цього твердження. Крім того, в різних релевантних джерелах [7, 8, 21, 22, 27] про ZKP йдеться як про деякий формальний криптографічний протокол, в рамках якого одна сторона (*prover*) доводить іншій стороні (*verifier*), що знає певну інформацію (наприклад, секрет, пароль, приватний ключ), не розкриваючи саму інформацію та не даючи можливості її обчислити.

1.1. Інтерактивні та неінтерактивні схеми доказу з нульовим розголошенням

Докази з нульовим розголошенням/знанням є проривом у прикладній криптографії, оскільки вони обіцяють підвищити безпеку інформації для окремих осіб. Спробуйте довести іншій особі/стороні (наприклад, представнику закону) свою заяву/твердження (наприклад, «Я громадянин України»). Для цього потрібно буде надати «докази», що підтверджують це твердження, наприклад, національний паспорт або посвідчення водія. Але цей підхід має недоліки, основний з яких – відсутність конфіденційності. Зокрема, інформація, що дозволяє ідентифікувати особистість і що передається стороннім службам/сервісам, як правило, зберігається в центральних базах даних, які є вразливими для злому. І оскільки крадіжка особистих даних стає гострою проблемою, тому потрібні захищені засоби обміну конфіденційною інформацією.

Докази з нульовим розголошенням вирішують цю проблему, позбавляючи необхідності розкривати інформацію для доказу істинності тверджень. Протокол ZKP використовує твердження (*statement*), зване свідком (*witness*), для генерації короткого доказу його достовірності. Цей доказ гарантує, що твердження є достовірним, без розкриття інформації, що використовується при його створенні.

Повертаючись до нашого прикладу, наведеного вище, єдиним доказом, який необхідний для підтвердження заяви про громадянство (у разі використання можливостей концепції ZK), є доказ з нульовим розголошенням. Верифікатор повинен лише перевірити, чи є істинними певні характеристики доказу, щоб переконатися у істинності основного твердження. Питання – як це здійснити?

Як було зазначено вище, доказ із нульовим розголошенням дозволяє довести істинність/правдивість твердження/висловлювання, не розкриваючи його змісту і не повідомляю-

чи, як вдалося виявити цю істину. Щоб зробити це можливим, протоколи ZKP покладаються на алгоритми, які приймають певні дані як вхідні та повертають результат «істинне» або «неправдиве». При цьому протокол ZKP повинен відповідати наступним критеріям (володіти властивостями) [20, 21, 26, 28, 29]:

1) *повноти* (*completeness*). Якщо вхідні дані є вірними, протокол завжди повинен повертати значення «істина». Отже, якщо твердження, що лежить в його основі, вірно, а той, хто доказує, і той, хто перевіряє, діють чесно, то доказ може бути прийнятий;

2) о*бґрунтованості/коректності* (*soundness*). Якщо вхідні дані не вірні, то теоретично неможливо обдурити протокол, щоб він повернув результат «істина» (іншими словами, якщо твердження хибне, жоден той, хто доказує, не зможе переконати того, хто перевіряє, у його істинності). Отже, нечесний той, хто доказує, не зможе обдурити чесного того, хто перевіряє, і змусити його повірити в істинність невірного твердження (за винятком мізерної ймовірності);

3) *нульового розголошення* (*zero-knowledge*). Той, хто перевіряє, нічого не дізнається про твердження, крім його істинності або помилковості (у нього «нульове знання» про твердження). Ця вимога також не дозволяє тому, хто перевіряє, відновити вихідні дані (зміст твердження) з доказу.

В основі доказу з нульовим знанням лежать три основні компоненти: *свідок/свідчення* (*witness*), *виклик* (*challenge*), *відповідь* (*response*).

Свідок. За допомогою доказу з нульовим розголошенням той, хто доводить, хоче довести знання деякої прихованої інформації. Секретна інформація є «свідком/свідченням» доказу, а передбачуване знання того, хто доводить (*prover*) «свідка/свідчення» визначає набір деяких питань, на які може відповісти тільки сторона, яка знає цю інформацію. Таким чином, той, хто доводить, починає процес доказу з випадкового вибору питання, обчислення відповіді та відправки його верифікатору.

Виклик. Той, хто перевіряє (*verifier* – верифікатор), випадково вибирає інше питання з набору і просить того, хто доводить, відповісти на нього.

Відповідь. Той, хто доводить, приймає питання, обчислює відповідь і повертає її верифікатору. Відповідь того, хто доводить, дозволяє верифікатору зрозуміти, чи справді він мав доступ до «свідка». Щоб переконатися, що той, хто доводить, не вгадує наосліп і не отримує правильні відповіді випадково, верифікатор/перевіряльник (*verifier*) ставить додаткові питання. При багаторазовому повторенні такої взаємодії можливість того, хто доводить підробити знання про «свідка», значно падає. Все повторюється доти, доки верифікатор не буде задоволений.

Все сказане вище визначає так звану схему інтерактивного доказу з нульовим розголошенням (*Interactive Zero-Knowledge Proof* – IZKP).

У ранніх протоколах нульового розголошення використовувалися *інтерактивні докази*, де перевірка достовірності твердження вимагала двостороннього зв'язку між тим, хто доводить та тим, хто перевіряє. Класичними прикладами таких протоколів є: протокол Фіата-Шаміра (Fiat-Shamir Protocol) – використовується для автентифікації, де *prover* доводить знання секретного ключа без його розкриття [30]; протокол Гольдвассера–Мікалі–Ракоффа (GoldWasser–Micali–Rackoff (GMW) Protocol) [7]. Інтерактивні докази з нульовим розголошенням є потужним інструментом розробки криптографічних протоколів [31]. Їхня принципова особливість полягає в тому, що:

– вони вимагають кількох раундів комунікації між стороною, що доводить (*prover*), і стороною, що перевіряє (*verifier*), для обміну інформацією та перевірки доказу;

– у процесі обміну активну участь бере той, хто перевіряє, надсилаючи запити тому, хто доводить, та отримуючи відповіді.

Спрощене подання процесу взаємодії між стороною, що доводить і перевіряє, в інтерактивних доказах з нульовим розголошенням наведено на рис. 1.

Його суть полягає в наступному:

– той, хто доводить, генерує зобов'язання/комітмент (*commitment* – криптографічний примітив, який дозволяє одній стороні (*prover*) зафіксувати (*commit*) певне значення або дані (наприклад, секрет або проміжний результат) таким чином, щоб це значення не можна було змінити після фіксації, але при цьому воно залишається прихованим від іншої сторони (*verifier*) до певного моменту) та відправляє його тому, хто перевіряє;

– той, хто перевіряє, надсилає випадковий виклик (*challenge*);

– той, хто доводить, обчислює відповідь (*response*) на основі виклику;

– той, хто перевіряє, верифікує відповідь (залежно від результату перевірки доказ може бути прийнятим або відхиленим).

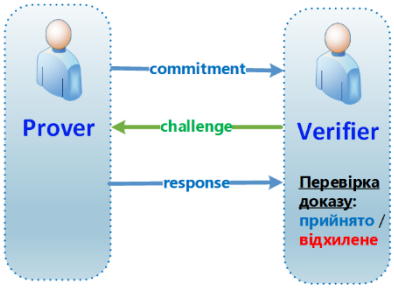

Рис. 1. Процес взаємодії між стороною, що доводить і перевіряє, в IZKP

Цикли *Commitment↔Challenge↔Response* дозволяють підвищити безпеку за рахунок багаторазових раундів. А саме, випадковий виклик (*challenge*) запобігає витоку інформації (той, хто перевіряє, не може передбачити відповідь заздалегідь), до того ж, якщо зловмисник не знає секрету, ймовірність підробити відповідь прагне до нуля після декількох раундів.

Однак, незважаючи на революційність, інтерактивний доказ мав досить обмежене застосування, оскільки вимагав від двох сторін постійної доступності та багаторазової взаємодії. Навіть якщо той, хто перевіряє, переконувався у чесності того, хто доводить, доказ був недоступний для незалежної перевірки (обчислення нового доказу вимагало нового набору повідомлень між тим, хто доводить і хто перевіряє). До того ж, IZKP не завжди могли задовольнити вимоги окремих застосунків, наприклад, вони не підходять для застосування у блокчейні, оскільки потреба у повторних взаємодіях у цьому випадку є недоцільною.

Для вирішення цієї проблеми було запропоновано інший тип ZKP, а саме неінтерактивні ZKP, в яких доказ може бути перевірено за один раз (раунд). Перші *неінтерактивні докази* з нульовим розголошенням (*Non-Interactive Zero-Knowledge Proof* – NIZKP), у яких той, хто доводить, і той, хто перевіряє, мають так званий «*загальний ключ*» (*common random string* – CRS), було викладено у роботі [21].

Ключова різниця між інтерактивними та неінтерактивними ZKP полягає в тому, що останні замінюють випадкові виклики верифікатора загальним еталонним значенням, що дозволяє передавати доказ третім особам [23].

Неінтерактивні докази з нульовим розголошенням, загалом:

Рис. 2. Процес взаємодії між сторонами, що доводить і перевіряє, в NIZKP

– передбачають передачу одного повідомлення/доказу (*proof*) від сторони, що доводить, до сторони, що перевіряє (рис. 2), що містить всю необхідну інформацію для доказу, причому сам доказ створюється один раз;

– не вимагають подальшої взаємодії між тим, хто перевіряє і хто доводить; той, хто перевіряє, може перевірити доказ самостійно в будь-який час і без участі сторони, що доводить;

– відмінно підходять для застосунків, наприклад, блокчейну (але не тільки), оскільки знижують потребу у додатковій комунікації, підвищують ефективність та можуть бути перевірені будь-яким учасником мережі.

Іншими словами, на відміну від інтерактивних, неінтерактивні докази вимагають лише одного раунду комунікації між учасниками (тим, хто доводить, та тим, хто перевіряє). Той, хто доводить, передає конфіденційну/чутливу інформацію до спеціального алгоритму для обчислення доказу з нульовим розголошенням. Цей доказ

(*proof*) відправляється верифікатору, який за допомогою іншого алгоритму перевіряє, чи знає той, хто доводить, цю конфіденційну/чутливу інформацію. Неінтерактивний доказ скорочує комунікацію між тим, хто доводить, і тим, хто перевіряє, що робить докази з нульовим розголошенням більш ефективними. Більше того, після генерації доказу він доступний для всіх інших (з доступом до так званого загального ключа та алгоритму верифікації) для перевірки.

Неінтерактивні докази стали проривом у розвитку технології нульового розголошення та послужили поштовхом до створення систем доказів, що використовуються сьогодні.

1.2. Основні класи неінтерактивних доказів із нульовим розголошенням

Основними класами (протоколами) неінтерактивних доказів із нульовим розголошенням є:

1) Zk-SNARK (*Zero-Knowledge Succinct Non-Interactive Argument of Knowledge* – короткий неінтерактивний аргумент знання з нульовим розголошенням) – клас неінтерактивного доказу з нульовим розголошенням/знанням, який є коротким та ефективним з точки зору вартості зберігання, а також, що дозволяє використовувати результат обчислення як твердження. Зазвичай використовується у застосунках, де потрібна компактність (компактні докази, що вимагають мінімальних обчислень) та відсутня інтерактивність.

Протокол zk-SNARK має такі характеристики (кожна частина назви вказує на відповідну характеристику):

– *нульове розголошення* (*zero-knowledge*). Той, хто доводить, може показати верифікатору, що він володіє певною інформацією, не надаючи саму інформацію. Той, хто перевіряє, може перевірити/підтвердити достовірність твердження без будь-яких інших відомостей про твердження (той, хто перевіряє, не отримує жодної інформації про сам секрет, крім факту його знання). Єдине, що той, хто перевіряє, знає про твердження – це те, що воно є істинним або хибним/неправдивим;

– *короткість* (*succinct*). Розмір доказу невеликий (займає мало місця), що дозволяє швидко та легко проводити перевірку;

– *неінтерактивність* (*non-interactive*). Доказ не інтерактивний, тому що той, хто доказує та перевіряє, взаємодіють лише один раз на відміну від інтерактивних доказів, які вимагають кілька раундів комунікації;

– *аргумент* (*argument*). Доказ задовольняє вимогу «обґрунтованості/коректності, надійності» (*soundness*), тому обман є вкрай малоймовірним. Ця частина назви додає якість обчислювальної надійності. Простіше кажучи, зловмисник навряд чи зможе обдурити систему, засновану на zk-SNARK, не маючи знань, що підтверджують його твердження (тобто не маючи базової інформації, яку він намагається підробити). Це засноване на теорії, що зловмисник має обмежені обчислювальні можливості, а значить, будь-яка людина з необмеженими обчислювальними можливостями може створити фальшиві докази. У деяких протоколах zk-SNARK є спосіб запобігти такій атаці [32];

– *знання* (*knowledge*). Доказ із нульовим розголошенням не може бути побудовано без доступу до секретної інформації (свідку). Обчислити вірний доказ з нульовим розголошенням для того, хто доводить, що не знає «свідка», важко, якщо взагалі можливо.

В цілому ж zk-SNARK властиві такі характерні особливості:

– компактні докази (кілька сотень байт). Як наслідок процес верифікації займає мілісекунди;

– необхідність фази довіреного налаштування (*Trusted Setup*). Потрібна одноразова генерація загальних параметрів (*proving key* та *verification key*). При компрометації параметрів фази *Trusted Setup* порушується безпека;

– нульове розголошення ґрунтується на обчислювальній складності (*computational zero-knowledge*);

– вразливість до квантових атак.

Характеризуючи zk-SNARK, слід зазначити, що з розвитком неінтерактивних доказів з нульовим розголошенням загалом та zk-SNARK, зокрема, виникла потреба в більш гнучких

системах, які могли б з одного боку підтримувати *універсальність* (використовувати один і той же набір публічних параметрів (CRS/SRS; CRS – *Common Reference String – загальнодоступний набір параметрів*; SRS – *Structured Reference String – структурований довідковий рядок*) для доказів різних програм/виразів (множини схем), а не тільки для однієї схеми, а з іншого – бути *оновлюваними* (тобто мати можливість оновлення CRS/SRS (будь-який учасник може оновити CRS/SRS) без компрометації безпеки (навіть один чесний учасник у ланцюжку оновлень робить всю систему безпечною)). У зв'язку з цим з'явилися так звані *універсальні* та *оновлювані* докази з нульовим розголошенням (*Universal and Updatable NIZKP*) [33–35].

Якщо дотримуватися строгої формальної класифікації, *Universal* та *Updatable* (*UU*) NIZKP можна розглядати як просунуте рішення (підкатегорію) zk-SNARK, оскільки їх архітектура та підхід до побудови доказів відрізняються. А саме в *Universal* та в *Updatable* NIZKP використовуються:

– більш гнучкі структури, наприклад, такі як *аргументи перестановки* (*permutation arguments*), наприклад, у PLONK [33]), які використовуються замість QAP (*Quadratic Arithmetic Program*), або *поліноміальні тотожності* (*polynomial identity* – це математичне твердження про рівність двох поліноміальних виразів, яке справедливе для всіх значень змінних, незалежно від констант або коефіцієнтів), наприклад, у Marlin [34, 35], що робить їх більш універсальними;

– більш складні поліноміальні зобов'язання (наприклад, KZG-схема, яка широко використовується в таких сучасних NIZKP як: PLONK, Marlin та інших, для забезпечення компактних доказів та швидкої перевірки). KZG-схема (*Kate-Zaverucha-Goldberg scheme*) [36] – це криптографічна конструкція, що є схемою поліноміальних зобов'язань (*polynomial commitment scheme*), яка дозволяє ефективно доводити і перевіряти властивості поліномів, такі як їх значення в певній точці, без розкриття самого полінома.

Наявність можливості оновлення CRS/SRS робить *Universal* та *Updatable* NIZKP безпечнішими для довгострокового використання в децентралізованих системах, знижує ризик компрометації.

2) Zk-STARK (*Zero-Knowledge Scalable Transparent Argument of Knowledge* – прозорий аргумент знання з нульовим розголошенням) – клас неінтерактивного доказу з нульовим розголошенням/знанням. Zk-STARK схожі на zk-SNARK, за винятком того, що вони мають такі відмінні властивості:

– масштабованість (*Scalable*). Zk-STARK швидше ніж zk-SNARK, генерує та перевіряє докази зі збільшенням розміру свідка. При використанні доказів STARK час доказу та перевірки лише трохи збільшується зі збільшенням розміру/величини свідка (час доказу та перевірки SNARK збільшується лінійно зі збільшенням розміру свідка);

– прозорість (*Transparent*). Zk-STARK покладається на публічно контрольовану випадковість при генерації публічних параметрів для доказу та перевірки замість довіреного налаштування (тобто не потрібна фаза *Trusted Setup*). Це, як заведено говорити, робить їх більш прозорими порівняно з zk-SNARK;

– Zk-STARK дозволяють створювати складніші докази (великих розмірів – зазвичай кілька кілобайт) ніж zk-SNARK, а це означає, що вони зазвичай мають вищі накладні витрати на перевірку. Однак у деяких випадках (наприклад, при доказі великих наборів даних) zk-STARK може бути ефективнішим/вигіднішим, ніж zk-SNARK;

– нульове розголошення засноване на статистичній невідмінності (*Statistical Zero-Knowledge*);

– квантово-стійкі (використовують геш-функції замість еліптичних кривих).

3) Bulletproofs – це клас NIZKP, спочатку розроблений для доказів діапазону (*range proofs*; тобто для підтвердження того, що деяке число лежить у певному діапазоні, наприклад, баланс – не негативний), але пізніше адаптований для більш загальних завдань.

Особливості Bulletproofs:

– заснований на дискретному логарифмі, але не використовує парне відображення;
– не вимагає фази довіреного налаштування (Trusted Setup);
– підходить для конфіденційних транзакцій;
– повільна перевірка (особливо за великої кількості доказів).

У табл. 1 для порівняння наведено деякі основні характеристики, властиві згаданим вище класам (протоколам) неінтерактивних доказів з нульовим розголошенням, а саме:

– *розмір доказу* – який обсяг даних потрібний для доказу;
– *час перевірки* – скільки часу займає перевірка доказу тим, хто перевіряє;
– *вимога присутності фази довіреного налаштування* – наявність/відсутність (прозорість) фази *Trusted Setup* та її особливості (універсальний, оновлюваний *Setup*);
– *тип нульового розголошення* – ідеальний (*Perfect*), статистичний (*Statistical*), обчислювальний з нульовим розголошенням (*Computational Zero-Knowledge*);
– *криптографічні припущення* – на яких припущеннях ґрунтується схема. *Криптографічні припущення* (*cryptographic assumptions*) – це математичні гіпотези, твердження, що лежать в основі безпеки криптографічних алгоритмів та протоколів. Вони формують фундамент довіри до систем захисту даних (стійкості різних схем), припускаючи, що певні завдання обчислювально складні для вирішення їх зловмисниками (наприклад, завдання факторизації великих чисел, дискретного логарифму, квадратичних лишків, Learning with Error (LWE) тощо). Припущення мають першорядне значення для криптографії [37];
– *стійкість до квантових атак* – чи може схема протистояти квантовим атакам (обчисленням);
– *застосування* – основні сфери використання;
– *приклади* – деякі конкретні реалізації.

Таблиця 1

Характеристики основних класів NIZKP

Характеристика	**ZK-SNARK**		**ZK-STARK**	**Bulletproofs**
	Класичні	***Universal & Updatable (UU)***		
Розмір доказу	~128–300 байт	~200–500 байт	~50–200 КБ	~1–10 КБ
Час перевірки	Дуже швидкий ($O(1)$, ~мс)	Швидкий ($O(1)$, ~мс)	Швидкий ($O(\log n)$, ~десятки мс)	Повільніший ($O(\log n)$, ~ сотні мс)
Фаза довіреного налаштування (Trusted Setup)	Є	Є (універсальний та оновлюваний)	Немає (Transparent)	Немає (Transparent)
Тип нульового розголошення	Computational	Computational	Statistical	Statistical
Криптографічні припущення	Парне відображення, дискретний логарифм	Парне відображення, KZG	Геші, поліноміальні зобов'язання	Дискретний логарифм
Стійкість до квантових атак	Вразливий	Вразливий	Стійкий	Вразливий
Застосування	Анонімні транзакції (Zcash)	Універсальні блокчейни (Mina)	Масштабування (StarkNet)	Конфіденційність (Monero)
Приклади	Groth16, Pinocchio, Sonic	PLONK, Marlin, Halo 2	StarkWare STARK (zk-STARK), Zilch	Bulletproofs

В результаті порівняльного аналізу представлених у табл. 1 даних можна зробити висновок, що:

– Zk-SNARK (як класичні, так і UU NIZKP) лідирують за компактністю доказів та швидкістю перевірки, що робить їх ідеальними для застосунків, де важлива ефективність з цієї точки зору (наприклад, блокчейни);

– Zk-STARK та Bulletproofs виграють у прозорості, тому що не вимагають фази довіреного налаштування, що підвищує безпеку системи;

– тільки zk-STARK протокол стійкий до квантових атак, що робить його кращим для довгострокових рішень;

– UU NIZKP (як розширення zk-SNARK) виділяються своєю універсальністю та оновлюваністю, що дозволяє використовувати один *Setup* для безлічі схем;

– кожен клас NIZKP має свою нішу: zk-SNARK класичний – для анонімності, UU NIZKP – для універсальних програм; zk-STARK – для масштабування; Bulletproofs – для конфіденційності.

У табл. 2 наведено список ZKP систем [38], а також їх порівняння з іншого боку, а саме на основі *прозорості* (*transparent*), *універсальності* (*universality*), *стійкості* (*resilience*) до відомих атак квантових комп'ютерів, парадигми програмування та року публікації. Щодо парадигми програмування, то розглядаються такі рішення: *використання арифметичних схем* (*arithmetic circuits* – AC), *мови асемблера* (A), *процедурного* (P) та *об'єктно-орієнтованого програмування* (OOP) відповідно.

Таблиця 2

Порівняльний аналіз сучасних систем ZKP

ZKP система	Рік	Протокол	Прозорість	Універсальність	Стійкість	Парадигма
Pinocchio [25]	2013	zk-SNARK	Немає	Немає	Немає	P
Geppetto [39]	2015	zk-SNARK	Немає	Немає	Немає	P
TinyRAM [40]	2013	zk-SNARK	Немає	Немає	Немає	P
Buffet [41]	2015	zk-SNARK	Немає	Немає	Немає	P
ZoKrates [42]	2018	zk-SNARK	Немає	Немає	Немає	P
xJsnark [43]	2018	zk-SNARK	Немає	Немає	Немає	P
vnTinyRAM [26]	2014	zk-SNARK	Немає	Є	Немає	P
MIRAGE [44]	2020	zk-SNARK	Немає	Є	Немає	AC
Sonic [45]	2019	zk-SNARK	Немає	Є	Немає	AC
Marlin [34]	2020	zk-SNARK	Немає	Є	Немає	AC
PLONK [33]	2019	zk-SNARK	Немає	Є	Немає	AC
Bulletproofs [46]	2018	Bulletproofs	Є	Є	Немає	AC
SuperSonic [47]	2020	zk-SNARK	Є	Є	Немає	AC
Hyrax [48]	2018	zk-SNARK	Є	Є	Немає	AC
Ligero [49]	2017	zk-SNARK	Є	Є	Є	AC
zk-STARK [50]	2019	zk-STARK	Є	Є	Є	A
Zilch [38]	2021	zk-STARK	Є	Є	Є	OOP

Слід зазначити, що у табл. 1 zk-SNARK системи позиціонувалися як непрозорі (тобто які вимагають фази довіреного налаштування), а у табл. 2 такі системи як Hyrax, Ligero, SuperSonic позиціонуються як прозорі. Насправді це вірно, вони є винятками класичних zk-SNARK, таких як Groth16 [51]. Hyrax [48] – це ZKP система класу zk-SNARK, яка заснована на дискретному логарифмі та використовує Fiat-Shamir перетворення для неінтерактивності. Hyrax дійсно не вимагає *Trusted Setup*, покладаючись на публічну випадковість моделі випадкового оракула (*Random Oracle Model*). Ligero [49] – це ще одна ZKP система, яка є прозорою. Вона заснована на інтерактивних оракулових доказах (Interactive Oracle Proofs – IOP) і використовує Fiat-Shamir перетворення для неінтерактивності, не вимагаючи *Trusted Setup*. Ligero також не зовсім типовий представник класу zk-SNARK, тому що його докази мають сублінійний розмір $O(\sqrt{n})$, що більше, ніж у класичних zk-SNARK. SuperSonic [47] – це ZKP система, яка досягає прозорості за рахунок використання інтерактивних оракулових доказів (IOP) та поліноміальних зобов'язань. Вона мінімізує криптографічні припущення та не вимагає *Trusted Setup*. SuperSonic – це новітня розробка, яка виходить за рамки класичних

zk-SNARK, таких як Groth16 [51], і є перехідним класом між zk-SNARK та іншими прозорими схемами.

Розгляд основних класів (протоколів) неінтерактивних ZKP був би неповним без протоколів доказів з нульовим розголошенням на основі решіток (*Lattice-Based Zero-Knowledge Proofs*) [52–54], що розвиваються сьогодні в контексті постквантової криптографії. Криптографічні докази з нульовим розголошенням на основі решіток – це вид доказів із нульовим розголошенням, який використовує складність обчислень над решітками для забезпечення конфіденційності та безпеки, зокрема для захисту від квантових комп'ютерів. Lattice-Based ZKP ґрунтуються на таких завданнях:

– SIS (*Short Integer Solution*) – завдання пошуку короткого вектора рішень у системі лінійних рівнянь над кінцевим полем, що використовується у криптографії для побудови деяких схем з відкритим ключем;

– LWE (*Learning With Errors*) – завдання пошуку прихованого вектора в системі лінійних рівнянь з додаванням «помилок», що ускладнює її вирішення та робить її основою для багатьох постквантових криптосистем;

– SVP (*Shortest Vector Problem*) – завдання пошуку найкоротшого ненульового вектора в решітках, яка є основою для багатьох криптографічних примітивів (складна для вирішення навіть для квантових комп'ютерів).

Перевагою Lattice-Based ZKP є забезпечення безпеки від квантових комп'ютерів (тобто забезпечують так звану постквантову безпеку/стійкість). Однак платою за це поки є велика складність та розмір доказу – десятки–сотні кілобайт або, навіть, мегабайти [53, 54].

У табл. 3 наведено порівняльну характеристику сучасних неінтерактивних систем доказів з нульовим розголошенням, включаючи системи Lattice-Based ZKP, за основними важливими характеристиками.

Таблиця 3

Порівняльна характеристика сучасних NIZKP

Характеристика	**ZK-SNARK**	**ZK-STARK**	**Bulletproofs**	**Lattice-Based ZKP**
Криптографічні припущення	Парне відображення, дискретний логарифм, KZG	Геші, поліноміальні зобов'язання	Дискретний логарифм	Решітки (SIS, LWE, SVP)
Ключові особливості	Компактні докази, швидкий верифікатор, потребують Trusted Setup	Без Trusted Setup, стійкі до квантових атак, масштабовані	Компактні докази діапазону, без Trusted Setup, використовуються у конфіденційних транзакціях	Постквантова безпека, стійкі до квантових атак, засновані на завданнях решіток
Розмір доказу	~128–500 байт	~50–200 kB	~1–10 kB	~100 kB – MB
Швидкість верифікації	Дуже висока (мілісекунди)	Середня (секунди)	Висока (мілісекунди)	Низька/середня (секунди)
Недоліки	Вимагають Trusted Setup, обмежена постквантова безпека	Більший розмір доказу, повільніша генерація	Відносно висока вартість обчислень	Більш громіздкі докази, складніші у практичній реалізації

І все ж, незважаючи на існування різних типів протоколів з нульовим розголошенням, у тому числі NIZKP-протоколів, протоколи zk-SNARK, завдяки їх короткому розміру доказу та сублінійному часу перевірки, сьогодні залишаються більш популярними [55]. Сімейство zk-SNARK протоколів в даний час є найбільш зрілою і широко використовуваною технологією, математичні основи якої – *системи обмежень рангу* 1 (*Rank-1 Constraint System* – R1CS), *програми квадратичної арифметики* (*Quadratic Arithmetic Program* – QAP – це інтерпретація арифметичної схеми, яка дозволяє стороні, що доводить, побудувати доказ [26, 27]), та *білінійні відображення* – є важливими елементами сучасної криптографії, що потребують детального розгляду. Привабливою властивістю zk-SNARK є те, що відповідна сторона, що

доводить і перевіряє, може бути автоматично згенерована на основі арифметичної схеми обчислень. Тому розглянемо їх ґрунтовніше.

2. Технологія zk-SNARK: концептуальне представлення

Спочатку розглянемо у найзагальнішому та найпростішому поданні основні етапи (процеси) та дії відповідних учасників інформаційного обміну при неінтерактивному доказі з нульовим розголошенням відповідно до протоколу zk-SNARK.

2.1. Основні етапи неінтерактивного доказу з нульовим розголошенням протоколу zk-SNARK

На рис. 3 представлено спрощену діаграму станів (*state diagram*) процесу неінтерактивного доказу з нульовим розголошенням для найбільш зрілого, широко використовуваного, ефективного з точки зору розміру доказу і часу верифікації, незалежно від розміру завдання, класу неінтерактивних систем ZKP – класу zk-SNARK протоколів.

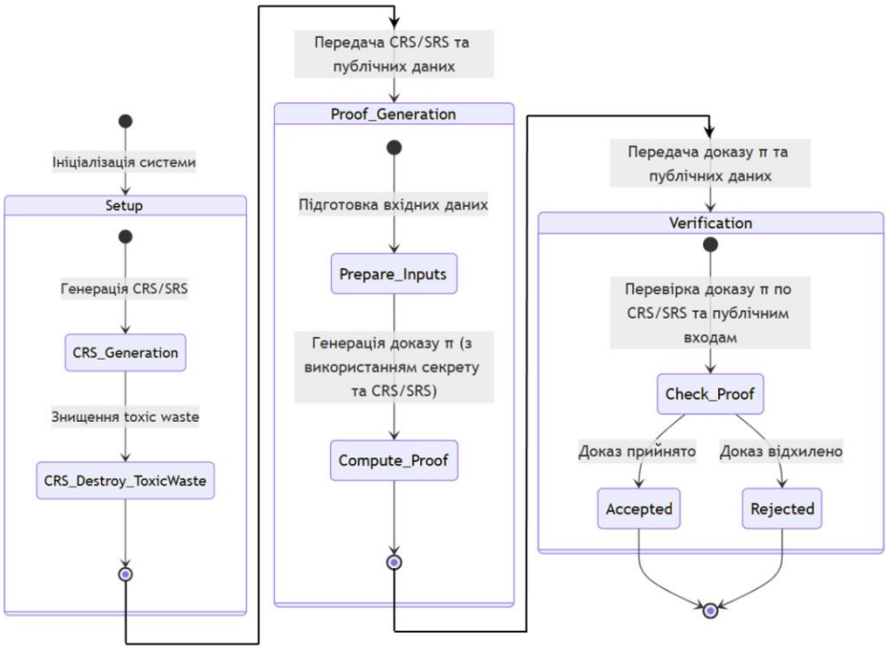

Рис. 3. Діаграма станів процесу неінтерактивного доказу з нульовим розголошенням

Наведемо короткий (на концептуальному рівні) опис цієї діаграми та надамо деякі важливі в даному контексті визначення та пояснення.

Етап 1. *Генерація загального ключа* (*Trusted Setup* або просто *Setup*). У контексті сучасних реалізацій неінтерактивних доказів з нульовим розголошенням, так званий, «*загальний ключ*» означає *загальнодоступний набір параметрів* (*Common Reference String* – CRS – рядок загального посилання (він же – загальний довідковий рядок, головний рядок посилань)), який використовується як стороною, що доводить (*prover*), так і стороною, що перевіряє (*verifier*). Це дозволяє продемонструвати своє знання деякої інформації (свідка – *witness*) без надання самої інформації.

У документі спільноти ZKP [9] термін CRS визначається як рядок, що формується алгоритмом-генератором NIZKP і доступний як стороні, що доводить, так і стороні, що перевіряє. CRS складається із параметрів доказу та параметрів перевірки. Може представлятися як URS або SRS.

URS (*Uniform Random String – уніфікований випадковий рядок*) – це загальний довідковий рядок, створений шляхом рівномірної вибірки з деякого простору і, зокрема, не має секретів при створенні (раніше Uniform Random String також називався Common Random String,

але через збіги з абревіатурою Common Reference String у подальшому, сьогодні намагаються уникати терміну Common Random String).

SRS (*Structured Reference String – структурований довідковий рядок*) – загальний довідковий рядок, створений шляхом вибірки з деякого складного розподілу, часто з використанням алгоритму вибірки з внутрішньою випадковістю, яка не повинна бути розкрита, оскільки це створить люк, що дозволяє створювати переконливі докази помилкових тверджень. SRS може бути неуніверсальним (залежати від конкретного відношення) або універсальним (не залежати від відношення, тобто служити для доказу будь-якого завдання з класу NP (Nondeterministic Polynomial time)).

Таким чином, можна помітити, що термін *Common Reference String* (CRS) – це більш загальне поняття. CRS – це будь-який публічний рядок (набір даних), відомий усім учасникам криптографічного протоколу. Він може бути випадковою, структурованою або навіть просто довільною послідовністю бітів. *Structured Reference String* (SRS) – це окремий випадок CRS, а саме, CRS, що має певну структуру [56]. Тому для коректності доцільно використовувати термін CRS, коли структура не є важливою, і SRS – коли потрібна певна математична форма публічних даних. Хоча в існуючих релевантних джерелах не завжди враховується специфіка цих термінів і використовуються обидва терміни (як синоніми) у сенсі «загального ключа» як загальнодоступного набору параметрів, що використовується стороною, що доводить і що перевіряє, для виконання протоколу (що можна пояснити їх зв'язком з історією розвитку, практичною близькістю, відсутністю суворої стандартизації і, просто, щоб не перевантажувати читача деталями). Далі також будуть використовуватися обидва ці терміни, але в конкретних схемах і прикладах для коректності використовуватиметься більш точний термін.

Генерація CRS (у широкому значенні, у тому числі, і окремий її випадок – SRS), як етап наведеної на рис. 3 діаграми, є конфіденційною операцією через її важливість для безпеки протоколу, яка виконується один раз (з використанням генератора випадкових чисел).

У процесі формування CRS/SRS мають місце так звані «*токсичні відходи*» ("*toxic waste*"), які можуть бути використані зловмисником для створення підроблених доказів, які будуть прийняті верифікатором. У зв'язку з цим *toxic waste* підлягають обов'язковому знищенню після генерації CRS/SRS.

Учасником цього етапу (назвемо його *довіреним органом початкової установки – Trusted Setup Authority – TSA*) є або окрема чесна особа/суб'єкт, що здійснює довірене налаштування (Trusted Setup) або кілька учасників (рисунок 4), які використовують так званий протокол конфіденційного (багатостороннього) обчислення (*multi-party computation protocol* – MPC, який також відомий як *secure computation* або *privacy-preserving computation*) для *Trusted Setup*.

Рис. 4. Довірене налаштування за допомогою протоколу MPC

Протокол конфіденційного обчислення – це спосіб зменшити ризики при генерації публічних параметрів. Мета безпечних *багатосторонніх обчислень* (MPC) – дати можливість групі незалежних власників даних, які не довіряють один одному або будь-якій спільній третій стороні, спільно обчислювати функцію, яка залежить від усіх їх приватних вхідних даних. MPC відрізняється від аутсорсингових обчислень тим, що всі учасники протоколу є власниками даних, які беруть участь у виконанні протоколу [57].

Безпечне генерування параметрів, зокрема для zk-SNARK, є найважливішим кроком у забезпеченні надійності результуючої системи

доказів. А можливість відкриття церемонії (генерація публічних параметрів для zk-SNARK називається «*церемонією налаштування*» (*setup ceremony*) [55]) для великої кількості учасників знижує ймовірність того, що отримані параметри будуть нечесними [58]. Використовуючи протокол MPC, можна децентралізовано запустити довірене налаштування, щоб відповідальність була розділена між усіма її учасниками. Під час цього процесу відбувається генерація випадкових чисел (секретів), їхнє шифрування, використання для генерації даних, а потім видалення, щоб забезпечити захист протоколу від несанкціонованого доступу. Якщо хоча б один учасник буде чесним і видалить свою частину «токсичних відходів», то ніхто не зможе створити фальшивих доказів.

На вхід цього етапу надходять такі дані:

– *арифметична схема* (*arithmetic circuit*) – це структура, отримана після компіляції деякої функції/програми, написаної спеціальною мовою DSL (*Domain Specific Language* (предметно-орієнтована мова) – мова, специфічна для вирішення вузького кола завдань у конкретній предметній області (на противагу мовам загального призначення типу Java або C#), наприклад, ZoKrates, Circom і т. д. (див. табл. 4 [27])), яка описує обчислення, для якого потрібно генерувати докази (інакше кажучи, оригінальна програма (чи логічне твердження) компілюється у певну систему рівнянь). Арифметична схема представляє обчислення як мережу множень та додавань над кінцевим полем, де кожне ребро та вузол відповідає операції. Зокрема, арифметична схема приймає деякі вхідні параметри/сигнали, які мають значення в діапазоні від 0 до деякого простого числа p і виконує додавання та множення по модулю p (mod p). На виході кожного додавання та множення виходить деякий параметр, який характеризується або як проміжний параметр, або як вихідний параметр, що означає кінцевий загальнодоступний вихід;

Таблиця 4

Статистика про реалізацію різних DSL
для побудови арифметичних схем

DSL	Кількість проєктів	Кількість схем
Circom	518	≈6400
Leo	96	396
Zinc	31	944
Halo2	83	187
Plonky2	17	≈1500
Noir	66	544
Gnark	43	≈3600
ZoKrates	301	≈5900

– *параметр безпеки* k – це ціле число, яке визначає рівень безпеки системи, зазвичай пов'язаний з максимальним ступенем полінома або кількістю обмежень у QAP. Даний параметр впливає на розмір еліптичної кривої, порядок груп, довжину ключів, забезпечуючи необхідну криптографічну стійкість (щоб атаки, наприклад дискретний логарифм, були обчислювально складними). Наприклад, k може встановити кількість раундів або розмір публічних параметрів для захисту від помилок і підробок. Чим вище k, тим складніше зламати систему, але при цьому слід враховувати, що збільшуються і обчислювальні витрати (збільшується час генерації доказу та перевірки). Хоча k називається «параметром безпеки», його вплив на безпеку скоріше непрямий. Зазвичай k вибирають так, щоб схема могла вмістити потрібні обчислення, а безпека забезпечується вибором криптографічних параметрів (наприклад, кінцевого поля F_p, в якому виконуються всі обчислення, або груп як додаткових вхідних параметрів). У ZoKrates [58], наприклад, параметр k не задається явно, але він вбудований у процес компіляції та налаштування. ZoKrates автоматично вибирає відповідне k, щоб вмістити схему, але в деяких випадках можна зіткнутися з обмеженнями, якщо схема занадто велика.

Проміжні перетворення цього етапу

Ці перетворення (як попередній крок перед генерацією ключів, щоб адаптувати обчислення до криптографічних примітивів) виконуються відповідним довіреним (і) учасником або учасниками (TSA), які мають доступ до повних даних схеми і можуть генерувати відповідні структури (рис. 4).

Процес перетворень:

– *арифметична схема* (*Circuit*) → *R1CS*. Арифметична схема перетворюється на систему лінійних обмежень. Кожне арифметичне обмеження представляється як: $\langle a_i, s\rangle \cdot \langle b_i, s\rangle = \langle c_i, s\rangle$, де a_i, b_i, c_i – це вектори коефіцієнтів для *i*-го обмеження (*constraint*) в R1CS; s – це вектор змінних (включаючи вхідні, вихідні та допоміжні змінні схеми) – вектор свідка (в літературі zk-SNARK часто використовуються такі терміни для нього: *witness*, *witness vector*, *private input*, які можуть застосовуватися в різних контекстах (у доказах SNARK сторона, що доводить, використовує весь вектор, а не тільки приватну частину; сторона, що перевіряє, при перевірці знає тільки публічні входи і *verification keys*, тому верифікація не розкриває весь *witness*) і це, дійсно, може викликати плутанину в розумінні; щоб цього уникнути, наведемо більш розширений контекст цих термінів: *а*) визначення терміну *witness* у широкому розумінні (у контексті R1CS, QAP та реалізації доказів): ***witness***=*весь вектор значень* (включаючи вхідні, вихідні та допоміжні змінні схеми), що використовується при побудові доказу (фактично та коректно це ***witness vector***); *б*) визначення у вузькому розумінні: ***witness***=*лише секретні дані* (наприклад, *z* у прикладі $x^2+y^2=z^2$) – ці значення приховані від сторони, що перевіряє (іноді їх називають *private inputs*); символ $\langle ... \rangle$ у виразі $\langle a_i, s\rangle$ – це позначення скалярного добутку (*inner product*, *dot product*) двох векторів: $\langle a_i, s\rangle = \sum_{k=1}^{\Theta} a_k^{(i)} s_k$; $\langle a_i, s\rangle$ і $\langle b_i, s\rangle$ дають два числа, які множаться, результат порівнюється з $\langle c_i, s\rangle$ – очікуваним значенням; $k = \overline{1,\Theta}$ – це індекс векторів a_i, b_i, c_i, s, Θ – розмірність векторів a_i, b_i, c_i, s. Таким чином, R1CS представляє обчислення як набір таких обмежень, що спрощує їхню перевірку;

– *R1CS* → *QAP*. Програма квадратичної арифметики QAP є зручною для криптографії, оскільки дозволяє використовувати властивості поліномів (наприклад, ділимість або рівність) з парним відображенням. Кожне обмеження R1CS перетворюється на поліном, де перевіряється рівність двох поліномів у певній точці.

Генерація секретних параметрів

TSA генерує випадкове значення τ (або набір випадкових значень r_v, r_w, s, α_v, α_w, α_y, β, $\gamma \in \mathrm{F}$ з кінцевого поля F_p, як у протоколі, запропонованому в роботі [25]), або τ, ρ_A, ρ_B, α_A, α_B, α_C, β, $\gamma \in \mathrm{F}_\mathrm{r}$ як – у роботі [26]), яке буде секретним параметром. τ вибирається випадково з використанням криптографічно безпечного генератора випадкових чисел, щоб забезпечити непередбачуваність. У загальному випадку (і це доцільно робити) секретні параметри можуть генеруватися в багатосторонній церемонії (MPC), де кожен учасник $(x = \overline{1,\sigma})$ вносить своє випадкове значення: τ_1, τ_2, ..., τ_σ (тоді маємо $\tau_{mpc} = \tau_1 \cdot \tau_2 \cdot ... \cdot \tau_\sigma$), що мінімізує ризик компрометації. Це (ці) значення використовується для створення публічних параметрів, але воно(вони) не розкривається. τ генерується незалежно від арифметичної схеми або k, але його розмір та властивості залежать від обраного поля та параметрів безпеки (k). Наприклад, τ має бути досить великим, щоб утруднити обчислення дискретного логарифму.

τ використовується для створення загальнодоступного набору параметрів (CRS/SRS). На основі CRS/SRS надалі формуються *ключ доказу* (*Proving Key – PK*) та *ключ перевірки*

(*Verification Key – VK*). *PK* та *VK* – це спеціалізовані ключі, що містять елементи CRS/SRS, а також додаткові дані (елементи), необхідні для конкретної схеми доказів. SRS зазвичай будується на основі прихованих секретів і має строгу математичну структуру (наприклад, послідовність $\{g^{\tau^i}\}_{i=0}^{d}$, де g – генератор групи; d – ступінь полінома з QAP; τ – прихований параметр).

PK включає необхідні дані (зокрема елементи CRS/SRS) для генерації доказу, а *VK* – для перевірки, що дозволяють стороні, що перевіряє (*verifier*), переконатися, що доказ π коректний, не знаючи секретних вхідних даних. Зокрема, *PK* містить інформацію (набір елементів, наприклад, точок на еліптичній кривій, обчислених з використанням τ і QAP), яка допомагає стороні, що доводить, створити доказ. А *VK* включає значення, необхідні для перевірки *парного відображення* (*pairing*):

$$e: \mathrm{G}_1 \times \mathrm{G}_2 \to \mathrm{G}_T, \tag{1}$$

де G_1 та G_2 – дві циклічні групи порядку r (зазвичай еліптичні криві з різними рівнями ефективності); G_T – цільова група, де знаходиться результат парного відображення. Парне відображення дозволяє зв'язати елементи з G_1 та G_2 у обчисленнях, що критично для перевірки доказу. Поділ ролей між G_1 та G_2 пов'язано з асиметрією обчислень (наприклад, операції у G_1 можуть бути оптимізовані для *prover*, а у G_2 – для *verifier*).

Витік τ (або набору випадкових значень) підвищує можливість компрометації системи, оскільки зловмисник потенційно зможе підробляти докази. Тому після використання τ або набору випадкових значень їх (як «*токсичні відходи*» – *toxic waste*) слід знищити.

Результатом (*виходом*) цього *етапу* є згенерований загальнодоступний набір параметрів (CRS/SRS). CRS/SRS – це повний набір параметрів, з якого виділяються *PK* та *VK*. Слід зазначити, що у деяких протоколах *PK* і *VK* разом покривають весь CRS/SRS, але часто CRS/SRS містить додаткові елементи, які безпосередньо не входять ні у *PK*, ні *VK* (наприклад, для універсальних схем або схем, що оновлюються).

Е т а п 2 . *Створення доказу* (*Proof Generation*). Концептуально: сторона, що доводить, використовуючи CRS/SRS, публічне твердження, яке потрібно довести, секретні дані (*witness*), формує компактний доказ π (*proof*), який не розкриває секрет, але переконує того, хто перевіряє, у знанні секрету. Після цього сформований доказ передається стороні, що перевіряє. Якщо ж докладніше, то – на вхід даного етапу надходить:

– *Proving Key* (*PK*) – публічний параметр (частина CRS/SRS), згенерований на попередньому етапі (*Trusted Setup/Setup*), що містить інформацію про обчислення, що засновані на QAP, та секретні параметри;

– *публічне твердження* (так званий публічний вхід) x – дані, які відомі як *prover*, так і *verifier* (наприклад, вхідні дані функції $f(x, w)$, такі як x=2);

– *свідок/свідоцтво* (*witness*) w (так званий приватний вхід – *private input* – визначення *witness* у вузькому значенні) – секретна інформація, що підтверджує істинність твердження, і яку знає лише сторона, що доводить (наприклад, w=3, якщо функція $f(x, w) = x \cdot w$).

Далі той, хто доводить, бере x, w і перетворює їх у форму, що відповідає R1CS/QAP (це вже частково зроблено на етапі *Setup*, але *prover* використовує ці дані для обчислень). Наприклад, x і w представляються як вектор змінних, який задовольняє обмеженням R1CS. QAP ж, створена на етапі *Setup*, дозволяє уявити обчислення як поліноми. Той, хто доводить, обчислює поліноми, які описують як x і w задовольняють обмеження QAP. Ці поліноми пов'язані з публічними та приватними входами. Тобто, той, хто доводить, використовуючи *PK*, який містить елементи, пов'язані з τ (наприклад, точки на еліптичній кривій, такі як g^{τ^i}), x і w, формує доказ:

$$\pi = Prove(PK, x, w), \quad (2)$$

комбінуючи поліноми та дані з *PK* так, щоб показати, що обчислення коректні. Тобто *prover* доводить, що поліноми, пов'язані з x і w, задовольняють рівності, заданому QAP, але робить це в зашифрованому вигляді, щоб не розкривати w.

Приклад, якщо функція $f(x, w) = x \cdot w$, $x = 2$, $w = 3$, то той, хто доводить, створює π, яке доводить, що $f(x, w) = 6$, не розкриваючи w.

Результат (*вихід*) *етапу*. Компактний доказ π.

Етап 3. *Перевірка доказу* (*Verification*). Сторона, що перевіряє, отримує доказ π і здійснює його перевірку, використовуючи CRS/SRS та публічні дані/входи (перевіряє коректність твердження без розкриття секрету). Результатом перевірки є рішення про прийняття чи відхилення доказу.

На вхід цього етапу надходять такі дані:

– *Verification Key* (*VK*) – публічний параметр, згенерований на етапі *Trusted Setup*, що містить дані для перевірки.

– *Публічний вхід x* – ті самі дані, що використовував той, хто доказує (наприклад, *x*=2).

– *Доказ* π – компактний доказ, отриманий від *prover*.

Сторона, що перевіряє (*verifier*), використовуючи *x*, π, *VK*, здійснює перевірку коректності доказу:

$$Verify\ (VK, x, \pi) \in \{true, false\}, \quad (3)$$

де логічний результат: *true* – доказ коректний/прийнятий (*Accepted*); *false* – доказ відхилений (*Rejected*).

VK містить інформацію, яка дозволяє перевірити, що π відповідає *x* та обчисленням, заданим через QAP (тобто містить зашифровані відомості про структуру схеми та правила перевірки, дозволяючи обчислити доказ так, щоб він був валідований тільки для коректних *w* та *x*). Той, хто перевіряє, переконується, що поліноми, закодовані в π, задовольняють QAP без необхідності перераховувати всю функцію (QAP дозволяє *verifier* перевірити рівність поліномів через *VK*, не знаючи *w*). Перевірка зазвичай зводиться до обчислення та порівняння певних криптографічних виразів (наприклад, перевірка рівності за допомогою парного відображення: $e(A, B) = e(C, D)$, де A, B, C, D – це елементи криптографічних груп, що беруть участь у перевірці коректності доказу на етапі верифікації; *e* – парне відображення (*pairing*)).

Результат (*вихід*) *етапу*. Ухвалення рішення: доказ *прийнятий* (*Accepted*) або доказ *відхилений* (*Rejected*).

На основі сказаного далі на прикладі схеми (*scheme*) Groth16, запропонованої Jens Groth [51] як однієї з популярних реалізацій zk-SNARK, що широко використовуються в таких проєктах, як Zcash [59–61], Tornado Cash [62, 63], ZoKrates [42, 58], викладемо більш ґрунтовно (з наведенням відповідних математичних виразів та простих прикладів, що пояснюють суть виконуваних процесів) технологію неінтерактивних доказів з нульовим розголошенням.

2.2. Характеристика основних учасників, їх завдань та алгоритмів протоколу zk-SNARK

Основними учасниками та їх завданнями у zk-SNARK є:

– *окремий чесний учасник*, який здійснює довірене налаштування (*Trusted Setup*) або *кілька учасників*, які використовують протокол конфіденційного (багатостороннього) обчислення (MPC) для *Trusted Setup*.

Завдання: створення загальнодоступного набору параметрів (CRS/SRS).

– *Сторона, що доводить* (*Prover*), – учасник, який знає секретні вхідні дані (*witness / private input*) і хоче довести істинність твердження, не розкриваючи ці дані.

Завдання: створення доказу π.

– *Перевіряльник/сторона, що перевіряє* (*Verifier*) – учасник, який перевіряє доказ, щоб переконатися в істинності твердження, не знаючи секретних даних.

Завдання: перевірка доказу π.

Виходячи з описаного, а також, спираючись на релевантні джерела [44, 51, 61, 64, 65], було виявлено, що zk-SNARK складається з наступних основних алгоритмів:

- довіреного налаштування;
- створення доказу;
- перевірки доказу.

Розглянемо їх докладніше з наведенням відповідних математичних виразів і простих прикладів, що пояснюють суть виконуваних дій.

2.2.1. Алгоритм довіреного налаштування.

На вхід алгоритму *довіреного налаштування* (*генератора ключів*) надходить:

1. *Арифметична схема* (*arithmetic circuit*), яка описує обчислення, для якого потрібно згенерувати доказ. Наприклад, схема може перевіряти, чи знає користувач рішення деякого рівняння (наприклад, такого, як $x^2+y^2=z^2$) або чи знає користувач пароль *x*, геш якого дорівнює *H*: *hash(x)=H* і т. д.

Процес перетворення деякого оператора обчислення/обчислювального твердження (*computation statement*), що відповідає деякому рівнянню (наприклад, *hash(x)=H* або $x^2+y^2=z^2$) можна представити такою послідовністю дій (етапів):

Етап 1. Запис обчислювального твердження у вигляді деякої функції спеціальною мовою DSL.

Етап 2. Процедура «згладжування» (*flattening*). Перетворення отриманого оператора обчислення на кілька дрібніших операторів, які мають одну з двох форм: 1) *x=y* (де *y* – може бути змінною або числом/константою); 2) *x=y*(op)*z* (де op – один із наступних операторів: +, -, *, /; при цьому *y* та *z* можуть бути змінними, числами/константами або самостійними виразами [66]). Кожен із цих операторів можна уявити як свого роду логічний вентиль/гейт (*gate*) у схемі (*circuit*).

Етап 3. Перетворення отриманого на попередньому етапі результату до R1CS. Даний етап необхідний для приведення довільних операцій до стандартної форми обмежень (тобто будь-які обчислення виражаються як набір простих обмежень).

R1CS складається з трьох наборів векторів (*a, b, c*), кожен з яких відповідає певному обмеженню. Кожне обмеження має форму:

$$(a \cdot s) \times (b \cdot s) = (c \cdot s), \tag{4}$$

де $s = (s_0, s_1, s_2, \ldots, s_n)$ – *вектор свідка* (*witness vector*), що містить $s_0 = 1$ і всі змінні схеми s_i (входи/вхідні параметри (сигнали), проміжні значення/проміжні параметри, виходи/вихідні параметри), $i = \overline{o,n}$; $\cdot$ – знак скалярного добутку.

У роботі [51] автор, узагальнюючи арифметичні схеми, представив відповідні залежності/відносини у вигляді наступної системи рівнянь з множиною змінних:

$$\sum_{i=0}^{n} a_i u_{i,q} \times \sum_{i=0}^{n} a_i v_{i,q} = \sum_{i=0}^{n} a_i w_{i,q}, \tag{5}$$

де $i = \overline{o,n}$, $a_0 = 1$ (нульовий елемент завжди дорівнює 1, що дозволяє включати константи в лінійні комбінації), змінні $a_1, \ldots, a_n \in \mathrm{F}$; $u_{i,q}$, $v_{i,q}$, $w_{i,q} \in \mathrm{F}$ – константи в кінцевому полі F, що уточнюють *q*-е рівняння (коефіцієнти в кінцевому полі F, які задають лінійні комбінації для *q*-го обмеження, $q = \overline{1,m}$).

Даний вираз (5) можна спростити шляхом приведення його (використовуючи вираз (4)) до матричного вигляду

$$(A\cdot s)\,\text{e}\,(B\cdot s)=(C\cdot s), \tag{6}$$

де A, B, C – матриці розміру $m\times(n+1)$, m – число обмежень, $(n+1)$ – довжина вектора; причому введені у виразі (4) вектора a, b, c є складовими відповідних матриць: $a\in A$, $b\in B$, $c\in C$; e – поелементне/покомпонентне множення (добуток Адамара – Hadamard product); $(A\cdot s)$, $(B\cdot s)$, $(C\cdot s)$ – вектори-стовпці розміром $m\times 1$.

Між виразами (5) та (6) існують наступні відповідності:

у виразі (6) вектор свідка $s=(s_0,s_1,s_2,\ldots,s_n)$ відповідає вектору $a=(a_0,a_1,...,a_n)$ виразу (5);

у виразі (5) $u_{i,q}$, $v_{i,q}$, $w_{i,q}$ – це коефіцієнти для q-го обмеження, які задають лінійні комбінації. У (6) вони є відповідними елементами відповідних рядків матриць A, B, C:

q-й рядок матриці A складається з коефіцієнтів $u_{i,q}$, а саме i-й елемент q-го рядка матриці A являє собою наступне значення: $a_{q,i}=u_{i,q}$.

Аналогічно:

$b_{q,i}=v_{i,q}$ – це i-й елемент q-го рядка матриці;

$c_{q,i}=w_{i,q}$ – це i-й елемент q-го рядка матриці.

Таким чином, відповідні q-і рядки матриць A, B, C будуть мати такий вигляд:

q-й рядок матриці A: $(u_{0,q},u_{1,q},...,u_{n,q})$;

q-й рядок матриці B: $(v_{0,q},v_{1,q},...,v_{n,q})$;

q-й рядок матриці C: $(w_{0,q},w_{1,q},...,w_{n,q})$.

Тоді з урахуванням (5), (6) ($\sum_{i=0}^{n} a_i u_{i,q}=\sum_{i=0}^{n} u_{i,q}s_i$, при $s_0=1$) та введених вище позначень маємо:

$$(A\cdot s)_q=\sum_{i=0}^{n} u_{i,q}s_i=u_{0,q}\times 1+u_{1,q}\times s_1+\ldots+u_{n,q}\times s_n \tag{7}$$

Аналогічно:

$$(B\cdot s)_q=\sum_{i=0}^{n} v_{i,q}s_i=v_{0,q}\times 1+v_{1,q}\times s_1+\ldots+v_{n,q}\times s_n, \tag{8}$$

$$(C\cdot s)_q=\sum_{i=0}^{n} w_{i,q}s_i=w_{0,q}\times 1+w_{1,q}\times s_1+\ldots+w_{n,q}\times s_n \tag{9}$$

Рівняння для q-го обмеження має вигляд:

$$\sum_{i=0}^{n} u_{i,q}s_i\cdot\sum_{i=0}^{n} v_{i,q}s_i=\sum_{i=0}^{n} w_{i,q}s_i \tag{10}$$

або у матричній формі запису:

$$(A\cdot s)_q\,\text{e}\,(B\cdot s)_q=(C\cdot s)_q\Leftrightarrow(A_q\cdot s)\,\text{e}\,(B_q\cdot s)=(C_q\cdot s). \tag{11}$$

Це рівняння перевіряється поелементно для кожного $q=\overline{1,m}$.

Тоді для всіх обмежень система рівнянь якраз і матиме вигляд виразу (6).

Для кращого розуміння та демонстрації справедливості наведених вище виразів розглянемо задачу доказу знання рішення рівняння $x^2+y^2=z^2$ у кінцевому полі, яке дозволить переконати верифікатора (який знає лише z – публічний вхід) у істинності твердження, не розкриваючи при цьому секретні дані (x та y – приховані/секретні входи).

Т в е р д ж е н н я . Існує такі значення x і y (приховані входи, відомі тільки стороні, що доводить), що їх квадрати, складені разом, рівні квадрату публічного входу z в кінцевому полі p, тобто $x^2+y^2=z^2 \bmod p$.

Обмеження та вектор свідка (s) для доказу знання рішення рівняння $x^2+y^2=z^2$ мають вигляд:

$r_1 = x \times x$ – перше обмеження, де r_1 – проміжна змінна для x^2;

$r_2 = y \times y$ – друге обмеження, де r_2 – проміжна змінна для y^2;

$r_3 = r_1 + r_2$ – третє обмеження;

$r_4 = z \times z$ – четверте обмеження.

$s = (s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7)^T = (1, x, y, z, r_1, r_2, r_3, r_4)^T$; $n+1=8$, $m=4$.

Коефіцієнти $u_{i,q}$, $v_{i,q}$, $w_{i,q}$ (де $i \in \{0,1,...,7\}$, $q \in \{1,2,3,4\}$):

q=1 ($r_1 = x \times x$):

$u_{i,1}$: $u_{0,1}=0,\ u_{1,1}=1,\ u_{2,1}=0,\ u_{3,1}=0,\ u_{4,1}=0,\ u_{5,1}=0,\ u_{6,1}=0,\ u_{7,1}=0.$

$v_{i,1}$: $v_{0,1}=0,\ v_{1,1}=1,\ v_{2,1}=0,\ v_{3,1}=0,\ v_{4,1}=0,\ v_{5,1}=0,\ v_{6,1}=0,\ v_{7,1}=0.$

$w_{i,1}$: $w_{0,1}=0,\ w_{1,1}=0,\ w_{2,1}=0,\ w_{3,1}=0,\ w_{4,1}=1,\ w_{5,1}=0,\ w_{6,1}=0,\ w_{7,1}=0.$

q=2 ($r_2 = y \times y$):

$u_{i,2}$: $u_{0,2}=0,\ u_{1,2}=0,\ u_{2,2}=1,\ u_{3,2}=0,\ u_{4,2}=0,\ u_{5,2}=0,\ u_{6,2}=0,\ u_{7,2}=0.$

$v_{i,2}$: $v_{0,2}=0,\ v_{1,2}=0,\ v_{2,2}=1,\ v_{3,2}=0,\ v_{4,2}=0,\ v_{5,2}=0,\ v_{6,2}=0,\ v_{7,2}=0.$

$w_{i,2}$: $w_{0,2}=0,\ w_{1,2}=0,\ w_{2,2}=0,\ w_{3,2}=0,\ w_{4,2}=0,\ w_{5,2}=1, w_{6,2}=0, w_{7,2}=0.$

q=3 ($r_3 = r_1 + r_2$):

$u_{i,3}$: $u_{0,3}=0,\ u_{1,3}=0,\ u_{2,3}=0,\ u_{3,3}=0,\ u_{4,3}=1,\ u_{5,3}=1,\ u_{6,3}=0,\ u_{7,3}=0.$

$v_{i,3}$: $v_{0,3}=1,\ v_{1,3}=0,\ v_{2,3}=0,\ v_{3,3}=0,\ v_{4,3}=0,\ v_{5,3}=0,\ v_{6,3}=0,\ v_{7,3}=0.$

$w_{i,3}$: $w_{0,3}=0,\ w_{1,3}=0,\ w_{2,3}=0,\ w_{3,3}=0,\ w_{4,3}=0,\ w_{5,3}=0,\ w_{6,3}=1,\ w_{7,3}=0.$

q=4 ($r_4 = z \times z$):

$u_{i,4}$: $u_{0,4}=0,\ u_{1,4}=0,\ u_{2,4}=0,\ u_{3,4}=1,\ u_{4,4}=0,\ u_{5,4}=0,\ u_{6,4}=0,\ u_{7,4}=0.$

$v_{i,4}$: $v_{0,4}=0,\ v_{1,4}=0,\ v_{2,4}=0,\ v_{3,4}=1,\ v_{4,4}=0,\ v_{5,4}=0,\ v_{6,4}=0,\ v_{7,4}=0.$

$w_{i,4}$: $w_{0,4}=0,\ w_{1,4}=0,\ w_{2,4}=0,\ w_{3,4}=0,\ w_{4,4}=0, w_{5,4}=0, w_{6,4}=0,\ w_{7,4}=1.$

Матриці:

Матриця розміром (4×8) складається з коефіцієнтів $u_{i,q}$:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Рядки матриці A:

q=1: (0, 1, 0, 0, 0, 0, 0, 0) – для вибору x.

q=2: (0, 0, 1, 0, 0, 0, 0, 0) – для вибору y.

q=3: (0, 0, 0, 0, 1, 1, 0, 0) – для вибору r_1, r_2.

q=4: (0, 0, 0, 1, 0, 0, 0, 0) – для вибору z.

Матриця розміром (4×8) складається з коефіцієнтів $v_{i,q}$:

$$B=\begin{bmatrix}0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0\end{bmatrix}.$$

Рядки матриці *B*:

q=1: (0, 1, 0, 0, 0, 0, 0, 0) – для вибору *x*.

q=2: (0, 0, 1, 0, 0, 0, 0, 0) – для вибору *y*.

q=3: (1, 0, 0, 0, 0, 0, 0, 0) – для вибору константи 1.

q=4: (0, 0, 0, 1, 0, 0, 0, 0) – для вибору *z*.

Матриця розміром (4×8) складається з коефіцієнтів $w_{i,q}$:

$$C=\begin{bmatrix}0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}.$$

Рядки матриці *C*:

q=1: (0, 0, 0, 0, 1, 0, 0, 0) – для вибору r_1.

q=2: (0, 0, 0, 0, 0, 1, 0, 0) – для вибору r_2.

q=3: (0, 0, 0, 0, 0, 0, 1, 0) – для вибору r_3.

q=4: (0, 0, 0, 0, 0, 0, 0, 1) – для вибору r_4.

Перевірка обчислень:

$(A\cdot s)$:

$(A\cdot s)_1 = 0\times 1+1\times x+0\times y+0\times z+0\times r_1+0\times r_2+0\times r_3+0\times r_4 = x$;

$(A\cdot s)_2 = y$; $(A\cdot s)_3 = r_1+r_2$; $(A\cdot s)_4 = z$.

Результат: $(A\cdot s)=(x, y, r_1+r_2, z)^T$.

$(B\cdot s)$:

$(B\cdot s)_1 = x$; $(B\cdot s)_2 = y$; $(B\cdot s)_3 = 1\times 1+0\times x+0\times y+0\times z+0\times r_1+0\times r_2+0\times r_3+0\times r_4 = 1$;

$(B\cdot s)_4 = z$.

Результат: $(B\cdot s)=(x, y, 1, z)^T$.

$(C\cdot s)$:

$(C\cdot s)_1 = r_1$; $(C\cdot s)_2 = r_2$; $(C\cdot s)_3 = r_3$; $(C\cdot s)_4 = r_4$.

Результат: $(C\cdot s)=(r_1, r_2, r_3, r_4)^T$.

Тоді:

$(A\cdot s)\,\text{e}\ (B\cdot s)=(x\times x, y\times y, (r_1+r_2)\times 1, z\times z)=(x^2, y^2, r_1+r_2, z^2)$.

А оскільки $r_1=x^2$, $r_2=y^2$, $r_3=r_1+r_2$, $r_4=z^2$, то отримуємо $(A\cdot s)\,\text{e}\ (B\cdot s)=(C\cdot s)$, що й потрібно було довести.

Таким чином, $(A\cdot s)\,\text{e}\ (B\cdot s)=(r_1, r_2, r_3, r_3)=C\cdot s$.

Е т а п 4. Перетворення отриманої R1CS у форму QAP, яка реалізує ту ж логіку, але замість використання векторів зі скалярним добутком будуть використовуватися поліноми. QAP дозволяє використовувати властивості поліномів (зокрема, ділимість одного полінома на інший) для ефективної побудови доказів у еліптичних кривих, тобто застосовувати ефективні криптографічні методи (*pairings*) та компактні, швидкі докази, що критично для zk-SNARK.

Основна ідея перетворення полягає в наступному:

R1CS встановлює систему обмежень у вигляді $(A\cdot s)$ е $(B\cdot s)=C\cdot s$, де s – вектор свідка, а $A(x),\ B(x),\ C(x)$ – матриці, що визначають лінійні комбінації змінних;

QAP перетворює ці лінійні обмеження на поліноміальну форму. Це дозволяє перейти від перевірки векторних рівнянь до перевірки рівності поліномів, що є зручним для криптографії (зокрема, при використанні пар еліптичних кривих).

Загальний процес перетворення можна представити у вигляді таких дій (кроків):

1. Вибір точок $x_1, x_2, ..., x_m$ та побудова полінома $Z(x)$.

2. Побудова для кожної вибраної точки x_q ($q=\overline{1,m}$) поліномів Лагранжа $l_q(x)$.

3. Побудова поліномів $A_i(x),\ B_i(x),\ C_i(x)$, використовуючи значення відповідних i-х стовпців відповідних матриць $A,\ B,\ C$, отриманих на попередньому етапі 3 (для системи R1CS).

4. Побудова загального виду поліномів $A(x),\ B(x),\ C(x)$ з урахуванням вектора s.

5. Перевірка рівняння QAP:

$$A(x)\times B(x)-C(x)=H(x)\times Z(x) \tag{12}$$

де $A(x),\ B(x),\ C(x)$ – поліноми, що представляють лінійні комбінації елементів вектора з коефіцієнтами, що залежать від x; $Z(x)$ – поліном, що задає точки, у яких перевіряються обмеження (його коріння – це $x_1, x_2, ..., x_m$); $H(x)$ – поліном, який є часткою від ділення та підтверджує, що обмеження виконуються.

Якщо загальний вигляд цього рівняння відомий заздалегідь, то його конкретні поліноми головним чином є результатом перетворення R1CS в поліноміальну форму, тобто формуються в процесі перетворення.

Розглянемо далі докладніше процес перетворення «*R1CS→рівняння QAP*» та перевірки останніх.

К р о к 1. Отже, на першому кроці спочатку необхідно для кожного обмеження $q=\overline{1,m}$ зіставити унікальну точку $x_q\in \mathrm{F}$ та побудувати поліном $Z(x)$:

$$Z(x)=\prod_{q=1}^{m}(x-x_q). \tag{13}$$

Цей поліном (ступеня m – позначимо як – $\deg Z(x)=m$) називають *цільовим поліномом* (*target polynomial*) [67]. Він визначає точки x_q, де перевіряються обмеження. У точках, що відповідають обмеженням R1CS (для всіх $q=\overline{1,m}$), $Z(x_q)=0$, в інших точках – $Z(x_q)\neq 0$. Точки x_q обираються довільно, але вони мають бути різними.

Для демонстрації справедливості наведених виразів знову звернемося до завдання доказу знання рішення рівняння $x^2+y^2=z^2$.

Отже, нехай:

- кількість обмежень – 4 ($m=4$);
- точки (x_q, де $q=\overline{1,m}$): $x_1=1,\ x_2=2,\ x_3=3,\ x_4=4$;
- ступінь полінома $Z(x)$ – $\deg Z(x)=4$ ($Z(x)=(x-1)(x-2)(x-3)(x-4)$).

К р о к 2. Для кожної точки x_q будується поліном Лагранжа $l_q(x)$, який дорівнює 1 у x_q і 0 – у решті точках:

$$l_q(x) = \prod_{q=1, k\neq q}^{m-1} \frac{x - x_k}{x_q - x_k}. \quad (14)$$

Для будь-якого $q = \overline{1,m}$ ступінь $l_q(x)$ дорівнює $(m-1)$ (у добутку $(m-1)$ множників) та

$$l_q(x) = \begin{matrix} 0, & k \neq q, \\ 1, & k = q. \end{matrix}$$

Приклад (для обраного завдання ($x^2+y^2=z^2$) та вибраних точок x):

$$l_1(x) = \frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} = -\frac{(x-2)(x-3)(x-4)}{6}.$$

$$l_2(x) = \frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} = \frac{(x-1)(x-3)(x-4)}{2}.$$

$$l_3(x) = \frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} = -\frac{(x-1)(x-2)(x-4)}{2}.$$

$$l_4(x) = \frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} = \frac{(x-1)(x-2)(x-3)}{6}.$$

Ступінь кожного $l_q(x)$ дорівнює 3 ($\deg l_q(x) = 3$).

К р о к 3. Після визначення полінома Лагранжа будуються поліноми $A_i(x)$, $B_i(x)$, $C_i(x)$ ($i \in \{0,1,...,n\}$). Поліноми формуються як лінійні комбінації поліномів Лагранжа з коефіцієнтами з рядків матриць *A*, *B* та *C*:

$$A_i(x) = \sum_{q=1}^{m} a_{q,i} l_q(x); \quad B_i(x) = \sum_{q=1}^{m} b_{q,i} l_q(x); \quad C_i(x) = \sum_{q=1}^{m} c_{q,i} l_q(x), \quad (15)$$

де $a_{q,i}$, $b_{q,i}$, $c_{q,i}$ – елементи *q*-го рядка та *i*-го стовпця матриць A, B, C; $(x_q) = a_{q,i}$; $B_i(x_q) = b_{q,i}$; $C_i(x_q) = c_{q,i}$; ступінь $A_i(x), B_i(x), C_i(x) \le m-1$, оскільки $\deg l_q(x) = m-1$.

Для випадку, що розглядається, для кожної змінної *i* (відповідної s_i):

– для *i*=0 (константа 1):

$a_{q,0} = 0,0,0,0$ – стовпець 0 матриці *A*; $A_0(x) = 0$;

$b_{q,0} = 0,0,1,0$ – стовпець 0 матриці *B*;

$$B_0(x) = 0 \times l_1(x) + 0 \times l_2(x) + 1 \times l_3(x) + 0 \times l_4(x) = l_3(x) = -\frac{(x-1)(x-2)(x-4)}{2};$$

$c_{q,0} = 0,0,0,0$ – стовпець 0 матриці *C*; $C_0(x) = 0$.

Ступені: $\deg A_0(x) = 0$; $\deg B_0(x) = 3$; $\deg C_0(x) = 0$;

– для *i*=1 (змінна *x*):

$a_{q,1} = 1,0,0,0$ – стовпець 1 матриці *A*; $A_1(x) = l_1(x) = -\frac{(x-2)(x-3)(x-4)}{6}$;

$b_{q,1} = 1,0,0,0$ – стовпець 1 матриці *B*; $B_1(x) = l_1(x)$;

$c_{q,1} = 0,0,0,0$ – стовпець 1 матриці *C*;

Ступені: $\deg A_1(x) = 3$; $\deg B_1(x) = 3$; $\deg C_1(x) = 0$;

– для *i*=2 (змінна *y*):

$a_{q,2} = (0,1,0,0)$; $A_2(x) = l_2(x) = \frac{(x-1)(x-3)(x-4)}{2}$;

$b_{q,2}=(0,1,0,0);\ B_2(x)=l_2(x);\ c_{q,2}=\ 0,0,0,0\ ;\ C_2(x)=0.$

Ступені: $\deg A_2(x)=3;\ \deg B_2(x)=3;\ \deg C_2(x)=0;$

– для i=3 (змінна z):

$$a_{q,3}=(0,0,0,1);\ A_3(x)=l_4(x)=\frac{(x-1)(x-2)(x-3)}{6};$$

$b_{q,3}=(0,0,0,1);\ B_3(x)=l_4(x);\ c_{q,3}=(0,0,0,0);\ C_3(x)=0.$

Ступені: $\deg A_3(x)=3;\ \deg B_3(x)=3;\ \deg C_3(x)=0;$

– для i=4 (змінна r_1):

$$a_{q,4}=(0,0,1,0);\ A_4(x)=l_3(x)=-\frac{(x-1)(x-2)(x-4)}{2};$$

$$b_{q,4}=(0,0,0,0);\ B_4(x)=0;\ c_{q,4}=(1,0,0,0);\ C_4(x)=l_1(x)=-\frac{(x-2)(x-3)(x-4)}{6}.$$

Ступені: $\deg A_4(x)=3;\ \deg B_4(x)=0;\ \deg C_4(x)=0;$

– для i=5 (змінна r_2):

$$a_{q,5}=(0,0,1,0);\ A_5(x)=l_3(x)=-\frac{(x-1)(x-2)(x-4)}{2};$$

$$b_{q,5}=(0,0,0,0);\ B_5(x)=0;\ c_{q,5}=(0,1,0,0);\ C_5(x)=l_2(x)=\frac{(x-1)(x-3)(x-4)}{2}.$$

Ступені: $\deg A_5(x)=3;\ \deg B_5(x)=0;\ \deg C_5(x)=3;$

– для i=6 (змінна r_3):

$$a_{q,6}=(0,0,0,0);\ A_6(x)=0;\ b_{q,6}=(0,0,0,0);\ B_6(x)=0;$$

$$c_{q,6}=(0,0,1,0);\ C_6(x)=l_3(x)=-\frac{(x-1)(x-2)(x-4)}{2}.$$

Ступені: $\deg A_6(x)=0;\ \deg B_6(x)=0;\ \deg C_6(x)=3;$

– для i=7 (змінна r_4):

$$a_{q,7}=(0,0,0,0);\ A_7(x)=0;\ b_{q,7}=(0,0,0,0);\ B_7(x)=0;$$

$$c_{q,7}=(0,0,0,1);\ C_7(x)=l_4(x)=\frac{(x-1)(x-2)(x-3)}{6}.$$

К р о к 4 . На підставі отриманих поліномів $A_i(x)$, $B_i(x)$, $C_i(x)$ загальний вигляд поліномів $A(x)$, $B(x)$, $C(x)$ буде таким:

$$A(x)=\sum_{i=0}^{n}s_iA_i(x);\ \ B(x)=\sum_{i=0}^{n}s_iB_i(x);\ \ C(x)=\sum_{i=0}^{n}s_iC_i(x),\qquad(16)$$

де s_i – елементи вектора свідка.

Для даного випадку матимемо наступний загальний вигляд поліномів $A(x)$, $B(x)$, $C(x)$ з урахуванням вектора свідка $s=(s_0,s_1,s_2,s_3,s_4,s_5,s_6,s_7)^T$:

$$\begin{aligned}A(x)=\sum_{i=0}^{7}s_i\cdot A_i(x)&=s_0\cdot A_0(x)+s_1\cdot A_1(x)+s_2\cdot A_2(x)+s_3\cdot A_3(x)+s_4\cdot A_4(x)+\\&+s_5\cdot A_5(x)+s_6\cdot A_6(x)+s_7\cdot A_7(x)=s_0\cdot 0-s_1\cdot\frac{(x-2)(x-3)(x-4)}{6}+\\&+s_2\cdot\frac{(x-1)(x-3)(x-4)}{2}+s_3\cdot\frac{(x-1)(x-2)(x-3)}{6}-s_4\cdot\frac{(x-1)(x-2)(x-4)}{2}-\\&-s_5\cdot\frac{(x-1)(x-2)(x-4)}{2}+s_6\cdot 0+s_7\cdot 0;\end{aligned}$$

$$
\begin{aligned}
B(x) &= \sum_{i=0}^{7} s_i \cdot B_i(x) = s_0 \cdot B_0(x) + s_1 \cdot B_1(x) + s_2 \cdot B_2(x) + s_3 \cdot B_3(x) + s_4 \cdot B_4(x) \\
&+ s_5 \cdot B_5(x) + s_6 \cdot B_6(x) + s_7 \cdot B_7(x) = s_0 \cdot \frac{-(x-1)(x-2)(x-4)}{2} - \\
&- s_1 \cdot \frac{(x-2)(x-3)(x-4)}{6} + s_2 \cdot \frac{(x-1)(x-3)(x-4)}{2} + \\
&+ s_3 \cdot \frac{(x-1)(x-2)(x-3)}{6} + s_4 \cdot 0 + s_5 \cdot 0 + s_6 \cdot 0 + s_7 \cdot 0; \\
C(x) &= \sum_{i=0}^{7} s_i \cdot c_i(x) = s_0 \cdot C_0(x) + s_1 \cdot C_1(x) + s_2 \cdot C_2(x) + s_3 \cdot C_3(x) + s_4 \cdot C_4(x) + \\
&+ s_5 \cdot C_5(x) + s_6 \cdot C_6(x) + s_7 \cdot C_7(x) = s_0 \cdot 0 + s_1 \cdot 0 + s_2 \cdot 0 + s_3 \cdot 0 - \\
&- s_4 \cdot \frac{(x-2)(x-3)(x-4)}{6} + s_5 \cdot \frac{(x-1)(x-3)(x-4)}{2} - \\
&- s_6 \cdot \frac{(x-1)(x-2)(x-4)}{2} + s_7 \cdot \frac{(x-1)(x-2)(x-3)}{6}.
\end{aligned}
\tag{17}
$$

У схемах на базі кінцевих полів потрібний поділ по модулю (див. вираз (17)), який зводиться до множення на зворотний/мультиплікативний зворотний елемент (або інверсію). Інверсією називається такий елемент $a^{-1} \in \mathrm{F}_p$, що $a \times a^{-1} \equiv 1 \bmod p$, де 1 – мультиплікативна одиниця поля. У полі завжди існує (і він єдиний) мультиплікативний зворотний для кожного ненульового елемента. У цьому випадку визначимо інверсії для 2 і 6. Для цього скористаємося Малою теоремою Ферма. Мала теорема Ферма стверджує, що з простого числа p і будь-якого цілого a, що не ділиться на p, є істинним вираз: $a^{p-1} \equiv 1 \bmod p$. Звідси випливає, що інверсія обчислюється як

$$
a^{-1} \equiv a^{p-2} \bmod p. \tag{18}
$$

Застосовуючи формулу (18) (для простоти подальших розрахунків скористаємося полем з малою кількістю елементів, наприклад, використовуємо кінцеве поле порядку 17: p=17 – F_{17}), отримуємо: $2^{-1} \equiv 2^{17-2} \bmod 17 = 9$, а $6^{-1} \equiv 6^{17-2} \bmod 17 = 3$.

Тоді для прикладу, що розглядається (схеми $x^2+y^2=z^2$), і введеного спрощення (нехай x=3, y=4, тоді z=5, так як $3^2+4^2=9+16=25=5^2$), враховуючи, що 25 mod 17=8, вектор свідка буде мати наступний вигляд: $s = (s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7)^T = (1, x, y, z, r_1, r_2, r_3, r_4)^T = (1,3,4,5,9,16,8,8)$, а загальний вигляд, наприклад, полінома $A(x)$ після відповідних перетворень матиме такий вигляд:

$$
\begin{aligned}
A(x) &= 8(x-2)(x-3)(x-4) + 2(x-1)(x-3)(x-4) + 15(x-1)(x-2)(x-3) + 4(x-1)(x-2)(x-4) + \\
&+ 9(x-1)(x-2)(x-4).
\end{aligned}
$$

Аналогічним чином знаходяться поліноми $B(x)$, $C(x)$.

Слід зауважити, що подібні обчислення поліномів $A(x)$, $B(x)$, $C(x)$ здійснюються щоразу при зміні значень вектора свідка (для інших значень x, y, z, наприклад, x=1, y=2, $z = \sqrt{5}$, оскільки $1^2+2^2=1+4=5=(\sqrt{5})^2$).

У точці x_q: $A(x_q) = \sum_{i=0}^{n} s_i a_{q,i} = A_q s$; $B(x_q) = B_q s$; $C(x_q) = C_q s$.

І якщо вираз $(A \cdot s) \mathrm{e} (B \cdot s) = (C \cdot s)$ для R1CS виконується, то це означає, що вираз $A(x_q)B(x_q) = C(x_q)$ справедливий для всіх q.

К р о к 5 . Перевірка рівняння QAP.

Оскільки коректно отримано, що $A(x_q)B(x_q)=C(x_q)$ для всіх q, то також можна коректно записати: $P(x)=A(x)B(x)-C(x)=0$, причому даний поліном $P(x)$ можна розділити на поліном $Z(x)=\prod_{q=1}^{m}(x-x_q)$. А оскільки відомо, якщо деякий поліном $P(x)$ ступеня $k\geq m$ має різне коріння в точках $x_1, x_2, ..., x_m$, то він ділиться без залишку (М) на поліном $Z(x)$ ступеня m з цими коренями: $P(x)\text{М}Z(x)=\prod_{q=1}^{m}(x-x_q)$. Це прямий наслідок теореми Безу та властивостей ділимості у кільці багаточленів. Виходячи з сказаного, слідує, що існує такий поліном $H(x)$, який виходить в результаті ділення без залишку полінома $P(x)$ на поліном $Z(x)$:

$$H(x)=\frac{A(x)\times B(x)-C(x)}{Z(x)}. \tag{19}$$

Тим самим ми підтвердили справедливість виразу (12).

Щодо полінома $H(x)$ слід зауважити, що його називають *частка від ділення полінома/многочлена* (*quotient polynomial*). $\deg H(x)\leq(m-1)$, так як $\deg A(x)\leq(m-1)$ і $\deg B(x)\leq(m-1)$, $A(x)\times B(x)$ має ступінь $\leq 2(m-1)$, а $\deg C(x)\leq(m-1)$ відповідно ступінь виразу $A(x)\times B(x)-C(x)$ $-\leq 2(m-1)$, при врахуванні того, що $\deg Z(x)=m$, отже, ступінь $\deg H(x)\leq 2(m-1)-m=m-2$. Якщо $H(x)$ існує (і його ступінь «правильна»/очікувана – $\leq m-2$), то R1CS коректно перетворюється на QAP.

Таким чином, можна зробити висновок, що якщо раніше в R1CS для перевірки рівняння потрібно було перевіряти обчислення покомпонентного/поелементного множення (скалярного добутку) для кожного обмеження окремо, тепер, використовуючи поліноми, можна перевірити всі обмеження відразу. У цьому випадку сторона, що доводить, намагатиметься переконати сторону, що перевіряє, через справедливість рівняння (12), тобто за допомогою того факту, що поліном $P(x)=A(x)B(x)-C(x)$ ділиться без залишку на поліном $Z(x)$ у точці τ. Якщо це твердження правильне, значить сторона, що доводить, знає рішення задачі (наприклад, у цьому випадку сторона, що доводить, знає коректні x і y, що задовольняють заданому рівнянню $x^2+y^2=z^2$), не розголошуючи самих значень.

На основі наведених вище теоретичних відомостей та виразів, а також обраної схеми, наведемо спрощений приклад (з поданням деяких конкретних значень для кращого розуміння процесу) формування Structured Reference String (SRS), використовуючи деяке згенероване Trusted Setup Authority (TSA) випадкове значення τ у полі F_p (де p – просте число). А саме:

- схема, що використовується: $x^2+y^2=z^2$;
- поліноми QAP: вираз (17);
- цільовий поліном: $Z(x)=(x-1)(x-2)(x-3)(x-4)$, $\deg Z(x)=4$;
- параметр безпеки k: k=4 (оскільки $\deg Z(x)=m=k=4$, то максимальний ступінь полінома $H(x)$ до 4: $\deg H(x)\leq m-2$);
- поле F_{17}: (кінцеве поле порядку 17: p=17).

Обчислення Structured Reference String (SRS)

SRS для Groth16 зазвичай містить послідовності ступенів τ у групах G_1 і G_2 з відповідними генераторами g і h порядку r. У контексті кінцевого поля F_{17} генератор групи (g або h) – це елемент (його також називають примітивний елемент), який породжує всю

мультиплікативну групу (всі ненульові елементи поля: {1, 2, …, 16}) через свої ступені (g^1, g^2, ..., g^{16}) ($\{g^1 \mod 17,\ g^2 \bmod 17,\ldots,\ g^{16} \bmod 17\} = \mathrm{F}_{17}$). Порядок примітивного елемента/генератора $g \in \mathrm{F}_p^*$ – це найменше позитивне число r, таке що $g^r \equiv 1 \bmod p$ (якщо цей порядок дорівнює $r=p-1$, то g – примітивний елемент/генератор).

Тобто для згенерованого на етапі *Trusted Setup* секретного значення τ і генераторів $g \in \mathrm{G}_1$, $h \in \mathrm{G}_2$ обчислюються елементи: $\{g^{\tau^i}\}_{i=0}^{d}$, $\{h^{\tau^i}\}_{i=0}^{d}$, де d – ступінь полінома з QAP ($d = m$). В даному випадку це: $\{g^{\tau^i}\}_{i=0}^{4}$, $\{h^{\tau^i}\}_{i=0}^{4}$ (хоча також можуть обчислюватися і додаткові елементи, що залежать від схеми, наприклад: $g^{\alpha}, g^{\beta}, g^{\delta}$; $h^{\beta}, h^{\gamma}, h^{\delta}$, де $\tau, \alpha, \beta, \gamma, \delta \in \mathrm{F}_p$ – секретні параметри, невідомі нікому, крім TSA, але в даному спрощеному демонстраційному прикладі вони (їх розрахунки) не наводяться).

Для визначеності, нехай $\tau=7$ (випадкове значення у F_{17}, $7<17$). Генератори $g=3$ (для G_1) і $h=5$ (для G_2) з порядком 16 (оскільки для мультикативної групи F_{17}: p-1=17-1=16; $3^{16} \equiv 1 \bmod 17$; $5^{16} \equiv 1 \bmod 17$). Для коректності слід зазначити, що у реальних zk-SNARK (зокрема Groth16) криптографічні групи G_1 і G_2 реалізуються за допомогою еліптичних кривих (наприклад, BN254, BLS12-381, BLS12_377 і т. д. [51, 68–71], і елементи CRS/SRS – це точки на еліптичній кривій (наприклад, $P = g^{\tau^i}$), а не просто числа.

Отже, для спрощеного прикладу, що розглядається, обчислимо спочатку g^{τ^i}, виходячи з того, що ступінь обчислюється з урахуванням порядку групи, а саме, $g^{\tau \bmod (p-1)} \bmod p$ (у нашому випадку $\bmod\ (p-1) = \bmod\ 16$). Тоді:

$g^{\tau^0} = 3^{7^0} = 3^1 = 3$; $g^{\tau^1} = 3^{7^1} = 3^7 \bmod 17 = 11$; $g^{\tau^2} = 3^{7^2} = 3^{49 \bmod 16} \bmod 17 = 3^1 \bmod 17 = 3$;

$g^{\tau^3} = 3^{7^3} = 3^{343 \bmod 16} \bmod 17 = 11$; $g^{\tau^4} = 3^{7^4} = 3^{2401 \bmod 16} \bmod 17 = 3^1 \bmod 17 = 3$.

Слід зауважити, що при обчисленні показників ступенів g^{τ^i} у групі порядку 16 показники будуть циклічні з періодом 2 для $\tau=7$ (це просто окремий випадок).

Тоді $\{g^{\tau^i}\}_{i=0}^{4} = \{3,11,3,11,3\}$.

Аналогічно обчислимо $\{h^{\tau^i}\}_{i=0}^{4}$: $h^{\tau^0} = 5^{7^0} = 5^1 = 5$; $h^{\tau^1} = 5^{7^1} = 5^7 \bmod 17 = 10$;

$h^{\tau^2} = 5^{7^2} = 5^{49 \bmod 16} \bmod 17 = 5^1 \bmod 17 = 5$; $h^{\tau^3} = 5^{7^3} = 5^{343 \bmod 16} \bmod 17 = 10$;

$h^{\tau^4} = 5^{7^4} = 5^{2401 \bmod 16} \bmod 17 = 5$.

Тоді $\{h^{\tau^i}\}_{i=0}^{4} = \{5,10,5,10,5\}$.

Таким чином, ми отримали основу обох ключів: *PK* і *VK* –

$\{g^{\tau^i}\}_{i=0}^{4} = \{g^1, g^7, g^1, g^7, g^1\} = \{3,11,3,11,3\}$, $\{h^{\tau^i}\}_{i=0}^{4} = \{h^1, h^7, h^1, h^7, h^1\} = \{5,10,5,10,5\}$.

У загальному випадку

$$SRS = \{\{g^{\tau^i}\}_{i=0}^{d};\ \{h^{\tau^i}\}_{i=0}^{d};\ g^{\alpha}, g^{\beta}, g^{\delta};\ h^{\beta}, h^{\gamma}, h^{\delta}\}. \qquad (20)$$

Формування Proving Key (PK)

PK включає елементи SRS, а також поліноміальні зобов'язання/комітменти (*commitments*) у точці τ, щоб *Prover* міг обчислити доказ, а саме:

$$PK = \{\{g^{A_i(\tau)}\}_{i=0}^{n}, \{g^{B_i(\tau)}\}_{i=0}^{n}, \{g^{C_i(\tau)}\}_{i=0}^{n}, \{h^{B_i(\tau)}\}_{i=0}^{n}; g^{\alpha}, h^{\beta}, g^{\delta}; \{g^{\tau^i}\}_{i=0}^{d}, \{h^{\tau^i}\}_{i=0}^{d}\}. \quad (21)$$

Елементи виду $g^{A_i(\tau)}$ ($g^{B_i(\tau)}, g^{C_i(\tau)}, h^{B_i(\tau)}$) – це поліноміальні зобов'язання, які обчислюються для відповідних базисних поліномів $A_i(x)$, $B_i(x)$, $C_i(x)$ у точці τ.

Поліноміальні зобов'язання (*commitments*) – це вид криптографічних *зобов'язань* (криптографічна примітивна конструкція, зашифроване (сховане) компактне уявлення полінома (наприклад, точка на еліптичній кривій)), що дозволяє зафіксувати деякий поліном (наприклад, $A(x)$, $B(x)$, $C(x)$ з QAP) так, щоб: *приховати сам поліном* (його структуру, коефіцієнти), *гарантувати незмінність* (сторона, що доводить не зможете потім змінити зобов'язання на інший поліном), *довести значення* (сторона, що доводить згодом може переконливо, компактно довести іншому учаснику, що значення зафіксованого полінома (наприклад, $A(x)$) у певній точці дійсно дорівнює $A(\tau)$), не розкриваючи структуру всього полінома. У zk-SNARK-системах поліноміальні зобов'язання застосовуються, щоб компактно доводити коректність обчислень без публікації самих обчислень.

Включення базисних елементів $g^{A_i(\tau)}, g^{B_i(\tau)}, g^{C_i(\tau)}$ дозволяє стороні, що доводить, обчислювати $g^{A(\tau)}, g^{B(\tau)}, g^{C(\tau)}$ для будь-якого вектора свідка s (*PK* повинен бути універсальним для всіх можливих векторів свідків s в рамках однієї схеми), використовуючи вираз

$$g^{A(\tau)} = g^{\sum_i s_i A_i(x)} = \prod_{i=0} (g^{A_i(\tau)})^{s_i}; \; g^{B(\tau)} = \prod_{i=0} (g^{B_i(\tau)})^{s_i}; \; g^{C(\tau)} = \prod_{i=0} (g^{C_i(\tau)})^{s_i}. \quad (22)$$

Вираз отримано на основі виразу (16) та властивостей ступенів $a^{xy} = (a^x)^y = (a^y)^x$; $a^{x+y} = a^x \times a^y$.

Елементи *PK* $g^{\alpha}, h^{\beta}, g^{\delta}$ – це спеціальні елементи групи, що згенеровані із секретних параметрів α, β, δ на етапі *Trusted Setup*. Вони використовуються стороною, що доводить, для побудови доказу, забезпечуючи коректність обчислень і захист від фальсифікацій.

Значення деяких елементів *PK* (для спрощеного демонстраційного прикладу з τ=7):

$\{g^{\tau^i}\}_{i=0}^{4} = \{g^1, g^7, g^1, g^7, g^1\} = \{3,11,3,11,3\}$; $\{h^{\tau^i}\}_{i=0}^{4} = \{h^1, h^7, h^1, h^7, h^1\} = \{5,10,5,10,5\}$;

$g^{A_0(7)} = 3^0 \mod 17 = 1$; $g^{A_1(7)} = 3^{l_1(7)} \mod 17 = 3^7 \mod 17 = 2187 \mod 17 = 11$;

$g^{A_2(7)} = 3^{l_2(7)} \mod 17 = 3^2 \mod 17 = 9$; $g^{A_3(7)} = 3^{l_4(7)} \mod 17 = 3^3 \mod 17 = 10$;

$g^{A_4(7)} = 3^{l_3(7)} \mod 17 = 3^6 \mod 17 = 15$; $g^{A_5(7)} = 3^{l_3(7)} \mod 17 = 3^6 \mod 17 = 15$;

$g^{A_6(7)} = 3^0 \mod 17 = 1$; $g^{A_7(7)} = 3^0 \mod 17 = 1$ і так далі.

Формування Verification Key (VK)

VK включає наступні елементи, необхідні стороні, що перевіряє, для перевірки коректності доказу за допомогою парних відображень (*pairings*) між елементами груп G_1 і G_2:

$$VK = \{g^{\alpha}, h^{\beta}, h^{\gamma}, h^{\delta}; \{L_k\}_{k=0}^{N}\}, \quad (23)$$

де $L_k = g^{l_k(\tau)}$ – зобов'язання до публічних входів, що обчислюються за поліномами Лагранжа $l_k(x)$, k – індекси публічних входів; N – число публічних входів з урахуванням константи (перша точка $L_0 = g^{l_0(\tau)}$ завжди відповідає константі та бере участь у верифікації як базова).

У нашому прикладі для фактично одного публічного входу (z), тобто N=1, матимемо два зобов'язання: $L_0 = g^{l_0(\tau)} = g^0 = 1$ (для константи; поліном Лагранжа в точці, що відповідає константі дорівнює 1) і $L_1 = g^{l_1(\tau)}$ (для z), де $l_1(x)$ – поліном Лагранжа для публічного входу z.

У зв'язку з тим, що у zk-SNARK (зокрема, у QAP-представленні) кожна змінна (включаючи публічні) відповідає певній позиції у векторі свідка, а також рядку матриць *A, B, C,* а вибір індексу полінома Лагранжа для публічного входу безпосередньо пов'язаний з індексом рядка матриць *A, B,* у яких цей вхід має участь, у прикладі, що розглядається, для вибору публічного входу z використовується рядок з індексом 4 (q=4). Тобто як зобов'язання публічного входу L_1 буде використовуватися поліном Лагранжа $l_4(x) = \frac{(x-1)(x-2)(x-3)}{6}$ (див. приклад вище). Таким чином, $L_1 = g^{l_4(\tau)}$ (індекс у виразі (23) k=1 відповідає індексу q=4 у виразі (14)).

2.2.2. Алгоритм створення доказу.

У розпорядженні сторони, що доводить, є наступні елементи (вхідні дані), необхідні для формування доказу:

– *Proving Key* (*PK*) – попередньо згенерований на етапі *Trusted Setup* ключ доказу, що включає зобов'язання до базисних поліномів $(g^{A_i(\tau)}, g^{B_i(\tau)}, h^{B_i(\tau)}, g^{C_i(\tau)})$, параметри $g^{\alpha}, h^{\beta}, g^{\delta}$ і т. д.;

– *публічні входи (твердження)* – відомі значення (значення публічних змінних, наприклад, z), які будуть використовуватися як стороною, що доводить, так і версифікатором;

– *приватні входи* (елементи вектора свідка – *witness vector*) – приватні значення (*private inputs*), а також допоміжні змінні схеми (проміжні значення/проміжні параметри), на підставі яких сторона, що доводить, згодом формує повний вектор свідка, що задовольняє схему (програму).

Опис схеми (R1CS, QAP), тобто формалізовані обмеження, зазвичай вже включені в *PK*.

Основні кроки (дії), які виконує сторона, що доводить:

1. Формування повного вектора свідка (*witness vector*): $s = (s_0, s_1, ..., s_n)^T$, де $s_0 = 1$ (константа), наступні елементи ($s_1, ..., s_n$) слідують у такій послідовності – спочатку публічні входи, потім секретні входи (*witness – private inputs*), потім проміжні значення, що обчислюються так, щоб задовольнити всі обмеження. Наприклад, для прикладу, що розглядається: $s = (s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7)^T = (1, x, y, z, r_1, r_2, r_3, r_4)^T = (1, 3, 4, 5, 9, 16, 8, 8)$.

2. Обчислення лінійних комбінацій зобов'язань (*commitments*) базисних поліномів $(g^{A(\tau)}, g^{B(\tau)}, g^{C(\tau)})$), використовуючи *PK* та вираз (22).

3. Рандомізація – вибір випадкових чисел r, σ для забезпечення нульового знання.

4. Побудова компонентів доказу. За допомогою *PK* та розрахованих значень будуються групи елементів. Наприклад, у Groth16 [51, 72, 73] це здійснюється за допомогою наступних виразів:

$$\pi = (\pi_A, \pi_B, \pi_C) = \begin{cases} \pi_A = g^{A(\tau)} \cdot (g^{\alpha})^r \in \mathrm{G}_1, \\ \pi_B = h^{B(\tau)} \cdot (h^{\beta})^{\sigma} \in \mathrm{G}_2, \\ \pi_C = g^{C(\tau)} \cdot (g^{\delta})^{r\sigma} \cdot (g^{A(\tau)})^{-\sigma} (g^{B(\tau)})^{-r} \in \mathrm{G}_1, \end{cases} \quad (24)$$

де зведення у ступінь $g^{A(\tau)}$ у π_A рандомізовано через параметр r: $(g^{\alpha})^r$; зведення у ступінь $h^{B(\tau)}$ у π_B рандомізовано через параметр σ: $(h^{\beta})^{\sigma}$; $(g^{A(\tau)})^{-\sigma}$ та $(g^{B(\tau)})^{-r}$ – реалізують відповідні перетворення (модифікацію/трансформацію, наприклад, шляхом використання зворотних ступенів та множників, щоб отримати форму, яка одночасно забезпечує безпеку та валідність, не змінюючи «повідомленого» математичного сенсу) для забезпечення властивостей zk-SNARK (збалансованості або коректності структурних перетворень) та рандомізації (щоразу навіть при однакових вхідних даних виведений доказ буде відрізнятися завдяки новим випадковим значенням); $(g^{\delta})^{r\sigma}$ – спеціальне маскування для параметра δ, що виключає можливість підробки. В цілому ж, наявність випадкових r, σ, а також $(g^{A(\tau)})^{-\sigma}$, $(g^{B(\tau)})^{-r}$ і $(g^{\delta})^{r\sigma}$, необхідні для забезпечення нульового знання та запобігання розкриттю інформації про *witness*.

Вираз (24) тісно пов'язаний з основною формулою коректності QAP – (19). Відомо, що вимога коректності полягає в тому, що для вектора свідка має бути виконане рівняння $A(\tau)\times B(\tau)-C(\tau)=H(\tau)\times Z(\tau)$. Це рівносильно тому, що $A(\tau)$, $B(\tau)$, $C(\tau)$ коректно побудовані, а ділимість виконується для деякого $H(x)$. π_A і π_B – це зобов'язання (*commitments*) до значень $A(\tau)$ і $B(\tau)$, рандомізовані за допомогою випадкових r і σ. Ці значення не розкривають сам вектор свідка (*witness vector*) і водночас приховують значення лінійних комбінацій. Вираз для π_C побудовано так, щоб реалізувати вкладену в QAP структуру, а саме, в показниках базових елементів «вбудована» вимога: «$A(\tau)\times B(\tau)-C(\tau)$ ділиться на $Z(\tau)$ без залишку», тобто π_C дозволяє перевірити, що існує таке $H(\tau)$, не обчислюючи його явно (*pairing*-перевірка з елементами *VK* пред'являє саме цю вимогу). Весь доказ $\pi=(\pi_A,\pi_B,\pi_C)$ організовано так, щоб публічно переконати в існуванні коректного $H(\tau)$ без розкриття секретів, тобто – у ділимості $A(\tau)\times B(\tau)-C(\tau)$ на $Z(\tau)$ для обраного τ.

Таким чином, наведені компоненти (вираз (24)) акуратно приховують знання свідка (*witness*) та коректність QAP усередині групових операцій (тобто вся необхідна інформація вбудовується та ховається через зобов'язання (*commitments*) та випадкові значення для нульового знання/розголошення).

Виходом цього етапу є доказ – зазвичай три елементи груп (точки еліптичної кривої) – $\pi=(\pi_A,\pi_B,\pi_C)$. Надалі сторона, що доводить, відправляє отриманий доказ разом з публічними входами стороні, що перевіряє (у прикладі, що розглядається, це *z*).

2.2.3 Алгоритм перевірки доказу.

Вхідною інформацією для сторони, що перевіряє, є:

– *Verifying Key (VK)* – попередньо згенерований на етапі *Trusted Setup* ключ перевірки (вираз (23)), що включає параметри та зобов'язання, потрібні для верифікації;

– *публічні входи* (*твердження*) $x=(x_0,...,x_k,...,x_N)$ – відомі значення (значення публічних змінних, наприклад, *z*), які використовуються як стороною, що доводить, так і верифікатором;

– *доказ* $\pi=(\pi_A,\pi_B,\pi_C)$, сформований стороною, що доводить.

Дії (кроки) сторони, що перевіряє:

1. Валідація вхідних даних:

приймаються публічні входи (твердження) та доказ π;

перевіряється коректність формату вхідних даних та докази (наприклад, чи належать елементи груп відповідним підгрупам).

2. Обчислення необхідних попередніх обчислень.

Якщо необхідно, обчислюються лінійні комбінації публічних входів з фіксованими зобов'язаннями (*commitments*) з *VK*, формуючи агрегований *commitment*, наприклад: $L = \sum_{k=0}^{N} x_k L_k$, що відповідає зобов'язанням до публічних входів, що обчислюються за поліномами Лагранжа. L відображає внесок усіх публічних входів одночасно. Замість перевірки кожного публічного входу окремо агрегування дозволяє перевірити всю сукупність входів одним груповим елементом, що дозволяє значно економити обчислювальні ресурси та час. У розглянутому прикладі з одним публічним входом і зробленими відповідними припущеннями L визначається наступним чином:

$$L = \sum_{k=0}^{1} x_k L_k = 1 \times L_0 + x_1 L_1 = 1 + z \times g^{l_4(\tau)} = 1 + z \times g^{l_4(\tau)} = 1 + 5 \times g^{l_4(7)} = 51 \bmod 17 = 0, \quad (25)$$

де $x_1 = z$; $L_1 = g^{l_4(\tau)}$; $l_4(x) = l_4(7) = \frac{(x-1)(x-2)(x-3)}{6} = \frac{120}{6} \bmod 17 = 3$; $g = 3$;

$g^{l_4(7)} = 3^{l_4(7)} \bmod 17 = 3^3 \bmod 17 = 27 \bmod 17 = 10$. Цей приклад є демонстраційним, тобто, це простий числовий лишок, а для криптографічної операції важливіша точка групи, а не число. У Groth16 кожен L_k – це елемент групи G_1 (точка на еліптичній кривій). Додавання з коефіцієнтами – це групова операція над точками, а не звичайне числове додавання по модулю. Однак отриманий результат демонстраційного прикладу заслуговує на увагу (важливий окремий випадок), так як за допомогою нього можна продемонструвати узагальнення використаного далі рівняння для перевірки (про що буде сказано нижче).

3. Перевірка парних відображень (*pairing*).

У схемі Groth16 основна перевірка зводиться до перевірки наступного рівняння:

$$e(\pi_A, \pi_B) \stackrel{?}{=} e(g^{\alpha}, h^{\beta}) \cdot e(\sum_{k=0}^{N} x_k L_k, h^{\gamma}) \cdot e(\pi_C, h^{\delta}), \quad (26)$$

яке можна перетворити на вигляд

$$e(\pi_A, \pi_B) \cdot e(\sum_{k=0}^{N} x_k L_k, h^{\gamma})^{-1} \cdot e(\pi_C, h^{\delta})^{-1} \stackrel{?}{=} e(g^{\alpha}, h^{\beta}) \quad (27)$$

або до більш короткого вигляду:

$$e(\pi_A, \pi_B) \cdot e(L, h^{\gamma})^{-1} \cdot e(\pi_C, h^{\delta})^{-1} \stackrel{?}{=} e(g^{\alpha}, h^{\beta}), \quad (28)$$

де знак $\stackrel{?}{=}$ – вказує, що це рівність, що перевіряється: якщо вона виконується, доказ приймається; $\pi_A \in \mathrm{G}_1$, $\pi_B \in \mathrm{G}_2$, $\pi_C \in \mathrm{G}_1$ – елементи доказу; $e: \mathrm{G}_1 \times \mathrm{G}_2 \to \mathrm{G}_T$ – білінійне парне відображення (*pairing*). У реальних криптосистемах часто використовують парне відображення типу Tate або Ate [74–78], наприклад на кривій Barreto-Naehrig, BN254.

Тобто верифікатор обчислює три значення парних відображень: $e(\pi_A, \pi_B)$, $e(L, h^{\gamma})^{-1}$, $e(\pi_C, h^{\delta})^{-1}$, потім перемножує їх і порівнює з відомим заздалегідь з *VK* – $e(g^{\alpha}, h^{\beta})$. Добуток зліва має збігатися з виразом праворуч. Це і є перевірка коректності доказу з прив'язкою до публічних входів. Це основна перевірка, що підтверджує, що доказ відповідає вимогам схеми (зокрема, ділимості відповідних поліномів).

Тепер звернемося до розглянутого прикладу (вираз (25)), в якому для демонстрації був отриманий числовий залишок, що дорівнює 0, що не відображає операцію додавання точок групи. Для коректного обчислення агрегованого зобов'язання необхідно використовувати

операції з точками на еліптичній кривій (додавання та множення на скаляр), а не просто арифметику за модулем для чисел. У зв'язку з цим, розглядаючи операції в термінах точок еліптичної кривої, отриманий результат слід інтерпретувати як підсумкову точку L, яка збіглася з нейтральним (нульовим – O) елементом групи G_1 (нульовою точкою, точкою нескінченності) – $L=\mathrm{O}$. Тоді, якщо $L=\mathrm{O}$, то елемент $e(L,h^{\gamma})^{-1}$ у виразі (28) можна записати так:

$$e(L,h^{\gamma})^{-1}=e(\mathrm{O},h^{\gamma})^{-1}=1_{\mathrm{G}_T}^{-1}=1_{\mathrm{G}_T}, \tag{29}$$

де 1_{G_T} – одиничний елемент (нейтральний елемент у мультиплікативних групах) цільової групи G_T.

Таким чином, множник парного відображення з L при $L=\mathrm{O}$ не робить внесок у перевірку і фактично може бути виключений. В результаті маємо наступну загальну формулу перевірки з урахуванням нейтрального елемента:

$$e(\pi_A,\pi_B)\cdot\left(\begin{cases}e(L,h^{\gamma})^{-1}, & L\neq\mathrm{O}\\ 1_{\mathrm{G}_T}, & L\neq\mathrm{O}\end{cases}\right)\cdot e(\pi_C,h^{\delta})^{-1}\overset{?}{=}e(g^{\alpha},h^{\beta}). \tag{30}$$

4. Результат перевірки:

якщо перевірки парних відображень відбуваються – повертається значення *true* (доказ валідний);

якщо перевірки парних відображень не відбуваються – повертається значення *false* (доказ відхиляється).

Отже,

– стороні, що перевіряє, не потрібні жодні секретні дані, вона використовує лише *VK*, публічні входи та доказ π;

– вся перевірка займає фіксований короткий час;

– якщо перевірка успішна – верифікатор переконується, що сторона, що доводить, дійсно знає секретні дані (*witness*), що задовольняють схему, без розкриття їх самих (самого *witness*).

Таким чином, у роботі викладено основні аспекти концепції нульового розголошення, включаючи аналіз застосовності інтерактивних та неінтерактивних підходів, оцінку існуючих ZKP-систем та концептуальне представлення технології zk-SNARK на базі популярної схеми Groth16 (що займає особливе місце завдяки мінімальному розміру доказів) з математичним обґрунтуванням. Здійснено систематизацію теоретичних основ та практичного застосування на простих та очевидних прикладах концепції нульового розголошення для кращого розуміння потенціалу ZKP у вирішенні завдань конфіденційності/приватності та верифікації даних.

Висновки

1. Наведені дослідження свідчать про значний прогрес у галузі криптографії, пов'язаної з нульовим розголошенням: від базових інтерактивних протоколів, що потребують постійного обміну даними, до високоефективних неінтерактивних систем. Перехід до неінтерактивності став каталізатором для впровадження ZK-технологій у реальні розподілені системи, де мінімізація комунікаційних витрат є критично важливою. Еволюція від інтерактивних схем (IZK) до неінтерактивних протоколів (NIZK) дозволила створити масштабовані та практично застосовні криптографічні інструменти, здатні забезпечувати безпечну та довірену верифікацію обчислень у розподілених системах.

2. Систематизовано теоретичні основи та практичні застосування концепції Zero Knowledge. Аналіз сучасних ZKP-систем показав їхню різноманітність, а також різноманіт-

ність використовуваних ними технологій, властивих їм переваг та обмежень, знання яких може надати суттєву допомогу фахівцям з кібербезпеки у виборі одного (можливо кількох) з них при реалізації відповідних рішень для забезпечення безпеки інформаційних систем сучасного ІТ-підприємства.

Протоколи з нульовим розголошенням є потужним інструментом у сучасній криптографії, що відкриває нові можливості для забезпечення приватності, безпеки та достовірності інформації у цифровому світі.

3. Особлива увага приділена концептуальному та математичному розбору технології zk-SNARK, зокрема, протоколу Groth16, що є одним з найефективніших рішень з точки зору компактності (мінімального розміру) доказів та швидкості їх перевірки. Схема Groth16 є яскравим прикладом успішної реалізації цих принципів, а її математичний фундамент у вигляді R1CS і QAP, що лежать в основі побудови доказів, дозволяє формалізувати обчислювальні завдання та перетворити їх на структуру, придатну для криптографічної перевірки. Розуміння цих математичних основ необхідне для коректного проєктування, аудиту та подальшого вдосконалення ZKP-систем в умовах зростаючих вимог до цифрової безпеки. І незважаючи на появу нових схем (PLONK, STARK), необхідність довіреної настройки, адаптації до квантових погроз, Groth16 все ще залишається еталоном за розміром доказу, що робить розуміння його математичного апарату доцільним для розробників та дослідників, які прагнуть поглибити свої знання у галузі прикладної криптографії.

3. Знання, викладені у статті, формують основу для подальшого вивчення криптографічних протоколів, розробки безпечних обчислювальних систем та застосування технологій ZKP у розподілених обчисленнях, блокчейнах, хмарних сервісах та інших галузях, де необхідний баланс між приватністю та перевіряльністю. Перспективи розвитку, перш за все, бачаться в переході до постквантових протоколів і універсальних SNARK, що відкриває нові горизонти для використання концепції нульового розголошення.

Список літератури:

1. Довгань О., Литвинова Л., Дорогих С. Кібербезпека в інформаційному суспільстві : Інформаційно-аналітичний дайджест. Державна наукова установа «Інститут інформації, безпеки і права Національної академії правових наук України», Національна бібліотека України імені В. І. Вернадського, Київ. 2025. Вип. 9. 176 с. URL: https://ippi.org.ua/sites/default/files/2025-09.pdf.
2. Kerman A., Borchert O., Rose S., Division E., Tan A. Implementing a zero trust architecture. National Institute of Standards and Technology. 2020. 17 p. URL: https://www.nccoe.nist.gov/sites/default/files/legacy-files/zta-project-description-final.pdf.
3. Buck C., Olenberger C., Schweizer A., Völter F., Eymann, T. Never trust, always verify: A multivocal literature review on current knowledge and research gaps of zero-trust. Computers & Security. 2021. 110. 102436.
4. National Cybersecurity Center of Excellence (NCCoE). Implementing a Zero Trust Architecture. URL: https://www.nccoe.nist.gov/projects/implementing-zero-trust-architecture.
5. Єсін В. І., Вілігура В. В., Узлов Д. Ю. Огляд існуючих моделей та основних принципів нульової довіри // Радіотехніка. 2024. Вип. 217. С. 39–54. https://doi.org/10.30837/rt.2024.2.217.03.
6. Єсін В. І., Вілігура В. В., Узлов Д. Ю. Архітектура нульової довіри: проблеми та рекомендації щодо успішного впровадження // Радіотехніка. 2024. Вип. 218. С. 7–34. https://doi.org/10.30837/rt.2024.3.218.01.
7. Goldwasser S., Micali S., Rackoff C. The knowledge complexity of interactive proof-systems // Proceedings of the seventeenth annual ACM symposium on Theory of computing (STOC '85). Association for Computing Machinery, New York, NY, USA, 1985. P. 291–304. https://doi.org/10.1145/22145.22178.
8. Goldwasser S., Micali, S., Rackoff C. The knowledge complexity of interactive proof systems. SIAM Journal on Computing. 1989. 18 (1). P. 186–208. https://doi.org/10.1137/0218012.
9. ZKProof Community Reference. Version 0.3. 2022.
URL: https://docs.zkproof.org/pages/reference/reference.pdf.
10. Schnorr C. P. Efficient signature generation by smart cards // Cryptology. 1991. 4. P. 161–174. https://doi.org/10.1007/BF00196725.
11. Schnorr C. P. Method for identifying subscribers and for generating and verifying electronic signatures in a data exchange system. U.S. Patent No. 4,995,082. 19 Feb. 1991.
12. Cui H., Zhang K., Chen Y., Liu Z., Yu Y. MPC-in-Multi-Heads: A Multi-Prover Zero-Knowledge Proof System // Bertino E., Shulman H., Waidner M. (eds) Computer Security – ESORICS 2021. ESORICS 2021. Lecture Notes in Computer Science, Springer, Cham. 2021. Vol. 12973. P. 332–351. https://doi.org/10.1007/978-3-030-88428-4_17.

13. Ozdemir A., Boneh D. Experimenting with Collaborative zk-SNARKs: Zero-Knowledge Proofs for Distributed Secrets // 31st USENIX Security Symposium (USENIX Security 22). 2022. P. 4291–4308. URL: https://www.usenix.org/system/files/sec22-ozdemir.pdf.

14. Kanjalkar S., Zhang Y., Gandlur S., Miller A. Publicly Auditable MPC-as-a-Service with succinct verification and universal setup // 2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Vienna, Austria. 2021. P. 386–411. https://doi.org/10.1109/EuroSPW54576.2021.00048.

15. Boneh D., Boyen X., Shacham H. Short Group Signatures // Franklin M. (eds) Advances in Cryptology – CRYPTO 2004. CRYPTO 2004. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg. 2004. Vol 3152. P. 41–55. https://doi.org/10.1007/978-3-540-28628-8_3.

16. Camenisch J., Lysyanskaya A. Signature Schemes and Anonymous Credentials from Bilinear Maps // Franklin M. (eds) Advances in Cryptology – CRYPTO 2004. CRYPTO 2004. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg. 2004. Vol. 3152. P. 56–72. https://doi.org/10.1007/978-3-540-28628-8_4.

17. Tessaro S., Zhu, C. Revisiting BBS Signatures // Hazay C., Stam M. (eds) Advances in Cryptology – EUROCRYPT 2023. EUROCRYPT 2023. Lecture Notes in Computer Science, Springer, Cham. 2023. Vol. 14008. P. 691–721. https://doi.org/10.1007/978-3-031-30589-4_24.

18. Ben-Sasson E., Bentov I., Horesh Y., Riabzev M. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive. 2018. 046. URL: https://eprint.iacr.org/2018/046.pdf.

19. Lavin R., Liu X., Mohanty H., Norman L., Zaarour G., Krishnamachari B. A survey on the applications of zero-knowledge proofs. arXiv preprint arXiv:2408.00243. 2024. URL: https://arxiv.org/pdf/2408.00243.

20. Goldreich O. Foundations of Cryptography. Basic Tools (Vol. 1). Cambridge University Press, 2001. 372 p.

21. Blum M., Feldman P., Micali S. Non-interactive zero-knowledge and its applications // Proceedings of the twentieth annual ACM symposium on Theory of computing (STOC '88). Association for Computing Machinery, New York, NY, USA, 1988. P. 103–112. https://doi.org/10.1145/62212.62222.

22. Fiege U., Fiat A., Shamir A. Zero knowledge proofs of identity. In Proceedings of the nineteenth annual ACM symposium on Theory of computing. 1987. P. 210–217. https://doi.org/10.1145/28395.28419.

23. Aad I. Zero-Knowledge Proof // Mulder, V., Mermoud, A., Lenders, V., Tellenbach, B. (eds) Trends in Data Protection and Encryption Technologies. Springer, Cham. 2023. P. 25–30. https://doi.org/10.1007/978-3-031-33386-6_6.

24. Boaz B. Zero knowledge proofs. URL: https://courses.csail.mit.edu/6.857/2018/files/L22-ZK-Boaz.pdf.

25. Parno B., Howell J., Gentry C., Raykova M. Pinocchio: nearly practical verifiable computation // Communications of the ACM. 2016. 59(2). P. 103–112. https://doi.org/10.1145/2856449.

26. Ben-Sasson E., Chiesa A., Tromer E., Virza, M. Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture // 23rd USENIX Security Symposium (USENIX Security 14). August 20–22, 2014. San Diego, CA. 2014. P. 781–796. URL: https://eprint.iacr.org/2013/879.pdf.

27. Wen H., Stephens J., Chen Y., Ferles K., Pailoor S., Charbonnet K., Feng Y. Practical Security Analysis of Zero-Knowledge Proof Circuits // 33rd USENIX Security Symposium (USENIX Security 24). 2024. P. 1471-1487.

28. Chen Z., Jiang Y., Song X., Chen L. A Survey on Zero-Knowledge Authentication for Internet of Things // Electronics. 2023. 12(5). 1145. https://doi.org/10.3390/electronics12051145.

29. Ethereum. What are zero-knowledge proofs? URL: https://ethereum.org/zero-knowledge-proofs/#what-are-zk-proofs.

30. Fiat A., Shamir A. How To Prove Yourself: Practical Solutions to Identification and Signature Problems // Odlyzko, A.M. (eds) Advances in Cryptology — CRYPTO' 86. CRYPTO 1986. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 1987. Vol 263. P. 186–194. https://doi.org/10.1007/3-540-47721-7_12.

31. Goldreich O., Oren Y. Definitions and properties of zero-knowledge proof systems // Cryptology. 1994. Vol. 7. P. 1–32. https://doi.org/10.1007/BF00195207.

32. Panther Academy. zk-SNARKs vs zk-STARKs: Comparing Zero-knowledge Proofs.
URL: https://blog.pantherprotocol.io/zk-snarks-vs-zk-starks-differences-in-zero-knowledge-technologies/

33. Gabizon A., Williamson Z. J., Ciobotaru O. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, 2019. 953.

34. Chiesa A., Hu Y., Maller M., Mishra P., Vesely N., Ward N. Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS // Canteaut A., Ishai Y. (eds) Advances in Cryptology – EUROCRYPT 2020. EUROCRYPT 2020. Lecture Notes in Computer Science. Springer, Cham. 2020. Vol. 12105. P. 738–768. https://doi.org/10.1007/978-3-030-45721-1_26.

35. Ward N., Chiesa A., Mishra P., Hu Y., Vesely N., Maller M. Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS // Technical Report No. UCB/EECS-2021-99. 2021. 102 p.

36. Kate A., Zaverucha G.M., Goldberg I. Constant-Size Commitments to Polynomials and Their Applications // Abe M. (eds) Advances in Cryptology – ASIACRYPT 2010. ASIACRYPT 2010. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2010. Vol. 6477. P. 177–194. https://doi.org/10.1007/978-3-642-17373-8_11.

37. Goldwasser S., Tauman Kalai Y. Cryptographic Assumptions: A Position Paper // Kushilevitz E., Malkin T. (eds) Theory of Cryptography. TCC 2016. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2016. Vol. 9562. P. 505–522. https://doi.org/10.1007/978-3-662-49096-9_21.

38. Mouris D., Tsoutsos N. G. Zilch: A Framework for Deploying Transparent Zero-Knowledge Proofs // IEEE Transactions on Information Forensics and Security. 2021. Vol. 16. P. 3269–3284. https://doi.org/10.1109/TIFS.2021.3074869.

39. Costello C., Fournet C., Howell J., Kohlweiss M., Kreuter B., Naehrig M., ... Zahur S. Geppetto: Versatile Verifiable Computation // 2015 IEEE Symposium on Security and Privacy, San Jose, CA, USA, 2015. P. 253–270. https://doi.org/10.1109/SP.2015.23.

40. Ben-Sasson E., Chiesa A., Genkin D., Tromer E., Virza M. SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge // Canetti R., Garay J.A. (eds) Advances in Cryptology – CRYPTO 2013. CRYPTO 2013. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2013. Vol 8043. P. 90–108. https://doi.org/10.1007/978-3-642-40084-1_6.

41. Wahby R. S., Setty S., Ren Z., Blumberg A. J., Walfish M. Efficient RAM and Control Flow in Verifiable Outsourced Computation // Proceedings 2015 Network and Distributed System Security Symposium. NDSS '15, 8–11 February 2015, San Diego, CA, USA. 2015. P. 1–15. https://doi.org/10.14722/ndss.2015.23097.

42. Eberhardt J., Tai S. Zokrates – scalable privacy-preserving off-chain computations // 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)., Halifax, NS, Canada. 2018. P. 1084–1091. https://10.1109/Cybermatics_2018.2018.00199.

43. Kosba A., Papamanthou C., Shi E. XJsnark: A Framework for Efficient Verifiable Computation // 2018 IEEE Symposium on Security and Privacy (SP). San Francisco, CA, USA, 2018. P. 944–961. https://10.1109/SP.2018.00018.

44. Kosba A., Papadopoulos D., Papamanthou C., Song D. MIRAGE: Succinct Arguments for Randomized Algorithms with Applications to Universal zk-SNARKs // 29th USENIX Security Symposium (USENIX Security 20). August 12–14, 2020. 2020. P. 2129–2146.

45. Maller M., Bowe S., Kohlweiss M., Meiklejohn S. Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings // Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19). Association for Computing Machinery, New York, NY, USA, 2019. P. 2111–2128. https://doi.org/10.1145/3319535.3339817.

46. Bünz B., Bootle J., Boneh D., Poelstra A., Wuille P., Maxwell G. Bulletproofs: Short Proofs for Confidential Transactions and More // 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2018. P. 315–334. https://doi.org/10.1109/SP.2018.00020.

47. Bünz B., Fisch B., Szepieniec A. Transparent SNARKs from DARK Compilers // Canteaut A., Ishai Y. (eds) Advances in Cryptology – EUROCRYPT 2020. EUROCRYPT 2020. Lecture Notes in Computer Science, Springer, Cham. 2020. Vol. 12105. P. 677–706. https://doi.org/10.1007/978-3-030-45721-1_24.

48. Wahby R. S., Tzialla I., Shelat A., Thaler J., Walfish, M. Doubly-Efficient zkSNARKs Without Trusted Set-up // 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2018. P. 926–943. https://doi.org/10.1109/SP.2018.00060.

49. Ames S., Hazay C., Ishai Y., Venkitasubramaniam M. Ligero: Lightweight Sublinear Arguments Without a Trusted Setup // Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17). Association for Computing Machinery, New York, NY, USA, 2017. P. 2087–2104. https://doi.org/10.1145/3133956.3134104.

50. Ben-Sasson E., Bentov I., Horesh Y., Riabzev, M. Scalable Zero Knowledge with No Trusted Setup // Boldyreva A., Micciancio D. (eds) Advances in Cryptology – CRYPTO 2019. CRYPTO 2019. Lecture Notes in Computer Science, Springer, Cham. 2019. Vol. 11694. P. 701–732. https://doi.org/10.1007/978-3-030-26954-8_23.

51. Groth J. On the Size of Pairing-Based Non-interactive Arguments // Fischlin M., Coron JS. (eds) Advances in Cryptology – EUROCRYPT 2016. EUROCRYPT 2016. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2016. Vol. 9666. P. 305–326. https://doi.org/10.1007/978-3-662-49896-5_11.

52. Katsumata S. A New Simple Technique to Bootstrap Various Lattice Zero-Knowledge Proofs to QROM Secure NIZKs // Malkin T., Peikert C. (eds) Advances in Cryptology – CRYPTO 2021. CRYPTO 2021. Lecture Notes in Computer Science,Springer, Cham. 2021. Vol 12826. P. 580–610. https://doi.org/10.1007/978-3-030-84245-1_20.

53. Lyubashevsky V., Nguyen N. K., Plançon M. Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General // Dodis Y., Shrimpton T. (eds) Advances in Cryptology – CRYPTO 2022. CRYPTO 2022. Lecture Notes in Computer Science, Springer, Cham. 2022. Vol. 13508. P. 71–101. https://doi.org/10.1007/978-3-031-15979-4_3.

54. Nguyen N. K., O'Rourke G. More Efficient Lattice-Based Zero-Knowledge Proofs with Straight-Line Extractability // Proceedings of the 12th ACM ASIA Public-Key Cryptography Workshop. 2025. P. 34–43. https://doi.org/10.1145/3709015.3728673.

55. Nikolaenko V., Ragsdale S., Bonneau J., Boneh D. Powers-of-Tau to the People: Decentralizing Setup Ceremonies // Pöpper C., Batina L. (eds) Applied Cryptography and Network Security. ACNS 2024. Lecture Notes in Computer Science, Springer, Cham. 2024. Vol. 14585. P. 105–134. https://doi.org/10.1007/978-3-031-54776-8_5.

56. Gailly N., Maller M., Nitulescu A. SnarkPack: Practical SNARK Aggregation // Eyal I., Garay J. (eds) Financial Cryptography and Data Security. FC 2022. Lecture Notes in Computer Science. Springer, Cham. 2022. Vol. 13411. P. 203–229. https://doi.org/10.1007/978-3-031-18283-9_10.

57. Evans D., Kolesnikov V., Rosulek M. A Pragmatic Introduction to Secure Multi-Party Computation // Foundations and Trends® in Privacy and Security. 2018. 2(2-3). P 70–246. http://dx.doi.org/10.1561/3300000019.
58. ZoKrates. URL: https://zokrates.github.io/toolbox/trusted_setup.html.
59. Zcash. URL: https://z.cash.
60. Kappos G., Yousaf H., Maller M., Meiklejohn S. An empirical analysis of anonymity in Zcash // 27th USENIX Security Symposium (USENIX Security 18). 2018. P. 463–477.
61. Hopwood D., Bowe S., Hornby T., Wilcox N. Zcash protocol specification. Version 2024.5.1-112-gcf7a5c [NU6]. 2024. 224 p. URL: https://caladex.io/api/files/papers/zcash%20white%20paper.pdf.
62. Pertsev A., Semenov R., Storm R. Tornado cash privacy solution version 1.4. 2019. URL: https://assets.super.so/c862512c-bc17-499b-9863-6144c0eb603c/files/e8d4f776-b0b1-422e-a492-f0861c002f64.pdf.
63. Nadler M., Schär F. Tornado cash and blockchain privacy: a primer for economists and policymakers. Federal Reserve Bank of St. Louis Review. 2023. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4352337.
64. Danezis G., Fournet C., Groth J., Kohlweiss M. Square Span Programs with Applications to Succinct NIZK Arguments // Sarkar P., Iwata T. (eds) Advances in Cryptology – ASIACRYPT 2014. ASIACRYPT 2014. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2014. Vol. 8873. P. 532–550. https://doi.org/10.1007/978-3-662-45611-8_28.
65. Sheybani N., Ahmed A., Kinsy M., Koushanfar F. Zero-Knowledge Proof Frameworks: A Systematic Survey. arXiv e-prints. 2025. arXiv: 2502.07063. https://doi.org/10.48550/arXiv.2502.07063.
66. Buterin V. Quadratic Arithmetic Programs: from Zero to Hero.
URL: https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649.
67. Gennaro R., Gentry C., Parno B., Raykova M. Quadratic Span Programs and Succinct NIZKs without PCPs // Johansson T., Nguyen P.Q. (eds) Advances in Cryptology – EUROCRYPT 2013. EUROCRYPT 2013. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2013. Vol. 7881. P. 626–645. https://doi.org/10.1007/978-3-642-38348-9_37.
68. ZoKrates. Proving schemes. https://zokrates.github.io/toolbox/proving_schemes.html.
69. Barreto P. S. L. M., Naehrig M. Pairing-friendly elliptic curves of prime order. International workshop on selected areas in cryptography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. P. 319–331.
70. Freeman D., Scott M. Teske E. A Taxonomy of Pairing-Friendly Elliptic Curves. 2010 // Cryptol. Vol. 23. P.224–280. https://doi.org/10.1007/s00145-009-9048-z.
71. Sakemi Y., Kobayashi T., Saito T., Wahby R. Pairing-Friendly Curves. 2021.
URL: https://www.ietf.org/archive/id/draft-irtf-cfrg-pairing-friendly-curves-10.html.
72. Groth16 Explained. URL: https://rareskills.io/post/groth16.
73. The RareSkills Book of Zero Knowledge. URL: https://rareskills.io/zk-book#brxe-mzqdsb.
74. Stange K.E. The Tate Pairing Via Elliptic Nets // Takagi T., Okamoto T., Okamoto E., Okamoto T. (eds) Pairing-Based Cryptography – Pairing 2007. Pairing 2007. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2007. Vol. 4575. P. 329–348. https://doi.org/10.1007/978-3-540-73489-5_19.
75. Hu L., Dong J. W., Pei D. Y. Implementation of cryptosystems based on Tate pairing // Journal of Computer Science and Technology. 2005. Vol. 20(2). P. 264–269.
76. Ogura N., Kanayama N., Uchiyama S., Okamoto E. Cryptographic Pairings Based on Elliptic Nets // Iwata T., Nishigaki M. (eds) Advances in Information and Computer Security. IWSEC 2011. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2011. Vol. 7038. P. 65–78. https://doi.org/10.1007/978-3-642-25141-2_5.
77. Granger R., Hess F., Oyono R., Theriault N., Vercauteren F. Ate Pairing on Hyperelliptic Curves // Naor M. (eds) Advances in Cryptology - EUROCRYPT 2007. EUROCRYPT 2007. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2007. Vol. 4515. P. 430–447. https://doi.org/10.1007/978-3-540-72540-4_25.
78. Nogami Y., Akane M., Sakemi Y., Kato H., Morikawa Y. Integer Variable χ–Based Ate Pairing // Galbraith S.D., Paterson K.G. (eds) Pairing-Based Cryptography – Pairing 2008. Pairing 2008. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2008. Vol 5209. P. 178–191. https://doi.org/10.1007/978-3-540-85538-5_13.

Надійшла до редколегії 11.01.2026
Прийнята до друку після рецензування 23.04.2026
Публікація (оприлюднення) 30.04.2026

Відомості про авторів:
Єсін Віталій Іванович – д-р техн. наук, професор, Харківський національний університет імені В.Н. Каразіна, професор кафедри кібербезпеки інформаційних систем, мереж і технологій, навчально-науковий інститут комп'ютерних наук та штучного інтелекту; Україна; e-mail: v.i.yesin@karazin.ua; ORCID: https://orcid.org/0000-0003-1977-7269
Вілігура Владислав Вікторович – PhD, Харківський національний університет імені В.Н. Каразіна, старший викладач кафедри кібербезпеки інформаційних систем, мереж і технологій, навчально-науковий інститут комп'ютерних наук та штучного інтелекту; Україна; e-mail: viligura93@gmail.com; ORCID: https://orcid.org/0000-0002-1137-2382