КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра математичної інформатики

Кваліфікаційна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформатика»

спеціальності 122 Комп'ютерні науки

на тему:

ЦИФРОВІ ПІДПИСИ ТА ЇХ ЗАСТОСУВАННЯ

Виконала студентка 4 курсу
Біляк Дарія Олександрівна _____
(підпис)

Науковий керівник:
Професор, доктор фіз.-мат. наук
Анісімов Анатолій Васильович _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри математичної інформатики
«____»_____2021 р.
протокол №_____

Завідувач кафедри

Терещенко В.М. _____
(підпис)

Київ – 2021

РЕФЕРАТ

Обсяг роботи 65 сторінок, 7 ілюстрацій, 25 джерел.

ЦИФРОВІ ПІДПИСИ, КРИПТОВАЛЮТИ, КІЛЬЦЕВІ ЦИФРОВІ ПІДПИСИ, ЕЛІПТИЧНІ КРИВІ, PYTHON.

Об’єктом аналізу є алгоритми цифрового підпису, їх будова, властивості та варіанти застосування. Об’єктом розробки є однин з розглянутих алгоритмів, а саме алгоритм зв’язної схеми кільцевого цифрового підпису з використанням еліптичних кривих.

Метою дипломної роботи є проведення дослідження характеристик існуючих алгоритмів цифрового підпису, аналіз їх застосування та розробка одного з них, а саме алгоритму зв’язної схеми кільцевого цифрового підпису з використанням еліптичних кривих LSAG.

Методи розроблення: аналіз існуючих алгоритмів, проектування майбутньої архітектури програми, покрокова розробка з розбиттям на підзадачі.

Інструменти розроблення: Для розробки програмного рішення було обрано мову програмування Python. У ролі інтегрованого середовища розробки програмного забезпечення використовувалось середовище PyCharm.

Результат роботи: проведено глибокий аналіз схем цифрового підпису, розглянуто найбільш розповсюджені та використовувані алгоритми цифрового підпису, а також їх застосування, в тому числі у криптовалютах. Розроблено програмну реалізацію алгоритму зв’язної схеми кільцевого цифрового підпису з використанням еліптичних кривих, модифікації якого досить часто застосовуються у криптовалютах.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта розробки. Розвиток комунікацій та інструментів, що дозволяють обмін даними на відстані, призвів до потреби у збережені електронних даних від модифікації, підробки та пошкодження. У цьому зацікавлені органи державної влади різних країн світу, комерційні та некомерційні організації, звичайні люди та громадяни в особистих та офіційних цілях. Цифрові підписи – досить нова криптографічна структура, що не так давно знайшла своє застосування у повсякденному житті, однак вони активно розвиваються, вдосконалюються та активно використовуються у криптовалютах, банківській сфері та у системах голосування.

Актуальність роботи та підстави для її виконання. В наш час людське життя стає все більш цифровим і без електронних даних та їх передачі через різноманітні канали зв’язку його важко уявити. Особливо з урахуванням дистанціювання у зв’язку з пандемією, на відстані відбувається майже все: від навчання у школі та університеті, до роботи та покупки продуктів харчування. Разом з тим зростає потреба у збереженні персональної інформації, даних про перекази коштів на банківських рахунках та іншого. Тому цифрові підписи, як інструменти верифікації даних та, у деяких випадках, як спосіб зберегти інформацію у секреті, є як ніколи актуальним у наш час.

Мета й завдання роботи. Метою дипломної роботи є дослідження та аналіз існуючих алгоритмів цифрового підпису та зв’язного алгоритму кільцевого підпису LSAG, який лежить в основі конфіденційних транзакцій деяких криптовалют. Для досягнення цієї мети поставлено такі завдання:

а) провести аналіз будови та безпеки цифрових підписів, механізмів, що лежать в їх основі;

б) провести аналіз існуючих алгоритмів цифрового підпису, їх особливостей та сфер застосування;

в) розробити програмну реалізацію алгоритму зв’язного цифрового підпису LSAG з можливістю подальшого використання та інтегрування.

Об'єкт, методи й засоби розроблення. Об’єктом аналізу є цифрові підписи, алгоритми цифрового підпису та огляд варіантів їх застосування. Об’єктом розробки є реалізація програмного рішення алгоритму зв’язної схеми кільцевого цифрового підпису з використанням еліптичних кривих. Вибір алгоритму був зроблений з врахуванням аналізу алгоритмів цифрового підпису та їх застосування з попередніх розділів.

Перед проектуванням та розробкою даного програмного рішення був проведений аналіз та створення власної моделі доступу до ресурсів відповідно до потреб даного продукту. Були використані напрацювання, присутні у розглянутих рішеннях, та нові власні ідеї, що спростили розробку та удосконалення програмного рішення.

Для розробки програмної реалізації було обрано мову Python. Мова програмування Python є скриптовою та підтримує процедурне, об’єктно-орієнтоване програмування. Код, написаний на Python, не є дуже швидким, однак він досить простий для розуміння, використання, написання та вдосконалення. Мова має багату стандартну бібліотеку, яка включає в себе поширені структури даних та алгоритми.

Під час розробки даного програмного засобу використовувалося інтегроване програмне середовище PyCharm. Воно підтримує всі можливості мови Python та системи контролю версій git. Для студентів вищих навчальних закладів є безкоштовна версія.

Можливі сфери застосування. Розроблене програмне рішення може бути використане як для включення у існуючу систему криптовалюти, голосування тощо, так і для побудови нових систем.

1 ЦИФРОВІ ПІДПИСИ

1.1 Загальний огляд

Цифровий підпис – це математична схема, що застосовується для перевірки автентичності та цілісності цифрових повідомлень, документів. Автор підпису не може відмовитись від факту підписання ним повідомлення чи документу, тому цифровий підпис є функціональним аналогом звичайного рукописного підпису та навіть розширює його можливості, гарантуючи більшу безпеку. [1]

Зазвичай цифрові підписи використовуються стороною, що підписує повідомлення або документ – підписантом, та певним набором сторін, що потенційно можуть перевіряти, чи справді підпис зробив конкретний підписант до конкретного повідомлення.

Непідробна схема цифрового підпису повинна відповідати таким вимогам :

- кожен користувач може ефективно підписати документ чи повідомлення за власним вибором;
- кожен користувач може перевірити, чи є даний рядок підписом конкретного користувача на конкретному документі;
- неможливо надати підписи користувачів до документів, яких вони не підписували.

Схеми цифрового підпису, в переважній більшості, базуються на використанні асиметричної криптографії – криптосистем, які використовують пари ключів: публічні та приватні. Публічні або відкриті ключі зазвичай дозволяється розголошувати без втрат для безпеки, а приватні повинні бути секретними, конфіденційними.

Зазвичай схема цифрового підпису складається з трьох алгоритмічних компонент:

а) алгоритм генерації приватного ключа, який з однаковою ймовірністю обирає один випадковий ключ з допустимої множини приватних ключів, результатом роботи алгоритму є приватний ключ та відповідний йому публічний ключ;

б) алгоритм, що відповідає безпосередньо за створення підпису, він отримує повідомлення та приватний ключ, а натомість створює підпис;

в) алгоритм перевірки підпису, який за даним повідомлення, публічним ключем та підписом встановлює, чи відповідають вони одне одному і чи є повідомлення автентичним.

Використання асиметричної криптографії дозволяє реалізувати одну з основних вимог до цифрового підпису – перевірку автентичності без можливості підписати інший документ, оскільки для цих цілей використовуються різні ключі. Для підписання повідомлення або документу використовується приватний ключ, який є конфіденційним, безпечність створення цифрового підпису полягає у тому, що без знання приватного ключа неможливо створити дійсний, тобто такий, що пройде перевірку з публічним ключем, підпис.

Цифрові підписи можуть використовуватись для підписання повідомлень, документів, онлайн-транзакцій тощо в сфері зв'язку, банківській сфері, при голосуваннях, в галузі охорони здоров'я і т.д.

Цифрові підписи використовуються з трьох основних причин. По-перше, вони дозволяють автентифікувати ідентичність повідомлення, тобто встановити, що підпис був породжений діями конкретного користувача. По-друге, цифрові підписи гарантують цілісність повідомлення, тобто виключається ймовірність того, що повідомлення було змінене під час передачі. По-третє, підписи мають властивість неспростовності походження, тобто якщо користувач підписав щось, пізніше він не зможе заперечити свій підпис.

1.2 Формальне визначення схеми цифрового підпису

Схема підпису – це трійка (G, S, V) ймовірнісних алгоритмів, що працюють за поліноміальний час і задовольняють наступні дві вимоги [3]:

а) На вхід 1^n, алгоритм генерації ключів G виводить пару бітових рядків – пару ключів: приватного s та публічного v відповідно, будемо записувати їх як (s, v).

б) Для кожної пари *(s, v)* у діапазоні $G(1^n)$ та для кожного $\alpha \in \{0,1\}^*$ алгоритм підпису S та верифікації V задовольняють умові $Pr\left[V\left(v, \alpha, S(s, \alpha)\right) = 1\right] = 1$, де ймовірність обчислюється з внутрішніх ймовірностей алгоритмів S та V.

Ціле додатне число n служить параметром безпеки схеми цифрового підпису. Алгоритм генерації ключів G отримує число n як вхідний параметр, від нього залежить довжина вихідних даних – пари рядків. Кожна пара (s, v) у діапазоні $G(1^n)$ являє собою пару відповідних одне одному ключів для підписання та верифікації, тобто пару з приватного та публічного ключа криптосистеми.

$S(s, \alpha)$ називають підписом документа або повідомлення α, зробленим за допомогою приватного ключа підписання s. Аналогічно, коли $V(v, \alpha, \beta) = 1$, кажуть, що β – дійсний підпис для α стосовно відкритого ключа верифікації v.

Пара з приватного та публічного ключів (s, v), породжена алгоритмом генерації ключів G, разом із трійкою алгоритмів (G, S, V) утворює екземпляр схеми цифрового підпису (G, S, V).

Будемо говорити, підписант підписує повідомлення, хоча насправді це може бути документ, рядок тощо.

Схема цифрового підпису використовується так. Нехай Аліса хоче підписати повідомлення. Вона запускає алгоритм генерації приватного та публічного ключа $G(1^n)$ та отримує пару (s, v). З вимог до схеми цифрового підпису маємо, що будь-яка інша сторона може отримати легітимну копію публічного ключа v. Тому Аліса може оприлюднити свій публічний ключ v як такий, що належить їй. Коли Аліса захоче підписати повідомлення α, вона використовує алгоритм підпису S, обчислює підпис $S(s, \alpha)$ за допомогою свого приватного ключа s. Тепер Аліса може надіслати пару з повідомлення α та підпису $S(s, \alpha)$. Він, знаючи відкритий публічний ключ v, може встановити автентичність повідомлення α, застосувавши алгоритм верифікації V та перевіривши, чи $V\left(v, \alpha, S(s, \alpha)\right) = 1$. Тепер Боб може бути впевненим у тому, що

дійсно Аліса надіслала йому це повідомлення та в тому, що повідомлення не було змінене в процесі передачі.

1.3 Безпека схеми цифрового підпису

Питання безпеки схем цифрового підпису пов'язане з захистом, який не дозволить жодному зловмиснику мати ефективну змогу сфальсифікувати дійсну пару з повідомлення α та підпису до нього S стосовно відкритого ключа v, який був створений чесним користувачем. Для того, щоб схема цифрового підпису вважалась безпечною, достатньо звести до якогось досить малого значення ймовірність того, що зловмиснику вдасться створити дійсну підробку цифрового підпису, з огляду на те, що час роботи зловмисника та ймовірність підробки підпису залежать від параметру безпеки схеми n.

Параметр безпеки $n \in \mathbb{N}$ використовується для параметризації як зловмисника, так і власне схеми цифрового підпису. Певною мірою n можна розглядати як показник рівня безпеки даного екземпляру схеми підпису.

Ефективного зловмисника будемо порівнювати з ймовірнісними алгоритмами, що працюють за поліноміальний час, де час вимірюється як функція від параметру безпеки n. Оскільки зловмисник обмежений у роботі поліноміальним часом, необхідно, щоб усі алгоритми, які використовуються чесною стороною, теж працювали за поліноміальний час.

Можна сказати, що подія відбувається з досить малою ймовірністю, якщо ймовірність того, що подія настане, є незначною при даному значення параметру n. Формально це можна означити так. Функція $\varepsilon: \mathbb{N} \to [0,1]$ є незначною для всіх $c \geq 0$ таких, що існує $n_c \geq 0$ таке, що $\varepsilon(n) < \frac{1}{n_c}$ для всіх $n > n_c$.

Оскільки зловмисник обмежений у роботі поліноміальним часом, необхідно, щоб усі алгоритми, які використовуються чесною стороною, як-от алгоритм генерації ключів G, підписання повідомлення S та верифікації підпису V, теж працювали за поліноміальний час.

Атака – це процес, в ході якого зловмисник може отримати підписи до документів стосовного деякої пари ключів (s, v) створеної алгоритмом генерування ключів G [2]. Можливі два випадки:

– випадок з приватним ключем s: зловмисник отримує на вхід 1^n, підписи генеруються стосовно приватного ключа s, тобто $(s, v) \leftarrow G(1^n)$;
– випадок з публічним ключем v: зловмисник отримує v на вхід, підписи генеруються стосовно приватного ключа s, тобто $(s, v) \leftarrow G(1^n)$.

Зловмисник успішно здійснює атаку, якщо в результаті певних дій він генерує дійсну пару з повідомлення α та підпису до нього $\beta = S(s, \alpha)$: $V(v, \alpha, \beta) = 1$, таку, що повідомлення α не належить до множини повідомлень A, які раніше підписував чесний користувач: $\alpha \notin A, A = \{\alpha_1, \dots\}$. Будемо називати схеми, у яких можливі такі атаки, неможливими для існування. Схема цифрового підпису сильно неможлива для існування, якщо зловмисник може згенерувати дійсну пару з повідомлення α та підпису $\beta = S(s, \alpha)$ стосовно пари з приватного та публічного ключа (s, v), при чому відмінну від усіх таких пар, створених чесним користувачем раніше: $(\alpha, \beta) \notin \{(\alpha_1, \beta_1), \dots\}$.

1.3.1 Безпека при атаці з вибором повідомлення

Модель атаки з вибором повідомлення є найсильнішою з можливих, оскільки надає зловмиснику повний контроль над тим, які повідомлення підписуються. Зловмисник може не тільки обрати повідомлення після того, як побачить публічний ключ, а й відібрати повідомлення на основі раніше обраних та підписаних повідомлень. Тому іноді такий вид атаки називають адаптивним.

Формально, атака повідомлення з вибором ототожнюється з ймовірнісним алгоритмом з оракулом – машиною з оракулом, що працює за поліноміальний час. Оракул зловмисника – певна «чорна скринька», він отримує доступ до процесу підписання документу, який виконує алгоритм підписання S, разом з секретним

приватним ключем s. Оракул обчислює підписи для фіксованого приватного ключа s, пов'язаного з публічним ключем v парою (s, v). Якщо атака здійснюється з використанням публічного ключа, зловмисник також отримує значення v.

Нехай маємо ймовірнісну машину з оракулом M. Позначимо як $Q_M^O(x)$ набір запитів, що виконуються машиною M на вхід x з доступом до оракула O. Тоді як $M^O(x)$ позначимо результат обчислень, що здійснила машина M з оракулом O отримавши на вхід x. Фактично $Q_M^O(x)$ та $M^O(x)$ є двома аспектами одного й того ж самого ймовірнісного обчислення, тому їх можна назвати двома залежними випадковими змінними.

Можливі два варіанти адаптивної атаки: з використанням механізму приватного ключа чи з використанням публічного ключа. У першому випадку зловмисник даватиме оракулу на вхід 1^n, а в другому, відповідно, публічний ключ v. Позначимо вхідні дані зловмисника як w і матимемо на увазі, що залежно від виду атаки w може набувати лише двох значень: 1^n та v.

Схема цифрового підпису (G, S, V) вважається непідробною для атаки з вибором повідомлення, якщо за наступних умов ймовірність успішної атаки є незначущою як функція від параметру безпеки n:

а) алгоритм генерації ключів G, отримуючи на вхід 1^n, створює пару з публічного та приватного ключів (s, v);

б) отримавши на вхід w, оракул M створює підпис β до повідомлення α стосовно алгоритму підпису повідомлення S та приватного ключа s;

в) повідомлення α не належить до множини повідомлень, що уже надходили до оракула з входом w стосовно алгоритму підпису повідомлення S та приватного ключа s;

г) алгоритм верифікації V підтверджує, що β – дійсний підпис для α стосовно відкритого ключа верифікації v.

У випадку атаки приватного ключа, w трактується як 1^n, якщо атака здійснюється з використанням публічного ключа, w – це v.

Більш стисло та формально атаку з вибором повідомлення або адаптивну атаку можна означити так. [4]

У випадку атаки з приватним ключем, схема цифрового підпису є безпечною стосовно приватного ключа, якщо для будь-якої ймовірнісної машини з оракулом M, що працює за поліноміальний час, для кожного додатного полінома p та для всіх достатньо великих n маємо, що

$$Pr\left[\begin{matrix} V(v,\alpha,\beta)=1 \ \& \ \alpha \notin Q_M^{S,s}(1^n) \\ \text{де } (s,v)\leftarrow G(1^n) \text{ та } (\alpha,\beta)\leftarrow M^{S,s}(1^n) \end{matrix}\right] < \frac{1}{p(n)}.$$

У випадку атаки з публічним ключем, схема цифрового підпису є безпечною стосовно приватного ключа, якщо для будь-якої ймовірнісної машини з оракулом M, що працює за поліноміальний час, для кожного додатного полінома p та для всіх достатньо великих n маємо, що

$$Pr\left[\begin{matrix} V(v,\alpha,\beta)=1 \ \& \ \alpha \notin Q_M^{S,s}(v) \\ \text{де } (s,v)\leftarrow G(1^n) \text{ та } (\alpha,\beta)\leftarrow M^{S,s}(v) \end{matrix}\right] < \frac{1}{p(n)}.$$

В усіх зазначених випадках, ймовірність, яка має місце у формулах, залежить від внутрішніх ймовірностей алгоритмів генерації ключів G, підписання S та верифікації V, а також від ймовірностей всередині машини з оракулом M.

Схема атаки може виглядати таким чином. Алгоритм генерації ключів G, отримавши на вхід 1^n, створює пару з приватного та публічного ключа (s, v), вони будуть залишатись незмінними принаймні до кінці атаки. Далі зловмисник намагається викликати на вхід 1^n у випадку атаки приватного ключа та v – у випадку публічного. В обох випадках зловмисник отримує доступ оракула до алгоритму підпису S стосовно приватного ключа підписанта s. Оракул може бути як детермінованим, так і ймовірнісним, тобто таким, що на однакові вхідні дані буде давати різну відповідь. Тоді зловмисник отримує пару рядків (α, β). Атака вважається успішною тоді і тільки тоді, коли:

а) рядок α відрізняється від усіх запитів на підписи, зроблених зловмисником, тому перший рядок у вихідній парі $(\alpha, \beta) = M^{S,s}(x)$ відрізняється від будь-якого іншого рядка в $Q_M^{S,s}(x)$, де $x = 1^n$ або $x = v$, залежно від того, яка це атака: з приватним ключем чи з публічним;

б) пара (α, β) відповідає дійній парі перевірки документу з підписом стосовно публічного ключа верифікації v, тобто $V(v, \alpha, \beta) = 1$.

Будь-яка схема цифрового підпису, яка є безпечною в моделі публічного ключа, також є безпечною в моделі приватного ключа, однак зворотне твердження не завжди істинне.

1.4 Цифрові підписи з обмеженнями та загальні цифрові підписи

Схеми цифрового підпису з певними обмеженнями відіграють важливу роль в розумінні структури та властивостей схем цифрового підпису з більш широкими можливостями. Окрім загальних схем цифрових підписів, екземпляри яких можна використовувати довільну кількість разів та застосовувати їх до повідомлень довільної довжини, варто розглянути два типи обмежень: схеми цифрового підпису з обмеженням довжини та одноразові схеми цифрового підпису, а також варіант поєднання цих обмежень між собою.

1.4.1 Схеми цифрового підпису з обмеженням довжини

Обмеження довжини у схемах цифрового підпису стосується довжини повідомлень, які підписуються. При чому питання про те, що буде результатом підпису, якщо підписати повідомлення неналежної довжини, не є критичним. Важливіше те, що результатом успішної фальсифікації підпису зловмисником може вважатись лише підпис до повідомлення фіксованої довжини.

Нехай $l: \mathbb{N} \to \mathbb{N}$, l-обмежена схема цифрового підпису це трійка (G, S, V) ймовірнісних алгоритмів що працюють за поліноміальний час і задовольняють наступні дві вимоги:

а) На вхід 1^n, алгоритм генерації ключів G виводить пару бітових рядків – пару ключів: приватного s та публічного v відповідно, будемо записувати їх як (s, v).

б) Для кожної пари (s, v) у діапазоні $G(1^n)$ та для кожного $\alpha \in \{0,1\}^{l(n)}$ алгоритм підпису S та верифікації V задовольняють умові

$Pr\left[V\left(v,\alpha,S(s,\alpha)\right)=1\right]=1$, де ймовірність обчислюється з внутрішніх ймовірностей алгоритмів S та V.

Така схема цифрового підпису називається захищеною в моделі приватного ($w=1^n$) або публічного ключа ($w=v$), якщо у випадку атаки з приватним або публічним ключем для будь-якої ймовірнісної машини з оракулом M, що працює за поліноміальний час, для кожного додатного полінома p та для всіх достатньо великих n маємо, що

$$Pr\left[\begin{matrix}V(v,\alpha,\beta)=1\ \&\ \alpha\notin Q_M^{S,s}(1^n)\ \&\ \alpha=l(n)\\ \text{де } (s,v)\leftarrow G(1^n) \text{ та } (\alpha,\beta)\leftarrow M^{S,s}(w)\end{matrix}\right]<\frac{1}{p(n)}.$$

Як і раніше, ймовірність, яка має місце у формулі, залежить від внутрішніх ймовірностей алгоритмів G, S та V, а також від ймовірностей всередині машини з оракулом M. Обмеження довжини у даному випадку суттєво впливає на визначення безпеки та відповідні вимоги до неї. [2]

Існує два методи переходу від l-обмежених схем цифрового підпису до загальних. Перший з них полягає у синтаксичному аналізі та розділенні оригінального повідомлення на блоки та застосуванні l-обмеженої схеми цифрового підпису до кожного з них. Суть другого методу у гешуванні повідомлення чи документу у значення довжиною $l(n)$ бітів з використанням відповідної схеми гешування та застосуванні l-обмеженої схеми цифрового підпису до отриманого гешу. У другого методу є суттєві переваги: схема цифрового підпису обмеженої довжини викликається лише один раз, а довжина підписів, згенерованих нею, має фіксовану довжину та залежить тільки від довжини ключів. Однак цей метод вимагає додаткового припущення про існування геш-функції без колізій.

1.4.1.1 Підписання блоків

Розглянемо l-обмежену схему цифрового підпису (G,S,V). З неї потрібно побудувати загальну схему цифрового підпису (G',S',V'), в якій алгоритм генерації ключа нової схеми G' такий же, як і в l-обмеженій, тобто $G'=G$, а алгоритми підписання повідомлення S' та верифікації підпису V' модифіковані та

дозволяють працювати з повідомленнями довільної довжини. Вважатимемо повідомлення, яке ми збираємось підписувати, послідовністю з блоків розміру $l(n) = l'(n)/O(1)$. Тоді повідомлення α розкладається на p блоків, тобто $\alpha = \alpha_1, \dots, \alpha_p$, при чому кожен з блоків α_i має довжину $l'(n)$.

Найпростіший варіант підписання повідомлення α блоками полягає у простому, без будь-яких інших кроків, підписанні усіх блоків α_i окремо. Тоді для послідовності блоків $\alpha_1, \dots, \alpha_p$ отримаємо відповідну послідовність підписів $\beta_1, \dots, \beta_p$, де кожне β_i є підписом до блоку повідомлення α_i. Однак такий спосіб підписання блоками є небезпечним. Нехай повідомлення α складається з двох блоків, тобто $\alpha = \alpha_1, \alpha_2$, при чому $\alpha_1 \neq \alpha_2$. Підпис повідомлення α теж складатиметься з двох блоків, тобто $\beta = \beta_1, \beta_2$. Тоді підпис $\beta' = \beta_2, \beta_1$ буде вважатись дійсним підписом повідомлення $\alpha' = \alpha_2, \alpha_1$, де $\alpha_1, \alpha_2 \neq \alpha_2, \alpha_1$.

Варіантом вирішення такої проблеми є підписання не одного блоку α_i, а підписання пари (i, α_i) натомість. Однак, якщо чесна сторона підпише повідомлення $\alpha = \alpha_1, \alpha_2, \alpha_3$ та отримає підпис $\beta = \beta_1, \beta_2, \beta_3$, зловмисник все ще зможе використовувати $\alpha = \alpha_1, \alpha_2$ та $\beta = \beta_1, \beta_2$ як дійсну пару повідомлення-підпис.

Тоді в кожен рядок довжиною $l(n)$ біт варто включити також загальну кількість p блоків α_i в повідомленні α. Нехай чесна сторона створює підпис $\beta = \beta_1, \beta_2$ до повідомлення $\alpha = \alpha_1, \alpha_2$ та підпис $\sigma = \sigma_1, \sigma_2$ до повідомлення $\lambda = \lambda_1, \lambda_2$, де $\alpha_1, \alpha_2 \neq \lambda_1, \lambda_2$. Тоді зловмисник може використовувати пару (β_1, σ_2) як дійсний підпис до повідомлення (α_1, λ_2).

Таким чином, для того, щоб зробити схему підписання блоками безпечною і уникнути можливості генерування нових підписів з блоків повідомлень та підписів, що були створені раніше, необхідно додати ще один параметр до набору даних, з яким підписується блок. Найочевидніший варіант такого параметру – ідентифікатор повідомлення.

Нехай (G, S, V) – l-обмежена схема цифрового підпису. З неї можна побудувати загальну схему цифрового підпису (G', S', V'), в якій $G = G'$, якщо

кожне повідомлення α представити як послідовність блоків $\alpha_1, \dots, \alpha_p$, кожен з яких має довжину $l'(n) = l(n)/4$.

Алгоритм підписання S' отримує на вхід приватний ключ s, який належить $G(1^n)$, та документ довільної довжини $\alpha \in \{0,1\}^*$. Алгоритм S' розділяє вхідне повідомлення α на послідовність з p блоків $\alpha_1, \dots, \alpha_p$ так, що будь-яке повідомлення α однозначно відновлюється з набору блоків α_i і кожен блок α_i являє собою рядок довжиною $l'(n)$ біт. Далі алгоритм S' рівномірно обирає $r \in \{0,1\}^{l(n)}$. Для кожного $i = 1, \dots, p$ алгоритм S' обчислює підпис β_i для блоку α_i: $\beta_i \leftarrow S(s, r, p, i, \alpha_i)$, де i та p представляють собою рядки довжиною $l'(n)$ біт. Як уже було сказано раніше, β_i фактично є підписом до структури, яку можна інтерпретувати як «α_i є i-тим блоком повідомлення α, що складається з p блоків та пов'язується з ідентифікаційним параметром r». Як результат, алгоритм підписання блоків S' генерує загальний підпис до повідомлення α як набір з ідентифікаційного параметру r, кількості блоків в повідомленні p та послідовності підписів β_i до блоків α_i: $(r, p, \beta_1, \dots, \beta_p)$.

Алгоритм верифікації повідомлення V' отримує на вхід публічний ключ v, який належить $G(1^n)$ та пов'язаний з приватним ключем s парою приватний-публічний ключ (s, v), повний підпис $\beta = (r, p, \beta_1, \dots, \beta_2)$ повідомлення α, згенерований алгоритмом S' та власне документ α. Алгоритм верифікації V' розділяє повідомлення α на послідовність з p' блоків $\alpha_1, \dots, \alpha_{p'}$ використовуючий той самий алгоритм, що і алгоритм підписання блоків S'. Алгоритм V' визначає підпис β до повідомлення α як дійсний тоді і тільки тоді, коли:

а) $p' = p$, тобто кількість блоків p, яку отримав алгоритм підписання повідомлення S' збігається з кількістю блоків p', яку отримав алгоритм верифікації V' при умові, що вони використовують один алгоритм розбиття повідомлення на блоки;

б) для кожного $i = 1, \dots, p$ алгоритм верифікації V' підтверджує, що підпис β_i є дійсним підписом блоку α_i повідомлення α, яке

складається з p блоків і пов'язане з ідентифікатором r, тобто $V(v,(r,p,i,\alpha_i),\beta_i)=1$.

Якщо l-обмежена схема цифрового підпису (G,S,V) є захищеною від адаптивної атаки в моделі приватного та публічного ключів, то побудована загальна повноцінна схема цифрового підпису (G',S',V') також захищена від адаптивної атаки в моделі приватного та публічного ключів. [1]

1.4.1.2 Підписання геш-значення

Розглянемо l-обмежену схему цифрового підпису (G,S,V). Узагальнення цієї схеми для підписання повідомлень будь-якої довжини полягає у гешуванні повідомлення у $l(n)$-бітне значення та підписанні цього значення замість вхідного повідомлення. Таким чином, окрім l-обмеженої схеми цифрового підпису, використовується також алгоритм гешування. Існує два основних способи реалізації: один з них базується на так званому «гешуванні без колізій», інший використовує «універсальне гешування в одну сторону». Для розуміння принципів роботи схем цифрового підпису, які використовують підписання геш-значення, розглянемо гешування без колізій.

Стійка до колізій схема гешування складається з набору функцій $\{h_s:\{0,1\}^*\to\{0,1\}^{|s|}\}_{s\in\{0,1\}^*}$ таких, що для заданих s та x легко обрахувати $h_s(x)$, але для випадкового s важко знайти $x\neq x'$ таке, що $h_s(x)=h_s(x')$. [2]

Нехай $l:\mathbb{N}\to\mathbb{N}$. Набір функцій $\{h_s:\{0,1\}^*\to\{0,1\}^{l(|s|)}\}_{s\in\{0,1\}^*}$ називається вільним від колізій, якщо існує ймовірнісний алгоритм I, що працює за поліноміальний час такий, що:

а) для деякого поліному p, для всіх досить великих n та кожного s в діапазоні $I(1^n)$ виконується нерівність $n\leq p(|s|)$. Окрім цього, n може бути обчислене з s за поліноміальний час;

б) існує алгоритм, що працює за поліноміальний час, який, отримавши на вхід s та x, повертає $h_s(x)$;

в) пара (x, x') утворює колізію, пов'язану з функцією h, якщо $h_s(x) = h_s(x')$, але $x \neq x'$; кожен ймовірнісний алгоритм, що працює за поліноміальний час, отримавши $I(1^n)$ як вхід, на виході дає колізію, пов'язану з $h_{I(1^n)}$ з незначною ймовірністю, тобто для кожного ймовірнісного алгоритму A, що працює за поліноміальний час, кожного додатного поліному n та достатньо великих s маємо, що $\Pr[A(I(1^n)) \text{ спричиняє колізію, пов'язану з } h_{I(1^n)}] < \frac{1}{p(n)}$, де ймовірність залежить від внутрішніх ймовірностей алгоритмів A та I.

Нехай *(G, S, V)* – l-обмежена схема цифрового підпису та набір функцій $\{h_p: \{0,1\}^* \to \{0,1\}^{l(|p|)}\}_{p \in \{0,1\}^*}$ вільний від колізій. Побудуємо загальну схему цифрового підпису (G', S', V') таким чином:

Алгоритм генерації ключів G', отримавши на вхід 1^n, викликає алгоритм генерації ключів G, щоб отримати пару з публічного та приватного ключів $(s, v) \leftarrow G(1^n)$. Він викликає I, індексуючий алгоритм з набору вільних від колізій функцій, щоб отримати $p \leftarrow I(1^n)$. На вихід алгоритм G' видає дві пари (p, s) та (p, v), де (p, s) використовується як приватний ключ, а (p, v) як публічний ключ.

Алгоритм підписання повідомлення S', отримавши на вхід приватний ключ для підписання (p, s) в діапазоні $G_1'(1^n)$ та повідомлення $\alpha \in \{0,1\}^*$, викликає алгоритм підписання S та отримує на виході підпис $\beta = S\left(s, h_p(\alpha)\right)$.

Алгоритм верифікації підпису V', отримавши на вхід публічний ключ (p, v) в діапазоні $G_2'(1^n)$, повідомлення $\alpha \in \{0,1\}^*$, а також підпис β, викликає алгоритм верифікації підпису V та на виході отримує значення $V\left(v, h_p(\alpha), \beta\right)$.

Означена схема цифрового підпису (G', S', V') викликає l-обмежену схему цифрового підпису (G, S, V) всього один раз, при цьому довжина генерованих нею підписів залежить виключно від довжин приватного та публічного ключів, на неї не впливає довжина вхідного повідомлення, тобто $|S'(p, s, \alpha)| = \left|S\left(s, h_p(\alpha)\right)\right|$, а це в свою чергу обмежується поліномом $\left(|s|, l(|r|)\right)$.

Якщо l-обмежена схема цифрового підпису (G, S, V) є захищеною від адаптивної атаки в моделі приватного та публічного ключів і якщо $\{h_p: \{0,1\}^* \to \{0,1\}^{l(|p|)}\}_{p\in\{0,1\}^*}$ – набір геш-функцій, вільний від колізій, то побудована загальна повноцінна схема цифрового підпису (G', S', V') також захищена від адаптивної атаки в моделі приватного та публічного ключів. [1] Описана загальна схема цифрового підпису, яка використовує гешування, являє собою парадигму hash-and-sign, тобто геш-та-підпис. Така схема часто використовується на практиці, коли довгий документ гешується в короткий рядок, до якого застосовується базова обмежена схема цифрового підпису.

1.4.2 Одноразові схеми цифрового підпису

Одноразова схема цифрового підпису – це звичайна загальна схема цифрового підпису, для якої є суттєво менше вимог, пов'язаних з безпекою, адже вважається, що зловмисник може отримати не більше, ніж одну пару з підписаного повідомлення та публічного ключа. [4]

Атака з вибором одного повідомлення або адаптивна атака в такому випадку, це атака, для якої зловмисник може отримати підпис щонайбільше до одного повідомлення за вибором. Тобто зловмисник отримує публічний ключ v як вхідні дані, та отримує підпис, породжений приватним ключем s, при чому приватний та публічні ключі в цьому випадку належать одній парі, згенерованій алгоритмом генерації ключів G: $(s, v) \leftarrow G(1^n)$.

Як і раніше, атака вважається успішною при підробці, якщо вона видає дійсний підпис до рядка, до якого раніше не було дійсного підпису і для якого підпис не вимагався власником приватного ключа в той час, коли зловмисник проводив атаку. Одноразова схема цифрового підпису вважається безпечною або непідробною, якщо кожна атака з вибором одного повідомлення успішна з незначною ймовірністю.

Формально, атака повідомлення з вибором одного повідомлення ототожнюється з ймовірнісним алгоритмом з оракулом – машиною з оракулом,

що працює за поліноміальний час. Оскільки мова йде лише про атаки з одним повідомленням, розглядаються лише оракули, які роблять щонайбільше один запит. Нехай M є також машиною з оракулом. Як і раніше, $Q_M^O(x)$ – набір запитів, що виконуються машиною M на вхід x з доступом до оракула O, при чому $\left|Q_M^O(x)\right| \leq 1$, а $M^O(x)$ – результат обчислень, що здійснила машина M з оракулом O отримавши на вхід x.

Одноразова схема цифрового підпису безпечна якщо для будь-якої ймовірнісної машини з оракулом M, що працює за поліноміальний час та робить щонайбільше один запит, якщо для кожного додатного полінома p та для всіх достатньо великих n маємо, що $Pr\left[\begin{matrix}V(v,\alpha,\beta)=1 \,\&\, \alpha \notin Q_M^{S,s}(v) \,\&\, \left|Q_M^{S,s}(x)\right| \leq 1 \\ \text{де } (s,v) \leftarrow G(1^n) \text{ та } (\alpha,\beta) \leftarrow M^{S,s}(v)\end{matrix}\right] < \frac{1}{p(n)}$. [2]

Ймовірність залежить від внутрішніх ймовірностей алгоритмів G, S та V, а також від ймовірностей всередині машини з оракулом M.

Далі розглянемо поєднання одноразових схем цифрового підпису з обмеженими по довжині вхідного повідомлення.

Нехай $l: \mathbb{N} \to \mathbb{N}$, l-обмежена одноразова схема цифрового підпису це трійка *(G, S, V)* ймовірнісних алгоритмів що працюють за поліноміальний час і задовольняють наступні дві вимоги:

а) На вхід 1^n, алгоритм генерації ключів G виводить пару бітових рядків – пару ключів: приватного s та публічного v відповідно, будемо записувати їх як *(s, v)*.

б) Для кожної пари *(s, v)* у діапазоні $G(1^n)$ та для кожного $\alpha \in \{0,1\}^{l(n)}$ алгоритм підпису S та верифікації V задовольняють умові $Pr\left[V\left(v, \alpha, S(s,\alpha)\right) = 1\right] = 1$, де ймовірність обчислюється з внутрішніх ймовірностей алгоритмів S та V.

Така схема цифрового підпису є безпечною в моделі атаки одного повідомлення, якщо для будь-якої ймовірнісної машини з оракулом M, що працює за поліноміальний час, для кожного додатного полінома p та для всіх достатньо великих n маємо, що

$$Pr\left[\begin{matrix}V(v,\alpha,\beta)=1\ \&\ \alpha\notin Q_M^{S,s}(1^n)\ \&\ \alpha=l(n)\ \&\ \left|Q_M^{S,s}(x)\right|\leq 1\\ \text{де } (s,v)\leftarrow G(1^n) \text{ та } (\alpha,\beta)\leftarrow M^{S,s}(v)\end{matrix}\right] < \frac{1}{p(n)}. \text{ [2]}$$

Побудуємо l-обмежену одноразову схему цифрового підпису.

Нехай $l\colon \mathbb{N}\to\mathbb{N}$ обмежене поліномом та таке, що обчислюється за поліноміальний час, і $f\colon\{0,1\}^*\to\{0,1\}^*$ обчислюване за поліноміальний час та зберігає довжину. Тоді l-обмежена одноразова схема цифрового підпису – це трійка (G,S,V), в якій алгоритми генерації ключів G, підписання повідомлення S та верифікації підпису V визначаються так.

Алгоритм генерації ключів G на вхід 1^n рівномірно обирає $s_1^0, s_{1,\dots}^1, s_{l(n)}^o, s_{l(n)}^1\in\{0,1\}^n$ та обчислює $v_i^j=f\left(s_i^j\right)$ для кожного $i=1,\dots,l(n)$ та $j=0,1$. Тоді $s=\left((s_1^0,s_1^1),\dots,\left(s_{l(n)}^0,s_{l(n)}^1\right)\right)$ та $v=\left((v_1^0,v_1^1),\dots,\left(v_{l(n)}^0,v_{l(n)}^1\right)\right)$, на виході алгоритм G генерує пару ключів (s,v), при чому $|s|=|v|=2\cdot l(n)\cdot n$.

Алгоритм підписання S повідомлення отримує на вхід ключ для підписання $s=\left((s_1^0,s_1^1),\dots,\left(s_{l(n)}^0,s_{l(n)}^1\right)\right)$ та $l(n)$-бітне повідомлення $\alpha=\alpha_1,\dots,\alpha_{l(n)}$, а на виході генерує підпис $\beta=\left(S_1^{\alpha_1},\dots,S_{l(n)}^{\alpha_{l(n)}}\right)$.

Алгоритм верифікації повідомлення V отримує на вхід ключ для верифікації $v=\left((v_1^0,v_1^1),\dots,\left(v_{l(n)}^0,v_{l(n)}^1\right)\right)$, $l(n)$-бітне повідомлення $\alpha=\alpha_1,\dots,\alpha_{l(n)}$ та підпис $\beta=\left(\beta_1,\dots,B_{l(n)}\right)$. Алгоритм V підтверджує, що підпис β дійсно є підписом повідомлення α стосовно ключа для верифікації v тоді і лише тоді, коли $v_i^{\alpha_i}=f(\beta_i)$ для кожного $i=1,\dots,l(n)$.

Якщо f одностороння геш-функція, то описана схема підпису (G,S,V) є безпечною l-обмеженою одноразовою схемою цифрового підпису. [1] Якщо до неї застосувати означену раніше парадигму hash-and-sign можна легко отримати загальну одноразову схему цифрового підпису.

Якщо (G,S,V) – безпечна l-обмежена одноразова схема цифрового підпису та набір функцій $\left\{h_p\colon\{0,1\}^*\to\{0,1\}^{l(|p|)}\right\}_{p\in\{0,1\}^*}$ вільний від колізій, то схема цифрового підпису, пов’язана з парадигмою hash-and-sign (G',S',V') є безпечною

одноразовою схемою цифрового підпису, при чому довжина підписів, які генеруються нею, залежить лише від довжини ключів. [1]

1.4.3 Загальні схеми цифрового підпису

Для побудови загальної багаторазової схеми цифрового підпису з загальної одноразової необхідно розглянути кілька важливих конструкційних рішень, які дозволять схемі бути безпечною. Такими рішеннями є парадигма оновлення та автентифікаційне дерево. [2]

1.4.3.1 Парадигма оновлення

Парадигма оновлення застосовується для зменшення ймовірності успішності атаки з вибором повідомлення. Її суть полягає у тому, що для підписання кожного нового повідомлення використовується новий екземпляр схеми цифрового підпису, автентифікований якимось одним, основним екземпляром. Схему цифрового підпису, яка використовує парадигму оновлення, можна означити так.

Нехай (G, S, V) – схема цифрового підпису, а (G', S', V') – одноразова схема цифрового підпису. Визначимо схему цифрового підпису (G'', S'', V''). В ній алгоритм генерації ключів G'' збігається з визначеним раніше: $G'' = G$, а алгоритми підписання S'' та верифікації V'' побудовані так.

Алгоритм підписання S'', отримавши на вхід приватний ключ s в діапазоні $G_1''(1^n)$ та повідомлення $\alpha \in \{0,1\}^*$, спочатку викликає алгоритм генерації ключів одноразової схеми G' та отримує нову пару ключів $(s', v') \leftarrow G'(1^n)$. Потім він викликає алгоритм підписання S, щоб отримати підпис $\beta_1 \leftarrow S(s, v')$ та S', щоб отримати підпис $\beta_2 \leftarrow S'(s', \alpha)$. На виході алгоритм отримує структуру з підпису до створеної пари ключів, створений публічний ключ та підпис створеним екземпляром схеми до повідомлення: (β_1, v', β_2).

Алгоритм верифікації V'', отримавши на вхід публічний ключ v в діапазоні $G_2''(1^n)$, повідомлення $\alpha \in \{0,1\}^*$ та підпис (β_1, v', β_2). Алгоритм V'' визначає підпис дійсним тоді і тільки тоді, коли $V(v, v', \beta_1) = 1$ та $V'(v, \alpha, \beta_2) = 1$.

Якщо (G, S, V) – безпечна схема цифрового підпису, а (G', S', V') – безпечна одноразова схема цифрового підпису, то (G'', S'', V'') – також безпечна схема.

Парадигма оновлення пропонує кожного разу використовувати новий екземпляр схеми підпису для підписання документу, і цей екземпляр повинен бути автентифікованим стосовно загальнодоступного ключа верифікації загального екземпляру. Тобто пряма реалізація такої схеми вимагає підписання багатьох екземплярів підпису єдиним ключем, тому одноразова схема підпису не може бути використана для таких цілей.

1.4.3.2 Автентифікаційне дерево

Ідея дерева автентифікацій полягає у побудові дерева з екземплярів одноразової схеми цифрового підпису, у якій кожен батьківський вузол автентифікує свої дочірні вузли. Для підписання повідомлень використовуватимуться листки такого дерева. Кожен з вузлів так підписуватиме не більше одного рядка (листові вузли підписуватимуть безпосередньо повідомлення, а внутрішні вузли – рядки, що складаються з кількох ключів верифікації дочірніх елементів). Схему цифрового підпису, яка використовує автентифікаційне дерево, можна описати так. [2]

Нехай (G, S, V) – одноразова схема цифрового підпису. Визначимо схему цифрового підпису, що базується на використанні автентифікаційного дерева, як (G', S', V'), де $G' = G$. З параметром безпеки n, схема використовує повне бінарне дерево глибини n. Кожен з вузлів цього дерева маркується як бінарний рядок, тоді корінь дерева матиме мітку порожнього рядка, позначимо її як λ. У будь-якого вузла x ліва дитина позначається $x0$, а права – як $x1$. Будемо позначати пару ключів (s, v), асоційовану з вузлом x, як (s_x, v_x). Поточний стан дерева називатимемо записом.

Для ініціювання схеми з певним параметром безпеки n викликається алгоритм генерації ключів G для отримання пари ключів (s, v): $(s, v) \leftarrow G(1^n)$. Ця пара ключів асоціюється з коренем дерева, тобто $(s_\lambda, v_\lambda) = (s, v)$.

Для підписання алгоритмом S' повідомлення α спочатку знаходимо листок, який ще не був не використаний для підписання. Позначимо мітки листка як $\sigma_1 \dots \sigma_n$. Далі для кожного $i = 1, \dots, n$ та для кожного $\tau \in \{0,1\}$ ми намагаємось отримати пару ключів, пов'язану з вузлом, у якого мітка $\sigma_1 \dots \sigma_{i-1}\tau$. Якщо така пара не знайдена, вона генерується через виклик алгоритму генерації ключів $G(1^n)$: $\left(s_{\sigma_1 \dots \sigma_{i-1}\tau}, v_{\sigma_1 \dots \sigma_{i-1}\tau}\right) \leftarrow G(1^n)$ та записується для зберігання та подальшого використання у запис. Далі для кожного $i = 1, \dots, n$ ми намагаємось дістати з запису підпис до рядка $v_{\sigma_1 \dots \sigma_{i-1}0} v_{\sigma_1 \dots \sigma_{i-1}1}$ пов'язний з ключем підписання $s_{\sigma_1 \dots \sigma_{i-1}}$. Якщо такий підпис не знайдено, він генерується через виклик алгоритму підписання S з приватним ключем $s_{\sigma_1 \dots \sigma_{i-1}}$ та зберігається для подальшого використання. Отримуємо підпис $S\left(s_{\sigma_1 \dots \sigma_{i-1}}, v_{\sigma_1 \dots \sigma_{i-1}0} v_{\sigma_1 \dots \sigma_{i-1}1}\right)$. Позначимо $auth_{\sigma_1 \dots \sigma_{i-1}} \stackrel{\text{def}}{=} \left(v_{\sigma_1 \dots \sigma_{i-1}0}, v_{\sigma_1 \dots \sigma_{i-1}1}, S\left(s_{\sigma_1 \dots \sigma_{i-1}}, v_{\sigma_1 \dots \sigma_{i-1}0} v_{\sigma_1 \dots \sigma_{i-1}1}\right)\right)$. Нарешті, після підписання документу α алгоритмом підписання $S_{\sigma_1 \dots \sigma_n}$ отримуємо структуру $\left(\sigma_1 \dots \sigma_n, auth_\lambda, auth_{\sigma_1}, \dots, auth_{\sigma_1 \dots \sigma_{n-1}}, S_{\sigma_1 \dots \sigma_n}(\alpha)\right)$.

Алгоритм верифікації V' отримує на вхід публічний ключ v, повідомлення α та підпис β. Алгоритм V' верифікує підпис β як дійсний підпис повідомлення α стосовно публічного ключа v тоді і тільки тоді, коли:

а) структура β має такий вигляд (σ_i – біти, інші символи – це рядки):
$\left(\sigma_1 \dots \sigma_n, \left(v_{0,0}, v_{0,1}, \beta_0\right), \left(v_{1,0}, v_{1,1}, \beta_1\right), \dots, \left(v_{n-1,0}, v_{n-1,1}, \beta_{n-1}\right), \beta_n\right)$;

б) $V\left(v, v_{0,0}\, v_{0,1}, \beta_0\right) = 1$;

в) для кожного $i = 1, \dots, n-1$ маємо, що $V\left(v_{i-1,\sigma_i}, v_{i,0}\, v_{i,1}, \beta_i\right) = 1$;

г) $V\left(v_{n-1,\sigma_n}, \alpha, \beta\right) = 1$.

Отримана схема цифрового підпису (G', S', V'), що побудована на дереві автентифікацій, називається залежною від пам'яті. Якщо одноразова схема

цифрового підпису (G, S, V) є безпечною, то і побудована на її основі схема цифрового підпису (G', S', V') також є безпечною. [1]

1.4.3.3 Загальна схема цифрового підпису

Конструкція загальних схем цифрового підпису базується на схемах цифрового підпису, які використовують автентифікаційне дерево. Суттєва відмінність між ними полягає у тому, що загальні схеми не потребують додаткової пам'яті. У них замість збережених даних до певних рядків застосовуються псевдовипадкові функції. Замість зберігання кожного вузла дерева, можна застосовувати псевдовипадкову функцію для кожного виклику алгоритму генерації ключів з використанням мітки вузла, таким чином, якщо колись ще виникне потреба у парі ключів з даного вузла, можна буде згенерувати їх знову. [2]

Припустимо, що для параметру безпеки n алгортм генерації ключів як для приватного, так і для публічного ключів використовує рівно n випадкових значень. Для $r \in \{0,1\}^n$ позначимо як $G(1^n, r)$ вихід алгоритму генерації ключів G на вхід 1^n та внутрішнє випадкове r. Аналогічно, для $r \in \{0,1\}^n$, $S(s, \alpha, r)$- вихід алгоритму підписання S, який отримав на вхід приватний ключ s та повідомлення α разом з внутрішнім випадковим r.

Нехай $r: \mathbb{N} \to \mathbb{N}$. Вважаємо, що $\{f_s: \{0,1\}^* \to \{0,1\}^{r(|s|)}\}_{s \in \{0,1\}^*}$ – псевдовипадкова функція з необмеженим входом, якщо існує алгоритм, який з поліноміальний час на вхід s та $x \in \{0,1\}^*$ повертає $f_s(x)$; для будь-якої ймовірнісної машини з оракулом M, що працює за поліноміальний час, для кожнного поліноміального p та достатньо великих n маємо, що $|Pr[M^{F_n}(1^n) = 1] - Pr[M^{H_n}(1^n) = 1]| < \frac{1}{p(n)}$, де F_n – випадкова величина, рівномірно розподілена на мультимножині $\{f_s\}_{s \in \{0,1\}^n}$, а H_n рівномірно розподілене між всіма функціями, що відображають рядки довільної довжини в $r(n)$-бітні рядки.

Нехай (G, S, V) – одноразова схема цифрового підпису та $\{f_r: \{0,1\}^* \to \{0,1\}^{|r|}\}_{r\in\{0,1\}^*}$- узагальнена псевдовипадкова функція. Побудуємо схему (G', S', V'), що використовує в своєму дизайні повне бінарне дерево, так.

Алгоритм генерації ключів G'на вхід 1^n обчислює пару ключів $(s, v) \leftarrow G(1^n)$ та обирає рівномірно $r \in \{0,1\}^n$. На виході алгоритм дає пару$\big((r, s), v\big)$, де (r, s) – приватний ключ підписання, v – публічний ключ.

Алгоритм підписання S' отримує на вхід приватний ключ (r, s) в діапазоні $G_1'(1^n)$ та повідомлення $\alpha \in \{0,1\}^*$, та підписує його за наступним алгоритмом. Спочатку він рівномірно обирає $\sigma_1 \dots \sigma_n \in \{0,1\}^n$. Далі для кожного $i = 1, \dots, n$ та для кожного $\tau \in \{0,1\}$ алгоритм S' викликає алгоритм генерації ключів G: $\big(s_{\sigma_1\dots\sigma_{i-1}\tau}, v_{\sigma_1\dots\sigma_{i-1}\tau}\big) \leftarrow G\big(1^n, f_r(10\sigma_1 \dots \sigma_{i-1}\tau)\big)$. Для кожного $i = 1, \dots, n$ алгоритм викликає S з ключем $\sigma_1 \dots \sigma_{i-1}$ та отримує

$$auth_{\sigma_1\dots\sigma_{i-1}} \stackrel{\text{def}}{=} \Big(v_{\sigma_1\dots\sigma_{i-1}0}, v_{\sigma_1\dots\sigma_{i-1}1}, S\big(s_{\sigma_1\dots\sigma_{i-1}}, v_{\sigma_1\dots\sigma_{i-1}0} v_{\sigma_1\dots\sigma_{i-1}1}, f_r(11\sigma_1 \dots \sigma_{i-1})\big)\Big).$$

Нарешті, алгоритм викликає S з ключем $\sigma_1 \dots \sigma_n$ та отримує структуру $\Big(\sigma_1 \dots \sigma_n, auth_\lambda, auth_{\sigma_1}, \dots, auth_{\sigma_1\dots\sigma_{n-1}}, S_{\sigma_1\dots\sigma_n}\big(\alpha, f_r(11\sigma_1 \dots \sigma_n)\big)\Big)$.

Алгоритм верифікації повідомлення V' отримує на вхід публічний ключ v, повідомлення α та підпис β. Алгоритм V' верифікує підпис β як дійсний підпис повідомлення α стосовно публічного ключа v тоді і тільки тоді, коли:

а) структура β має такий вигляд (σ_i – біти, інші символи – це рядки):
$\big(\sigma_1 \dots \sigma_n, (v_{0,0}, v_{0,1}, \beta_0), (v_{1,0}, v_{1,1}, \beta_1), \dots, (v_{n-1,0}, v_{n-1,1}, \beta_{n-1}), \beta_n\big)$;

б) $V\big(v, v_{0,0}\ v_{0,1}, \beta_0\big) = 1$;

в) для кожного $i = 1, \dots, n-1$ маємо, що $V\big(v_{i-1,\sigma_i}, v_{i,0}\ v_{i,1}, \beta_i\big) = 1$;

г) $V\big(v_{n-1,\sigma_n}, \alpha, \beta\big) = 1$.

Якщо одноразова схема цифрового підпису (G, S, V) є безпечною та $\{f_r: \{0,1\}^* \to \{0,1\}^{|r|}\}_{r\in\{0,1\}^*}$ – набір узагальнених псевдовипадкових функцій, схема цифрового підпису (G', S', V') також є безпечною. [1]

1.5 Кільцеві підписи

Кільцеві цифрові підписи вперше були представлені Рівестом, Шаміром і Тауманом в 2001 році. [5] Вони дозволяють користувачеві підписувати повідомлення таким чином, щоб можна було ідентифікувати коло потенційних підписантів, одним з яких є сам підписант, однак без можливості визначити, хто сам з підписантів підписав повідомлення. Кільцеві підписи не потребують централізації та координації між різними учасниками кола підпису, користувачі схеми можуть навіть не знати одне одного. Крім того, користувачі мають повний контроль над рівнем анонімності кожного конкретного підпису.

1.5.1 Визначення

Назвемо набір потенційних підписантів колом. Користувача, який підписує повідомлення, називатимемо підписантом, а інших користувачів – не-підписантами. Кожен можливий підписант пов'язаний з публічним ключем v_k, який визначає його схему підпису та, відповідно, є його ключем верифікації. Відповідний йому секретний ключ позначимо як s_k. Загальне поняття схеми кільцевого підпису не вимагає особливих властивостей індивідуальних схем підпису.

Схема кільцевого підпису – це трійка ймовірнісних алгоритмів, що працюють за поліноміальний час (RG, RS, RV). Вони визначені так.

Нехай U – коло з r учасників.

Алгоритм генерації ключів RG на вхід 1^n обчислює пару ключів $(s_k, v_k) \leftarrow G(1^n)$ для k-го користувача з кола.

Алгоритм підписання повідомлення RS отримує на вхід повідомлення α, набір з публічних ключів учасників кола U $v_1, \dots, v_l$, згенерованих алгоритмом генерації ключів RG, а також приватний ключ k-го члену кола s_k. На виході алгоритм обчислює підпис $\sigma = RS(s_k, v_1, \dots, v_l, \alpha)$.

Алгоритм верифікації підпису RV отримує на вхід набір публічних ключів учасників кола на вхід повідомлення α, набір з публічних ключів учасників кола U $v_1, \dots, v_l$, згенерованих алгоритмом генерації ключів RG, а також підпис σ,

створений алгоритмом RS. Він підтверджує, що підпис σ є дійсним підписом повідомлення α стосовно публічних ключів $v_1, \dots, v_l$ тоді і тільки тоді, коли $RV(v_1, \dots, v_l, \alpha, RS(s_k, v_1, \dots, v_l, \alpha)) = 1$. Це означає, що підпис σ породжений з використанням приватного ключа одного з учасників кола U.

Для підписання повідомлення підписанту не потрібне знання, згода чи допомога інших учасників кола, усе, що необхідно – знання їх публічних ключів.

1.5.2 Специфічні кільцеві схеми цифрового підпису

Схеми кільцевих підписів, залежно від сфери їх застосування, володіють різними властивостями, що суттєво відрізняють їх між собою та забезпечують той чи інший функціонал. До специфічних схем кільцевого підпису належать заперечувані, порогові, перевірочні, загальні, зв'язні, підзвітні, короткі, розділені та інші схеми кільцевого підпису, а також ідентифікаційні схеми, що базуються на них. [6] [7] Далі наведені визначення деяких з них.

Заперечувана схема кільцевого підпису дозволяє члену кільця переконати верифікатора в тому, що повідомлення автентифікується одним із членів кільця, не розкриваючи, ким саме, а верифікатор не зможе переконати третю сторону в тому, що повідомлення справді автентифіковано.

t-порогова схема кільцевого підпису – це схема кільцевого підпису, у якій кожен підпис у кільці є доказом того, що принаймні t учасників кільця підтверджують повідомлення.

Перевірочна схема кільцевого підпису пропонує додаткову властивість: якщо підписувач, що належить до кільця, хоче довести, що саме він створив підпис, то верифікатор зможе визначити, чи дійсно це так.

Зв'язна схема кільцевого підпису – це схема кільцевого підпису, яка дозволяє встановити, чи були два підписи створені одним і тим же учасником кільця.

Розділена схема кільцевого підпису – це схема кільцевого підпису, в якій учасники можуть створювати ключі та обирати домені параметри схеми незалежно одне від одного.

2 АЛГОРИТМИ ЦИФРОВИХ ПІДПИСІВ

2.1 Схеми цифрового підпису, що базуються на використанні односторонніх функцій

2.1.1 Одноразова схема Лампорта

Підпис Лампорта або одноразова схема цифрового підпису Лампорта – це схема цифрового підпису, в основі якої лежить використання односторонньої функції, зазвичай криптографічної геш-функції. Вважається, що підписи Лампорта, за умови використання досить потужних геш-функцій, будуть безпечними навіть у квантову епоху. Кожну пару ключів схеми Лампорта можна використовувати лише один раз, але з використанням автентифікаційного дерева схема може бути досить ефективною та дозволить безпечно створювати кілька підписів з використанням однієї пари ключів.

Криптосистема Лампорта була винайдена в 1979 році та названа в честь Леслі Лампорта [8].

2.1.1.1 Формальне визначення

Функція $f: \{0,1\}^* \to \{0,1\}^*$ називається односторонньою, якщо: існує детермінований алгоритм, що працює за поліноміальний час, $Eval_f$ такий, що для всіх k та для всіх $x \in \{0,1\}^k$ маємо, що $Eval_f(x) = f(x)$; ймовірність $Pr[x \leftarrow \{0,1\}^k; y \coloneqq f(x); x' \leftarrow A(1^k, y): f(x') = y]$ незначна для всіх ймовірнісних алгоритмів A, що працюють за поліноміальний час.

Нехай f – одностороння функція, а $l = l(n)$, як функція від параметру безпеки n, позначає бажану довжину повідомлення.

Схема цифрового підпису Лампорта – це трійка алгоритмів (G, S, V). Вони визначені наступним чином.

Алгоритм генерації ключів G для усіх $i \in \{1, \dots, l\}$ обирає випадкове $x_{i,0}, x_{i,1} \leftarrow \{0,1\}^n$ та обчислює $y_{i,0} \coloneqq f(x_{i,0})$ та $y_{i,1} \coloneqq f(x_{i,1})$. Тоді приватний

ключ s та публічний ключ v виглядатимуть, відповідно, так: $s := \begin{pmatrix} x_{1,0} & \dots & x_{l,0} \\ x_{1,1} & \dots & x_{l,1} \end{pmatrix}$ та $v := \begin{pmatrix} y_{1,0} & \dots & y_{l,0} \\ y_{1,1} & \dots & y_{l,1} \end{pmatrix}$.

Алгоритм підписання повідомлення S отримує на вхід приватний ключ s та повідомлення $\alpha \in \{0,1\}^l$, при чому $\alpha = \alpha_1 \dots \alpha_l$. На виході алгоритм S генерує підпис $S(s, \alpha) = S(s, \alpha_1 \dots \alpha_l) = \left(x_{1,\alpha_1}, \dots, x_{l,\alpha_l}\right) = (s_1, \dots, s_l) = S$.

Алгоритм верифікації підпису V отримує на вхід публічний ключ v, повідомлення $\alpha \in \{0,1\}^l$, при чому $\alpha = \alpha_1 \dots \alpha_l$ та підпис $(s_1, \dots, s_l)$. Алгоритм верифікації підтверджує, що S – дійсний підпис повідомлення α стосовно публічного ключа v тоді і тільки тоді, коли $f(s_i) = y_{i,m_i}$ для $1 \leq i \leq l$.

2.1.1.2 Приклад

Розглянемо приклад застосування схеми Лампорта для підписання повідомлення довжиною три біти. Нехай f – одностороння функція. Тоді публічний ключ v складатиметься з шести елементів: $y_{1,0}, y_{1,1}, y_{2,0}, y_{2,1}, y_{3,0}, y_{3,1}$, а приватний ключ s складатиметься з їх праобразів: $x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, x_{3,0}, x_{3,1}$. Їх можна зобразити як матриці: $s := \begin{pmatrix} x_{1,0} & x_{2,0} & x_{3,0} \\ x_{1,1} & x_{2,1} & x_{3,1} \end{pmatrix}$ та $v := \begin{pmatrix} y_{1,0} & y_{2,0} & y_{3,0} \\ y_{1,1} & y_{2,1} & y_{3,1} \end{pmatrix}$. Для підписання повідомлення $\alpha = \alpha_1\alpha_2\alpha_3$, де кожне α_i представляє один біт, підписувач знаходить відповідний праобраз x_{i,α_i} для кожного $1 \leq i \leq 3$, тобто кожен підпис складатиметься з трьох значень $\left(x_{1,\alpha_1}, x_{2,\alpha_2}, x_{3,\alpha_3}\right)$. Наприклад, підписуючи повідомлення $\alpha = 011$, отримаємо підпис $S = \left(x_{1,0}, x_{2,1}, x_{3,1}\right)$. Далі алгоритм верифікації підтверджує, що підпис дійсний стосовно даного повідомлення тоді і тільки тоді, коли $f(x_i) = y_{i,\alpha_i}$ для $1 \leq i \leq l$. Наприклад, для даного повідомлення $\alpha = 011$ та підпису $S = (x_1, x_2, x_3)$, алгоритм верифікації буде перевіряти, чи рівні між собою $f(x_1)$ та $y_{1,0}$, $f(x_2)$ та $y_{2,1}$, $f(x_3)$ та $y_{3,1}$ відповідно.

2.2 Схеми цифрового підпису, що базуються на складності факторизації великих чисел

2.2.1 Криптосистема RSA

RSA (абревіатура від прізвищ Rivest, Shamir та Adleman) – криптосистема з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел. RSA – перший алгоритм, придатний і для шифрування, і для цифрового підпису [9].

Перевагами RSA є зручне розповсюдження відкритих ключів, що не вимагає секретності; великі мережі мають набагато меншу кількість ключів в порівнянні з асиметричними криптосистемами.

Недоліками RSA є низька швидкість роботи; високі обчислювані витрати зі забезпеченням рівня шифрування щодо спроб фальсифікації; чутливість до мультиплікативної атаки.

2.2.1.1 Формальне визначення

Схема цифрового підпису RSA – це трійка алгоритмів (G, S, V). Вони визначені наступним чином.

Алгоритм генерації ключів $G(1^n)$ обирає два досить великих простих числа p та q довжиною n біт, а потім обчислює їх добуток $N = pq$. Далі обчислюється функція Ейлера $\varphi(N) = (p-1)(q-1)$ та обирається таке ціле число e, що $1 < e < \varphi(N)$ та e взаємнопросте з $\varphi(N)$. За допомогою розширеного алгоритму Евкліда знаходиться число d таке, що $ed \equiv 1 (\mathrm{mod}\, \varphi(N))$. Тоді (N, e) – публічний ключ, а (N, d) – приватний.

Алгоритм підписання повідомлення S отримує на вхід секретний ключ (N, d) та повідомлення α. Він генерує підпис $s = S\big((N, d), \alpha\big) = \alpha^d \ mod\ N$.

Алгоритм верифікації підпису V отримує на вхід публічний ключ (N, e), підпис s та повідомлення α. Він обчислює $\alpha' = s^e \ mod\ N$ та підтверджує, що s – дійсний підпис повідомлення α стосовно публічного ключа (N, e) тоді і тільки тоді, коли $\alpha' = \alpha$.

Вважається, що схема RSA є безпечною від атак класичного комп'ютера, якщо використовується досить великий параметр безпеки, однак квантовий алгоритм Шора здатний розв'язувати задачу факторизації за поліноміальний час.

2.2.1.2 Можливі атаки на RSA

Криптосистема RSA є вразливою до двох основних видів потенційно можливих атак: атаки цілочисельної факторизації та атаки з використанням мультиплікативної властивості RSA.

Атака цілочисельної факторизації полягає у тому, що якщо зловмисник може факторизувати, тобто розкласти на прості множники, публічний модуль N, то потім він зможе обчислити $\varphi(N)$ та, використовуючи алгоритм Евкліда, знайти приватний ключ d з $\varphi(N)$ та публічної експоненти e, розв'язавши рівняння $ed \equiv 1 \; mod \; \varphi$. Щоб уникнути можливості такої атаки, p та q повинні бути такими, щоб факторизація $N = pq$ була дуже складною задачею.

Мультиплікативну властивість RSA іноді називають гомоморфною властивістю. Якщо $s_1 = m_1^d \; mod \; N$ та $s_2 = m_2^d \; mod \; N$ є підписами до повідомлень m_1 та m_2 відповідно, то $s = s_1 s_2$ має таку властивість, що $s = (m_1 m_2)^d \; mod \; n$. Якщо $m = m_1 m_2$, то s є дійсним підписом до повідомлення m.

Вирішенням цієї проблеми є підписання не повідомлення m, а підписання натомість $m' = R(m)$, де R – функція надмірності, яка переводить повідомлення в простір $\mathbb{Z}_n$. Відповідно, при відновленні зашифрованого повідомлення на останньому етапі m обчислюється як $m = R^{-1}(m')$. Важливо, щоб ця функція була не мультиплікативною, тобто для усіх a, b, що належать простору допустимих повідомлень, $R(ab) \neq R(a)R(b)$.

2.2.1.3 Приклад

Нехай алгоритм генерації ключів G отримав числа $p = 3557$ та $q = 2579$. Він отримує модуль $N = pq = 9173503$. Функція Ейлера $\varphi(N) = (p-1)(q-1) = 9167368$. Нехай $e = 3$. Тоді $d = e^{-1}(\mathrm{mod}\, \varphi(N)) = 6111579$.

Тоді $(N, e) = (9173503, 3)$ – публічний ключ, а $(N, d) = (9173503, 6111579)$ – приватний. Алгоритм підписання повідомлення підписує повідомлення $\alpha = 11111$ приватним ключем $(N, d) = (9173503, 6111579)$: $s = S\big((N, d), \alpha\big) = \alpha^d \ mod \ N = 11111^{6111579} \ mod \ 9173503 = \ 4221506$. Алгоритм верифікації підпису V отримує на вхід публічний ключ $(N, e) = (9173503, 3)$, підпис $s = 4221506$ та повідомлення $\alpha = 11111$. Він обчислює $\alpha' = s^e \ mod \ N = 4221506^3 \ mod \ 9173503 = 11111$ та підтверджує, що s – дійсний підпис повідомлення α стосовно публічного ключа (N, e), оскільки $\alpha' = \alpha = 11111$.

2.2.1.4 Використання RSA на практиці

Система RSA використовується у гібридних криптосистемах з симетричними криптографічними алгоритмами. RSA також використовується в операційних системах, в тому числі у системах, які розробляють компанії Microsoft, Apple, Sun та інші. Апаратне застосування RSA включає в себе захищений голосовий зв'язок, мережеві карти Ethernet, смарт-карти та великомасштабні програми в криптографічному обладнанні. Окрім цього, алгоритм RSA широко застосовується основними захищеними протоколами Інтернет-зв'язку, включаючи, але не обмежуючись S/MIME, SSL, S/WAN та державними органами, корпораціями та лабораторіями.

2.2.1.5 Схеми цифрового підпису на основі RSA

Існує декілька схем цифрового підпису, в основі яких лежить криптосистема RSA, а точніше припущення RSA та сильне припущення RSA.

Нехай GM – ймовірнісний алгоритм, що працює за поліноміальний час і на вхід 1^n видає модуль N з двома різними n-бітовими простими числами p та q, при чому $N = qp$. Нехай GP – ймовірнісний алгоритм, що працює за поліноміальний час, який на вхід 1^n видає просте число e довжиною принаймні n, при чому алгоритм GP повинен генерувати прості числа випадковим чином, тобто, навіть якщо ми запускаємо його поліноміально велику кількість разів, ймовірність того,

що він згенерує повторно одне число повинна бути незначною. Якщо e – число, довжиною принаймні n біт та N – результат добутку n-бітних простих чисел, то їх найбільший спільний дільник дорівнює одиниці: $gcd\big(e, \varphi(N)\big) = 1$.

Припущення RSA: задача RSA складно пов'язана з алгоритмом GM, якщо для всіх поліномів p (де $p(n) \geq n$ для всіх n) та ймовірнісних алгоритмів, що працюють за поліноміальний час A, наступна ймовірність незначна: $Pr\big[(N, p, q) \leftarrow GM(1^n); y \leftarrow \mathbb{Z}_N^*; e \leftarrow GP\big(1^{p(n)}\big); x \leftarrow A(N, e, y) : x^e = y \; mod \; N\big]$. На цьому припущенні базуються схеми Дворк-Наора [10] та Крамера-Демґарда [11].

Посилене припущення RSA: задача RSA складно пов'язана з алгоритмами GM та GP, якщо для всіх поліномів p (де $p(n) \geq n$ для всіх n) та ймовірнісних алгоритмів, що працюють за поліноміальний час A, та з публічними монетами наступна ймовірність незначна: $Pr[(N, p, q) \leftarrow GM(1^n); y \leftarrow \mathbb{Z}_N^*; \omega \leftarrow \{0,1\}^n; \; e \leftarrow GP(1^n, \omega); \; x \leftarrow A(N, e, y, \omega) : x^e = y \; mod \; N]$. Іншими словами, посилене припущення RSA пов'язане з «випадковою» великою публічною експонентною e, навіть якщо випадкові жеребкування, що використовуються для генерації, відомі. Дане припущення використовує схема Гогенберг-Ватерс [12].

Сильне припущення RSA: задача RSA складно пов'язана з алгоритмами GM та GP, якщо для всіх поліномів p (де $p(n) \geq n$ для всіх n) та ймовірнісних алгоритмів, що працюють за поліноміальний час A, наступна ймовірність незначна: $Pr[(N, p, q) \leftarrow GM(1^n); y \leftarrow \mathbb{Z}_N^*; (x, e) \leftarrow A(N, y) : e \geq 2 \; \& \; x^e = y \; mod \; N]$. Тут e не обов'язково просте та знімається умова $gcd\big(e, \varphi(N)\big) = 1$. Це припущення є основою для схем Крамера-Шоупа [13], Фішлінга [14] та Геннаро-Халеві-Рабіна [15].

2.2.2 Схема цифрового підпису Рабіна

Схема цифрового підпису Рабіна тісно пов'язана з криптосистемою Рабіна, є одною з найперших схем цифрового підпису [16]. Питання безпеки в ній напряму пов'язане з проблемою цілочисельної факторизації. Її безпечна

конструкція використовує гешування повідомлення. Фактично, схема цифрового підпису Рабіна є модифікацією схеми цифрового підпису RSA.

2.2.2.1 Формальне визначення

Схема цифрового підпису Рабіна – це трійка ймовірнісних алгоритмів (G, S, V), що працюють за поліноміальний час та визначені так.

Нехай $H: \{0,1\}^* \to \{0,1\}^k$ – стійка до колізій геш-функція. Припускається, що H є випадковим оракулом.

Алгоритм генерації ключів G обирає два простих числа p та q, які мають приблизний розмір $k/2$ та $p, q \ mod \ 4 \ = \ 3$. Тоді $f \ = \ pq$. Відкритим ключем буде число f, а закритим – пара (p, q).

Алгоритм підписання повідомлення S для підписання повідомлення α обирає випадкове u та обчислює $H(\alpha, u)$. Якщо виходить так, що $H(\alpha, u)$ не є квадратом за модулем f, обирається нове значення U, тобто шукається таке x, що $x^2 = H(\alpha, u) \ mod \ f$. Підписом повідомлення буде пара (u, x).

Алгоритм верифікації підпису V за даним повідомленням α та підписом (u, x) обчислює $x^2 \ mod \ f$ та $H(\alpha, u) \ mod \ f$ та підтверджує, що (u, x) – дійсний підпис повідомлення α стосовно публічного ключа f тоді і тільки тоді, коли $x^2 \ mod \ f = H(\alpha, u) \ mod \ f$.

Якщо вихідні дані геш-функції H дійсно випадкові, то атака з підробкою повідомлення порівнюється у своїй складності з обчисленням квадратного кореня випадкового елемента в $\mathbb{Z} \ / \ n\mathbb{Z}$.

2.3 Схеми цифрового підпису, що базуються на складності обчислення дискретного логарифму

2.3.1 Схема Ель-Гамаля

Схема Ель-Гамаля – схема цифрового підпису, що базується на складності обчислення дискретного логарифму та алгебраїчних властивостях модульного

піднесення до степеня. Схема була описана Тахером Ель-Гамалем у 1985 році [17].

Перевагами схеми Ель-Гамаля є імовірнісний характер шифрування, забезпечує високий рівень стійкості; можливість генерувати цифрові підписи для великої кількості повідомлень з використанням одного секретного ключа.

Недоліком схеми Ель-Гамаля є подвоєння довжини зашифрованого тексту в порівнянні з його початковою довжиною, що спричиняє більший час та створює жорсткі вимоги до каналу зв'язку.

2.3.1.1 Формальне визначення

Схема цифрового підпису Ель-Гамаля – це трійка алгоритмів (G, S, V). Вони визначені наступним чином. Нехай H – геш-функція, що стійка до колізій.

Алгоритм генерації ключів G знаходить досить велике просте число p довжиною n біт, де n – параметр безпеки. Він випадковим чином обирає генератор g цілочисельної мультиплікативної групи за модулем p: $g \in \mathbb{Z}_p^*$. Далі алгоритм генерації ключів випадковим чином обирає секретний ключ x такий, що $1 < x < p - 1$ та обчилює $y = g^x \ mod \ p$. Тоді публічним ключем буде структура (p, g, y), а приватним – значення x.

Алгоритм підписання повідомлення S для підписання повідомлення α виконує наступні кроки. Обирає випадкове k таке, що $1 < k < p - 1$ та найбільше спільне кратне k та $p - 1$ рівне одиниці: $gcd(k, p - 1) = 1$. Обчислює $r \equiv g^k \ (mod \ p)$ та $s \equiv (H(\alpha) - xr)k^{-1} (mod \ p - 1)$. У малоймовірному випадку, якщо $s = 0$, алгоритм підписання виконується знову. Підписом повідомлення є пара (r, s).

Алгоритм верифікації підпису V отримує на вхід публічний ключ (p, g, y), підпис (r, s) та повідомлення α. Він перевіряє, чи $g^{H(\alpha)} = y^r r^s \ (mod \ p)$, $0 < r < p$ та $0 < s < p - 1$ підтверджує, що s – дійсний підпис повідомлення α, тоді і тільки тоді, коли усі умови виконуються.

Алгоритм є коректним у тому сенсі, що згенеровані підписи завжди можна буде перевірити і вони дійсно свідчитимуть про те, чи було насправді підписане повідомлення. Генерація підписів фактично реалізує $H(\alpha) = xr + sk \ (mod\ p-1)$, а з малої теореми Ферма маємо, що $g^{H(\alpha)} \equiv g^{xr} g^{ks} \equiv (g^x)^r (g^k)^s \equiv (g)^r (g)^s \ (mod\ p)$.

Зловмисник може підробити підписи знайшовши секретне значення x або знайшовши колізій в геш-функції $H(\alpha) = H(A)(mod\ p-1)$. Обидві задачі вважаються досить складними. Підписувач повідомлення повинен бути уважним при виборі різних рівномірно випадкових k для різних повідомлень, оскільки з них зловмисник може легко знайти секретний ключ x.

2.3.1.2 Приклад

Нехай H – геш-функція, що стійка до колізій, припустимо, що $H(\alpha) = \alpha$.

Алгоритм генерації ключів G знаходить просте число $p = 23$. Він випадковим чином обирає генератор $g = 5$ цілочисельної мультиплікативної групи за модулем p: $g \in \mathbb{Z}_{23}^*$. Далі алгоритм генерації ключів випадковим чином обирає секретний ключ $x = 3$ такий, що $1 < x < 22$ та обчилює $y = g^x \ mod\ p = 5^3 \ mod\ 23 = 10$. Тоді публічним ключем буде структура $(23, 5, 10)$, а приватним – значення 3.

Алгоритм підписання повідомлення S для підписання повідомлення $\alpha = 7$ виконує наступні кроки. Обирає випадкове $k = 9$ таке, що $1 < k < 22$ та найбільше спільне кратне k та 22 рівне одиниці: $gcd(9, 22) = 1$. Обчислює $r \equiv g^k \ (mod\ p) = 5^9 \ (mod\ 23) = 11$ та $s \equiv (H(\alpha) - xr)k^{-1}(mod\ p-1) = (H(7) - 33)9^{-1}(mod\ 22) = -5(7-33)(mod\ 22) = -2 \ (mod\ 22) = 20 \ (mod\ 22)$. Підписом повідомлення є пара $(11, 20)$.

Алгоритм верифікації підпису V отримує на вхід публічний ключ $(23, 5, 10)$, підпис $(11, 20)$ та повідомлення $\alpha = 7$. Він перевіряє, чи $5^{H(7)} = 5^7 = y^r r^s \ (mod\ p) = 10^{11} 11^{20} \ (mod\ 23) = 17$, $0 < r < 23$ та $0 < s < 22$ підтверджує, що s – дійсний підпис повідомлення α.

2.3.1.3 Використання схеми цифрового підпису Ель-Гамаля на практиці

Схема цифрового підпису широко застосовується в сертифікатах відкритих ключів з метою захисту з'єднань у TLS, а саме в SSL, HTTPS, WEB; з метою захисту повідомлень у XML-підписі, тобто у шифруванні XML, та для забезпечення цілісності IP-адрес та доменних імен DNSSEC.

2.3.2 Алгоритм цифрового підпису DSA

Алгоритм цифрового підпису DSA використовує ідеї підпису Ель-Гамаля та Шнорра, також базується на складності взяття логарифмів в скінченних полях. Був запропонований в 1991 Національним інститутом стандартів [18] та технологій США і є запатентованим.

Перевагами схеми DSA є менша довжина підпису в порівнянні зі схемою Ель-Гамаля при однаковому рівні безпеки; менший час на обчислення в порівнянні зі схемою Ель-Гамаля; менші витрати простору для зберігання даних.

Недоліком схеми DSA є те, що верифікація підпису може спричиняти складні залишкові оператори, що значно впливає на швидкість обчислень.

2.3.2.1 Формальне визначення

Схема цифрового підпису DSA – це трійка алгоритмів (G, S, V). Вони визначені наступним чином.

Алгоритм генерації ключів G складається з двох етапів. На першому етапі обираються основні параметри системи. Нехай H – криптографічна геш-функція, що стійка до колізій. Зазвичай використовуються геш-функції сімейства SHA-2. Обирається просте число q, розмір якого в бітах дорівнює розмірності значень геш-функції $H(x)$, вважатимемо, що така кількість бітів – N. Обирається просте число p, таке, що q є дільником $(p-1)$. Тоді позначимо як L бітову довжину p, $2^{L-1} < p < 2^L$. Далі обирається таке число g, що його мультиплікативний порядок по модулю p рівний q. Це робиться за допомогою формули $g = h^{(p-1)/q}$,

де h – довільне число з діапазону $(1, p-1)$ та $g \neq 1$. Досить часто $h = 2$ задовольняє таку умову. Параметри (p, q, g) можуть використовувати будь-які користувачі системи. Найголовнішими параметрами безпеки схеми є геш-функція H, бітова довжина вхідної послідовності N та параметр L. В стандарті цифрових підписів зазначаються такі можливі значення пар (L, N): $(1024, 160), (2048, 224), (2048,256), (3072,256)$. Їх рекомендується використовувати з сімейством геш-функцій SHA-2. На другому етапі генеруються приватний та публічний ключі для одного користувача. Випадковим чином обирається таке x, що $0 < x < q$. Далі обчислюється $y = g^x \ mod \ p$. Тоді відкритим ключем буде структура (p, q, g, y), а закритим – значення x.

Алгоритм підписання повідомлення S для підписання повідомлення α виконує наступні кроки. Обирає випадкове k таке, що $1 < k < q$. Обчислює $r \equiv g^k \ (mod \ q)$ та $s \equiv (H(\alpha) + xr)k^{-1} (mod \ q)$. У малоймовірному випадку, якщо $s = 0$ або $r = 0$, алгоритм підписання виконується знову. Підписом повідомлення є пара (r, s).

Алгоритм верифікації підпису V отримує на вхід публічний ключ (p, q, g, y), підпис (r, s) та повідомлення α. Він перевіряє, чи $0 < r < q$ та $0 < s < q$, якщо ні, то підпис не є дійсним. Далі обчислюються $w = (s)^{-1} \ mod \ q, u_1 = (H(\alpha)w) mod \ q, u_2 = (rw) mod \ q, v = \left((q^{u_1} y^{u_2}) \ mod \ p\right) mod \ q$. Алгоритм V підтверджує, що s – дійсний підпис повідомлення α, тоді і тільки тоді, коли $v = r$.

Алгоритм є коректним у тому сенсі, що згенеровані підписи завжди можна буде перевірити і вони дійсно свідчитимуть про те, чи було насправді підписане повідомлення. Якщо $g = h^{(p-1)/q}$, то $g^q \equiv h^{p-1} \equiv 1 \ (mod \ p)$ за малою теоремою Ферма. Оскільки $g > 1$ та q – просте, g повинне мати порядок q. Підписант обчислює $s \equiv (H(\alpha) + xr)k^{-1} (mod \ q)$, тож $k \equiv H(\alpha)s^{-1} + xrs^{-1} = H(\alpha)w + xrw$. Оскільки g має порядок $q \ (mod \ p)$: $g \equiv g^{H(\alpha)w} g^{xrw} \equiv g^{H(\alpha)w} g^{rw} \equiv g^{u_1} g^{u_2} \ (mod \ p)$. Нарешті, коректність DSA випливає з $r = (g^k \ mod \ p) \ mod \ q = \left(g^{u_1} y^{u_2} \ (mod \ p)\right) mod \ q = v$.

2.3.2.2 Використання схеми цифрового підпису DSA на практиці

Алгоритм був запропонований Національним інститутом стандартів та технологій США, є запатентованим, але доступним для використання без ліцензійних відрахувань. Разом з криптографічною геш-функцією SHA-1 є частину стандарту цифрового підпису DSS, опублікованого в 1994 році. Використовується як в секретних, так і не в секретних комунікаціях.

2.3.3 Схема цифрового підпису Шнорра

Цифровий підпис Шнорра генерується за допомогою алгоритму підпису Шнорра [19]. Схема цифрового підпису Шнорра відома своєю простотою і базується на нерозв'язності дискретного логарифму, є досить ефективною і генерує короткі підписи.

2.3.3.1 Формальне визначення

Схема цифрового підпису Шнорра – це трійка ймовірнісних алгоритмів (G, S, V), що працюють за поліноміальний час та визначені так.

Нехай $H: \{0,1\}^* \to \mathbb{Z}_p$ – стійка до колізій геш-функція. Припускається, що H є випадковим оракулом. Нехай G – група простого порядку q з генератором g, в якій важко розв'язати задачу пошуку дискретного логарифму. Зазвичай використовують групу Шнорра. Це велика підгрупа простого порядку $\mathbb{Z}_p^\times$, мультиплікативної групи цілих чисел за модулем p для деякого простого p. Для генерації такої групи знаходять числа p, q, r такі, що $p = qr + 1$, де p, q – прості. Далі обирається будь-яке h таке, що $1 < h < p$ поки не знайдеться таке, що $h^r \not\equiv 1 \ (mod\ p)$. Значення $g = h^r \ mod\ p$ є генератор підгрупи $\mathbb{Z}_p^\times$ порядку q.

Алгоритм генерації ключів G. Обирається просте число q, розмір якого в бітах дорівнює розмірності значень геш-функції $H(x)$, вважатимемо, що така кількість бітів – N. Обирається просте число p, таке, що q є дільником $(p-1)$. Тоді позначимо як L бітову довжину p, $2^{L-1} < p < 2^L$. Далі обирається таке число

g, що його мультиплікативний порядок по модулю p рівний q. Це робиться за допомогою формули $g = h^{(p-1)/q}$, де h – довільне число з діапазону $(1, p-1)$ та $g \neq 1$. Досить часто $h = 2$ задовольняє таку умову. Параметри (p, q, g) можуть використовувати будь-які користувачі системи. Випадковим чином обирається таке w, що $0 < w < q$. Далі обчислюється $y = g^{-w} \ mod \ p$. Тоді відкритим ключем буде структура (p, q, g, y), а закритим – значення w.

Алгоритм генерації підпису S для підписання повідомлення α обирає випадкове число r таке, що $0 < r < q$ та обчислює $x = g^r \ (mod \ p)$. Важливо, що це r повинне бути різним для різних повідомлень. Далі обчислюється $s_1 = H(\alpha|x)$, де $\alpha|x$ – об'єднання повідомлення та знайденого значення x в один рядок. Далі $s_2 = r + ws_1 \ mod \ q$. Тоді підписом буде пара (s_1, s_2).

Алгоритм верифікації підпису V отримує на вхід повідомлення α, підпис (s_1, s_2) та публічний ключ (p, q, g, y). Обчислюється $X = q^{s_2} y^{s_1} \ mod \ p$. Підпис (s_1, s_2) вважається дійсним підписом повідомлення α тоді і тільки тоді, коли $H(\alpha|X) = s_1$.

Алгоритм є коректним у тому сенсі, що згенеровані підписи завжди можна буде перевірити і вони дійсно свідчитимуть про те, чи було насправді підписане повідомлення. Легко показати, що $H(\alpha|X) = s_1$ якщо підписане повідомлення ідентичне тому, яке використовує алгоритм верифікації. $X = q^{r+ws_1} \ g^{ws_2} \ mod \ p = g^r = x$, звідси $H(\alpha|X) = H(\alpha|x) = s_1$.

2.3.3.2 Використання схеми цифрового підпису Шнорра

Схема цифрового підпису Шнорра лежить в основі багатьох інших схем цифрового підпису, які є актуальними та широко використовуються у наш час. Алгоритм цифрового підпису Шнорра лежить в основі національного стандарту цифрового підпису Республіки Білорусь та в основі стандартів KCDSA та EC-KCDSA.

Важливою особливістю схеми цифрового підпису Шнорра є те, що за дизайном вона не вимагає великих обчислень на стороні користувача, тому може

використовуватись навіть на обчислювано слабких платформах, як от смарт-карти чи RFID, а також на платформах, що не мають апаратної підтримки для множинної точності арифметики.

Особливо застосованою є модифікація алгоритму цифрового підпису Шнорра, яка використовує не скінченні поля для обчислень, а еліптичні криві натомість. Модифікація схеми цифрового підпису Шнорра описана в протоколі прикладного рівня CryptoNote та лежить в основі конфіденційних транзакцій криптовалюти Monero.

2.3.4 ECDSA

Elliptic Curve Digital Signature Algorithm – ECDSA – алгоритм цифрового підпису, який вперше був запропонований у 1992 році [20]. Він використовує ідеї DSA, однак визначений не над кільцем цілих чисел, а базується на використанні криптографії на еліптичних кривих. Тобто стійкість алгоритму заключається в складності обчислення дискретного логарифму над полем цілих чисел.

Перевагами схеми ECDSA є придатність до застосування в набагато менших полях, ніж алгоритм DSA; відсутність проблем з продуктивністю; швидкий процес підписання повідомлення та верифікації підпису; відповідність постійно зростаючим вимогам безпеки; підтримка національних стандартів захисту.

Недоліком схеми ECDSA є можливість виникнення помилки, яка дасть змогу вибрати значення приватного ключа та отримувати однакові підписи для різних документів, однак це можливо лише з величезними обчислюваними результатами.

2.3.4.1 Формальне визначення

Схема цифрового підпису ECDSA – це трійка алгоритмів (G, S, V). Вони визначені наступним чином. Нехай H – криптографічна геш-функція, що стійка до колізій. Зазвичай використовуються геш-функції сімейства SHA-2.

У ECDSA пари з відкритого та закритого ключів створюються залежно від певного набору доменних параметрів. Такі параметри можуть використовуватись групою користувачів та можуть бути загальнодоступними, а також залишатись фіксованими протягом тривалого періоду часу. Доменні параметри ECDSA можна записати у вигляді такої структури: $(E_p(a,b), G, n)$, де $E_p(a,b)$ – обрана еліптична крива, її поле та рівняння, що задається параметрами a та b, число її точок кратне параметру n, G – генератор або базова точка циклічної підгрупи еліптичної кривої, n – просте ціле число, мультиплікативний порядок точки G, тобто $n \times G = O, O$ – елемент ідентичності. Тут і далі $\times$ позначає операцію множення точки еліптичної кривої на скалярне значення.

Алгоритм генерації ключів G випадково обирає значення d в інтервалі $(1, n)$, а потім обчислює значення $Q = d \times G$. Тоді приватним ключем буде d, а публічним – структура (a, b, G, n, Q).

Алгоритм генерації підпису S для підписання повідомлення α виконує наступні кроки. Спочатку обчислюється геш значення повідомлення α, позначимо його як $l = H(\alpha)$. Далі для підписання можуть використовуватись крайні ліві біти гешу l, кількість яких дорівнює бітовій довжині порядку групи n. Потім криптографічно безпечним випадковим чином обирається випадкове ціле число k інтервалі $(1, n)$, а потім обчислюється точка кривої $(x_1, y_1) = k \times G$. Тоді $r = x_1 \ mod \ n$, якщо вийде так, що $r = 0$, то випадкове k обирається знову і для нього обчислюється нова точка (x_1, y_1). Далі, з використанням значення r обчислюється $s = k^{-1}(H(\alpha) + rd) \ mod \ n$. Якщо $s = 0$, то випадкове k обирається знову і для нього повторюються усі наступні обчислення. Підписом буде пара (r, s). Пара $(r, -s \ mod \ n)$ також є дійсним підписом. Важливо, щоб значення параметру k було секретним та не використовувалось для створення різних підписів повторно. Інакше, маючи два різних підписи (r, s) та (r', s') з використанням одного k' для різних повідомлень α та α' зловмисник може обчислити $H(\alpha)$ та $H(\alpha')$, звідки $s - s' = k^{-1}\big(H(\alpha) - H(\alpha')\big) \ mod \ n$. Тоді зловмисник може отримати $k = \big(H(\alpha) - H(\alpha')\big)/(s - s')$, а оскільки $s = k^{-1}(H(\alpha) + rd) \ mod \ n$, то і $d = \frac{sk - H(\alpha)}{r}$.

Алгоритм верифікації підпису V перевіряє чи є точка Q допустимою точкою еліптичної кривої. Для цього досліджуються три твердження: точка Q не дорівнює ідентичному елементу O та її координати є допустимим; точка Q лежить на кривій; добуток порядку генератора G, n, на точку Q, рівний ідентичному елементу, тобто $n \times Q = O$. Якщо хоч одне з тверджень не є істинним, алгоритм верифікації підпису V вважає, що підпис не є дійсним. Інакше він перевіряє, чи належать частини підпису (r, s) діапазону $(1, n)$, тобто чи справджуються нерівності $1 < r < n$ та $1 < s < n$ одночасно. Якщо ні, підпис не є дійсним, інакше виконуються наступні кроки. Обчислюється геш значення повідомлення α, позначимо його як $l = H(\alpha)$. Далі для перевірки можуть використовуватись крайні ліві біти гешу l, кількість яких дорівнює бітовій довжині порядку групи n. Обчислюються значення $u_1 = ls^{-1} mod\ n$ та $u_2 = rs^{-1} mod\ n$. Шукається точка кривої $(x_1, y_1) = u_1 \times G + u_2 \times Q$. Якщо $(x_1, y_1) = O$, підпис не є дійсним. Підпис є дійсним тоді і тільки тоді, коли $r \equiv x_1\ (mod\ n)$.

Алгоритм є коректним у тому сенсі, що згенеровані підписи завжди можна буде перевірити і вони дійсно свідчитимуть про те, чи було насправді підписане повідомлення. Позначимо точку кривої з алгоритму верифікації підпису як $P = (x_1, y_1) = u_1 \times G + u_2 \times Q$. За визначенням, $Q = d \times G$, тому $P = u_1 \times G + u_2 d \times G$, звідси з властивостей точок еліптичних кривих маємо, що $P = (u_1 + u_2 d) \times G$. З визначень u_1 та u_2 маємо, що $P = (ls^{-1} + rs^{-1} d) \times G = (l + rd)s^{-1} \times G$. Враховуючи, що в алгоритмі підписання повідомлення S $s = k^{-1}(H(\alpha) + rd)\ mod\ n$, $P = (l + rd)(l + rd)^{-1}(k^{-1})^{-1} \times G$. Оскільки обернене значення оберненого значення елемента є самим елементом, а добуток елемента на обернений утворюють нейтральний елемент, отримуємо $P = k \times G$, що і перевіряється в алгоритмі верифікації в $r \equiv x_1\ (mod\ n)$. Звідси маємо, що правильний підпис завжди буде дійсним.

2.3.4.2 Використання схеми цифрового підпису ECDSA

Алгоритм цифрового підпису ECDSA застосовується в протоколах TLS, PGP, SHH. Також алгоритм ECDSA має широке застосування в криптовалютах та блокчейні, особливо хорошиш прикладом системи, що покладається на ECDSA є криптовалюта Біткоін та похідні від неї.

2.3.5 Схема цифрового підпису BLS (Boneh-Lynn-Shacham)

Схема цифрового підпису BLS спирається на криві, зручні для сполучення [21]. Використовуючи схему BLS можна створювати короткі підписи для великих груп підписів, наприклад, для набору підписів $(\sigma_1, \dots, \sigma_n)$ можна створити загальний підпис σ, через який можна буде верифікувати увесь набір підписів $(\sigma_1, \dots, \sigma_n)$. Алгоритм використовує модифіковані геш-функції, які на виході дають не число, а координати точки на еліптичній кривій.

Для алгоритму BLS є важливим поняття розривної групи – групи, в якій обчислювана задача Діффі-Хелмана неров'язна, але вирішальна задача Діффі-Хелмана може бути ефективно вирішена. Невироджені, ефективно обчислювані білінійні сполучення дозволяють такі групи.

Обчислювана задача Діфффі-Хелмана полягає в тому, щоб за заданим значенням елементу g, а також маючи значення g^x та g^y знайти значення g^{xy}. Формально g – генератор деякої групи, x та y – деякі цілі числа.

Вирішальна задача Діффі-Хелмана полягає у тому, що у циклічній мультиплікативній групі G порядку q з генератором g, маючи рівномірно обрані $a, b \in \mathbb{Z}_q$ та з відомими g^a та g^b, значення g^{ab} «схоже» на випадкове.

2.3.5.1 Формальне визначення

Схема цифрового підпису BLS – це трійка ймовірнісних алгоритмів (G, S, V), що працюють за поліноміальний час та визначені так.

Нехай $H(x)$ – функція, яка дозволяє гешувати повідомлення в еліптичну криву. Нехай G, G_T – циклічні групи простого порядку r з генератором g.

Сполученням називається ефективно обчислювана функція $e: G_1 \times G_2 \to G_T$, яка є невиродженою, тобто $e(g, g) \neq 1$, та білінійною, тобто $e(g^a, g^b) = e(g, g)^{ab}, a, b \in \mathbb{Z}$.

Алгоритм генерації ключів G випадковим чином обирає приватний ключ s такий, що $0 \leq s \leq r - 1$. Тоді публічним ключем буде значення $v = g^s$.

Алгоритм генерації підпису S отримує на вхід повідомлення α та приватний ключ s. Він гешує повідомлення α в криву, тобто $h = H(\alpha)$, а потім обчислює підпис $\sigma = h^s$.

Алгоритм верифікації підпису V отримує на вхід повідомлення α, підпис σ та публічний ключ v. Він обчислює $d_1 = e(v, h)$ та $d_2 = e(g, \sigma) = e(g, h^s) = e(g^s, h)$. Підпис σ вважається дійсним підписом повідомлення α стосовно публічного ключа v тоді і тільки тоді, коли $d_2 = d_1$.

Нехай маємо групу підписів, яка містить n пар (σ_i, v_i). Тоді скажемо, що агрегованим підписом σ_A цієї групи є сума σ_i для всіх $i = 1, \dots, n$. Для верифікації цього підпису необхідно перевірити рівність $e(g, \sigma) = e(v_1, h_1) \dots e(v_n, h_n)$. Дійсно, $e(g, \sigma) = e(g, \sigma_1 + \cdots + \sigma_n) = e(g, \sigma_1) \dots e(g, \sigma_n) = e\left(g, h_1^{v_1}\right) \dots e\left(g, h_1^{v_1}\right) = e\left(h_1^{v_1}, g\right) \dots e\left(h_n^{v_n}, g\right) = e(v_1, h_1) \dots e(v_n, h_n)$

2.3.5.2 Використання схеми цифрового підпису BLS

До 2020 року алгоритм цифрового підпису BLS широко використовувався у другій версії блокчейну Ethereum, як зазначено в проекті специфікації блокчейну – для криптографічного підтвердження того, що конкретний валідатор Ethereum підтвердив транзакцію. Використання підписів BLS в Ethereum не вважалось довгостроковим, оскільки потенційно підписи BLS не є квантово захищеними.

Конфіденційна криптовалюта Navcoin використовує підписи BLS у своєму протоколі конфіденційності.

2.4 Кільцеві схеми цифрового підпису

2.4.1 Схема кільцевого підпису на основі RSA

Схема кільцевого підпису на основі RSA – схема, яку Рівест, Шамір та Тауман запропонували у своїй оригінальній роботі про кільцеві підписи [5].

2.4.1.1 Формальне визначення

Нехай $C_{k,v}(y_1, y_2, \dots, y_n)$ – комбінуюча функція. Вона отримує на вхід ключ k, ініціалізаційне значення v та ряд довільних значень $y_1, y_2, \dots, y_n$. На виході вони генерує одне значення z. Функцію $C_{k,v}(y_1, y_2, \dots, y_n)$ можна обчислити для будь-якого входу з y_i з $\{0,1\}^b$, однак неможливо обчислити $C_{k,v}(g_1(x_1), g_2(x_2), \dots, g_n(x_3))$ для $x_1, x_2, \dots, x_3$, якщо зловмисник не може обчислити обернене значення жодної з односторонніх функцій з секретом $g_1, g_2, \dots, g_n$. Функції g_i визначені таким чином, що з деякими секретними даними k_i легко обчислити x таке, що $y = g_i(x)$, а $y = g_i(x)$ можна обчислити і без k_i. Функція $C_{k,v}(y_1, y_2, \dots, y_n)$ називається кільцевим рівнянням та базується на симетричній функції шифрування E_k. Вона визначається так: $C_{k,v}(y_1, y_2, \dots, y_n) = E_k\left(y_n \oplus E_K\left(y_{n-1} \oplus E_K\left(\dots \oplus E_k(y_1 \oplus v)\right)\right)\right) = v$. В схемі кільцевого підпису вихід v – те ж саме, що і вхід. Функція тривіальна при $n - 1$ параметрі.

$H: \{0,1\}^* \to \{0,1\}^k$ – стійка до колізій геш-функція. Припускається, що H є випадковим оракулом.

Нехай i-му користувачу кільця з r користувачів необхідно підписати повідомлення α своїм приватним ключем s_i, при цьому відомі публічні ключі усіх користувачів кільця $v_1, v_2, \dots, v_n$.

Алгоритм підписання повідомлення RS спочатку обчислює симетричний ключ k як геш повідомлення, що необхідно підписати: $k = H(\alpha)$, у більш складному варіанті може використовуватись натомість $k = H(\alpha, v_1, \dots, v_2)$. Далі рівномірно обирається випадкове значення v з $\{0,1\}^b$. Випадковим чином з $\{0,1\}^b$ рівномірно обираються x_i для усіх не-підписантів $1 \le i \le r, i \ne s$ та

обчислюються $y_i = g_i(x_i)$. Обчислюється кільцеве рівняння для y_s: $C_{k,v}(y_1, y_2, \dots, y_n) = v$. Далі це значення використовується для отримання x_s з односторонньої функціїї зі своїм приватним ключем: $x_s = g_{s_i}^{-1}(y_s)$. Підписом повідомлення α буде $(2r+1)$-розмірна структура: $\sigma = (v_1, \dots, v_n, v, x_1, \dots, x_r)$.

Алгоритм верифікації підпису RV отримує на вхід повідомлення α, підпис σ та набір з публічних ключів користувачів кільця $v_1, v_2, \dots, v_n$. Для усіх $i = 1, 2, \dots, r$ обчислюється $y_i = g_i(x_i)$. Далі з гешу повідомлення α знаходиться ключ симетричного шифрування k: $k = H(\alpha)$. Підпис σ вважається дійсним підписом повідомлення α стосовно публічних ключів $y_1, y_2, \dots, y_n$ тоді і тільки тоді, коли для $v_i, i = 1, \dots, r$ рівність істинна: $C_{k,v}(y_1, y_2, \dots, y_n) = v$.

2.4.2 Схема кільцевого підпису на основі схеми Шнорра

Схема кільцевого підпису на основі схеми Шнорра булав перше описана як схема «1-з-n» підпису у 2002 році. [18]

2.4.2.1 Формальне визначення

Нехай у кільці r учасників, для кожного $i = 0, \dots, r-1$ визначимо індивідуальні параметри так.

Позначимо як p_i та q_i великі прості числа. Тоді $\langle g_i \rangle$ – проста підгрупа $\mathbb{Z}_{p_i}^*$ порядку q_i, згенерована g_i. Нехай x_i, y_i такі, що $y_i = g_i^{x_i} \bmod p_i$. Тоді x_i – приватний ключ, а (y_i, p_i, q_i, g_i) – публічний ключ. Визначимо геш-функцію $H_i: \{0,1\}^* \to \mathbb{Z}_p$.

Тоді L – список, що містить публічні ключі учасників кільця.

Нехай k-му користувачу кільця необхідно підписати повідомлення m своїм приватним ключем x_k.

Алгоритм підписання повідомлення RS на етапі ініціалізації обирає випадкове $\alpha \leftarrow \mathbb{Z}_{q_i}$ та обчислює $c_{k+1} = H_{k+1}(L, m, g_k^{\alpha} \bmod p_k)$. Далі для кожного $i = k+1, \dots, n-1, 0, \dots, k-1$ обирає $s_i \leftarrow \mathbb{Z}_{q_i}$ та обчислює $c_{i+1} = H_{i+1}\left(L, m, g_i^{s_i} y_i^{c_i} \bmod p_i\right)$. Нарешті, для формування кільця обчислюється $s_k =$

$\alpha - x_k c_k \ mod \ q_k$. Підписом до повідомлення m стосовно приватного ключа x_k та набору публічних ключів L буде структура $\sigma = (\mathrm{c}_0, s_0, s_1, \dots, s_{r-1})$.

Алгоритм верифікації підпису RV отримує на вхід повідомлення m, підпис σ та набір з публічних ключів користувачів кільця L. Для усіх $i = 0, 1, 2, \dots, r-1$ обчислюється $e_i = g_i^{s_i} y_i^{c_i} \ mod \ p_i$ та $c_{i+1} = H_{i+1}(L, m, e_i)$ якщо $i \neq r-1$. Підпис σ вважається дійсним підписом повідомлення m стосовно публічних ключів L тоді і тільки тоді, коли для істинна рівність $c_0 = H_0(L, m, e_{r-1})$.

Фактично, схема є кільцем з підписів Шнорра, де кожне обчислення проводиться з врахуванням обчислень на попередньому кроці. Якщо кількість учасників буде рівня одиниці, тобто $r = 1$, то схема перетвориться в звичайну схему Шнорра.

2.4.3 Зв'язна схема кільцевого підпису на основі схеми Шнорра LSAG

Зв'язна схема кільцевого підпису на основі Шнорра, тобто схема цифрового підпису, яка дозволяє пов'язати два кільцевих підписи в одному кільці з одним підписувачем без розкриття його особи, була вперше представлена у 2004 році як спонтанний анонімний підпис для спеціальних груп LSAG. [23]

2.4.3.1 Формальне визначення

Нехай у кільці r учасників, для кожного $i = 1, \dots, r$ визначимо індивідуальні параметри так.

Позначимо як p та q великі прості числа. Тоді $G = \langle g \rangle$ – проста підгрупа $\mathbb{Z}_p^*$ порядку q, згенерована g. Нехай x_i, y_i такі, що $y_i = g^{x_i} \ mod \ p$. Тоді x_i – приватний ключ, а y_i – публічний ключ. Визначимо геш-функції $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q$ та $H_2 : \{0,1\}^* \rightarrow G$.

Тоді $L = \{y_1, \dots, y_r\}$ – список, що містить r публічних ключів учасників кільця.

Нехай k-му користувачу кільця необхідно підписати повідомлення m своїм приватним ключем x_k.

Алгоритм підписання повідомлення RS на етапі ініціалізації обчислює $h = H_2(L)$ та $y' = h^{x_k}$, обирає випадкове $u \leftarrow \mathbb{Z}_q$ та знаходить $c_{k+1} = H_1(L, y', m, g^u \bmod p, h^u \bmod p)$. Далі для кожного $i = k+1, \dots, n-1, 0, \dots, k-1$ обирає $s_i \leftarrow \mathbb{Z}_q$ та обчислює $c_{i+1} = H_1\left(L, y', m, g^{s_i} y_i^{c_i} \bmod p, h^{s_i} y'^{-c_i} \bmod p\right)$. Нарешті, для формування кільця обчислюється $s_k = u - x_k c_k \bmod q$. Підписом до повідомлення m стосовно приватного ключа x_k та набору публічних ключів L буде структура $\sigma = (\mathrm{c}_0, s_1, \dots, s_r, y')$.

Алгоритм верифікації підпису RV отримує на вхід повідомлення m, підпис σ та набір з публічних ключів користувачів кільця L. Алгоритм знаходить значення $h = H_2(L)$. Для усіх $i = 1, 2, \dots, r$ обчислюється $z_i' = g^{s_i} y_i^{c_i} \bmod p$, $z_i'' = h^{s_i} y'^{-c_i} \bmod p$ та $c_{i+1} = H_1(L, y', m, z_i', z_i'')$ якщо $i \neq r$. Підпис σ вважається дійсним підписом повідомлення m стосовно публічних ключів L тоді і тільки тоді, коли для істинна рівність $c_1 = H_1(L, y', m, z_n', z_n'')$.

2.4.3.2 Зв'язність

Нехай є фіксований список публічних ключів $L = \{y_1, \dots, y_r\}$ та два підписи $\sigma_L'(m') = (c_1', s_1', \dots, s_r', y'')$ та $\sigma_L''(m'') = (c_1'', s_1'', \dots, s_r'', y''')$, де m' та m'' позначають деякі повідомлення. Сторона, що перевіряє підписи, може визначити, чи були вони ініційовані одним і тим же учасником кільця, перевіривши, чи $y'' = y'''$. Якщо рівність істинна, то автор один, якщо вони відрізняються, то автори різні.

Можливий сценарій, за якого «слідчий» для дійсного підпису $\sigma_{L(m)} = (c_1, s_1, \dots, s_r, y')$, повідомлення m та списку публічних ключів L зможе встановити, чи справді конкретний користувач i ініціював створення підпису. Для цього він повинен отримати від користувача i його приватний ключ x_i. Якщо приватний ключ x_i пов'язаний з одним з публічних ключів $y_i \in L$ (тобто $y_i = g^{x_i}$) та $y' = H_2(L)^{x_i}$, «слідчий» може підтвердити, що підпис створив користувач i.

2.4.3.3 Пороговість

Розширення 1-з-n схеми кільцевого підпису LSAG до порогової t-з-n схеми є тривіальною задачею, якщо для цього використовується зв’язна властивість. Кожен користувач просто генерує 1-з-n підпис LSAG, а потім вони з’єднують їх разом для того, щоб сформувати t-з-n пороговий LSAG підпис. Використовуючи зв'язну властивість, кожен користувач може переконатись, що усі підписи були згенеровані різними t користувачами.

2.4.3.4 Використання схеми зв’язного кільцевого цифрового підпису на основі схеми цифрового підпису Шнорра для голосування

Схема кільцевого підпису LSAG може використовуватись як основа для анонімних голосувань. Вона складається з двох фаз: фази голосування та фази відкриття голосів. Припускається, що інфраструктура така: кожен виборець має власний відкритий опублікований ключ. Існує надійний список відкритих ключів усіх виборців, які можна завантажити з надійного сховища, наприклад, перелік відкритих ключів усіх громадян. Припустимо, що усі виборці мають ключі дискретного логарифму p, q, g. Позначимо список виборців як L.

Для початку виборів деякі повідомлення генеруються надійними засобами. Потім повідомлення публікуються. Для спрощення будемо вважати, що це референдум з допустимими відповідями «так» чи «ні». В такому випадку будуть опубліковані повідомлення m_1 = так та m_2 = ні.

Під час фази голосування кожен «так»-виборець підписує повідомлення m_1 та список відкритих ключів L. Кожен «ні»-виборець робить те ж саме з повідомленням m_2.

На фазі відкриття голосі просто підраховуються підписи за повідомлення m_1 та за повідомлення m_2. Якщо хтось проголосував двічі, це легко буде відстежити через зв'язну властивість.

2.4.3.5 Використання схеми LSAG у криптовалютах

Схема цифрового підпису LSAG описана у протоколі CryptoNote, на якому базуються анонімні криптовалюти, що підтримують конфіденційні транзакції. Наприклад, Monero [24], ByteCoin, DarkNote, Carbo. Кільцеві підписи в них приховують відправника у транзакціях. На відміну від Біткоіна, у цих криптовалютах не можна ідентифікувати особистість власника адреси за ланцюжком блоків, оскільки в них також використовуються одноразові адреси.

Транзакція, підписана з використанням схеми кільцевого підпису, посилається на певну кількість транзакцій в ланцюжку блоків. При цьому немає вимоги, щоб транзакції були адресовані одному заданому відправнику. Таким чином, транзакція з однаковою ймовірністю може використовувати на вхід будь-яку з операцій, на які вона посилається і зі збільшенням кількості подібних посилань збільшується складність задачі встановлення істинного відправника. Такий метод анонімізації, однак, збільшує розмір підпису. Відправник обирає між рівнем комісії та рівнем анонімності.

Виходи транзакцій при цьому відправляються на адресу, яка використовується один раз та генерується зі справжньої адреси відправника з використанням випадкового ключа транзакції. Адресат генерує приватний ключ зі свого приватного ключа та випадкового ключа, що зазначений у тілі транзакції. Таким чином, кілька транзакції на одну адресу виглядають як випадкові транзакції на випадкові адреси.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ СХЕМИ КІЛЬЦЕВОГО ПІДПИСУ

3.1 Опис реалізації

Базуючись на результатах попередніх розділів, в ролі алгоритму для реалізації було обрано алгоритм зв’язних кільцевих підписів на основі схеми підпису Шнорра для еліптичних кривих, описаний у розділі 2.4.3. Такий вибір було зроблено тому, що дана схема лежить в основі багатьох сучасних схем кільцевого підпису, що використовуються у криптовалютах. Вона задовольняє основі потреби криптовалют, а більшість модифікацій, що застосовуються до неї, направлені на зменшення розмірів підписів, ключів та більш ощадливе використання пам’яті.

Для реалізації даного алгоритму було обрано мову програмування Python версії 3.7. Для розробки програми було обрано інтегроване середовище програмування PyCharm. Воно має повний набір необхідних інструментів для розробки та тестування програм, написаних мовою програмування Python. PyCharm має повну підтримку підсвітки коду, інтеграцію з системою контролю версій, підтримує різні сценарії запуску програм. Це середовище розробки доступне безкоштовно для некомерційної розробки студентам вищих навчальних закладів.

В якості еліптичної кривої програма використовує SECP256k1.

3.2 Структура програми

Програма складається з кількох основних модулів та частково побудована за принципами об’єкто-орієнтованого програмування. Структурно можна виділити такі модулі:

а) Point – клас, що представляє собою точку еліптичної кривої; у ньому визначена структура точки еліптичної кривої та операції, що застосовуються до неї: порівняння, додавання, множення;

б) Curve – клас, що представляє еліптичну криву; надає доступ до основних параметрів кривої, обчислює її генератор, порядок

генератора та довжину ключів та підписів, які можна генерувати з її використанням;

в) LSAG – файл, що містить інструменти для генерації та верифікації підпису, а також створення, експорту та імпорту приватних та публічних ключів та підписів.

Ключовими методами програми є $sign$ – метод, що генерує підпис, отримавши на вхід набір відкритих ключів учасників кільця, приватний ключ ініціатора підпису та повідомлення, та $verify$ – метод, що верифікує підпис стосовно набору публічних ключів учасників кільця та повідомлення.

3.3 Інтерфейс програми

Окрім використання реалізованого алгоритму в якості програмного модуля, який можна інтегрувати в інші програмні рішення, доступний інтерфейс командного рядка для користувача.

Після збірки та компіляції програми вона доступна для виклику через консоль. При виклику програми з аргументом –h буде надруковано допоміжне повідомлення з короткою інформацією про доступні параметри.

```
usage: ring_signature.py [-h]
                         {generate_keys,generate_signature,verify_signature,linked}
                         ...

positional arguments:
  {generate_keys,generate_signature,verify_signature,linked}
    generate_keys       generate new keys for ring signature
    generate_signature  generate signature for for message
    verify_signature    verify signature
    linked              check if signatures is linked

optional arguments:
  -h, --help            show this help message and exit
```

Рис.1 Інструкція для користувача

Програма складається з чотирьох модулів, кожен з яких відповідає за одну з чотирьох алгоритмічних частин схеми зв’язного кільцевого цифрового підпису:

генерацію ключів, генерацію підпису, верифікацію підпису та перевірку зв’язності повідомлення.

```
usage: ring_signature.py generate_keys [-h] -n N [-k [K]]

optional arguments:
  -h, --help  show this help message and exit
  -n N        number of participants in the ring
  -k [K]      path to save [default: ./data]
```

Рис.2 Модуль генерації ключів

Модуль генерації ключів створює приватні та публічні ключі для учасників кільця та зберігає усі публічні ключі у один файл, в приватний ключ потенційного підписувача – у окремий файл, а також фіксує час своєї роботи. Докладніше про аргументи:

а) -h – показати довідку по параметрам;

б) -n – кількість учасників кільця, для яких необхідно створити публічний ключ;

в) -k – шлях для збереження ключів, за замовчуванням ключі зберігаються в папку «data» в поточній директорії, при цьому приватні ключі зберігаються в окремі файли, які мають назви «secreti.txt», де *i* означає порядковий номер ключа в кільці, а публічні ключі записуються по порядку в один файл «publics.txt».

```
usage: ring_signature.py generate_signature [-h] -i I [-k [K]] [-l [L]] [-m M]
                                            [-s [S]]

optional arguments:
  -h, --help  show this help message and exit
  -i I        number of signer in the ring
  -k [K]      path to public keys [default: ./data/publics.txt]
  -l [L]      path to private key [default: ./data/secreti.txt]
  -m M        message
  -s [S]      path to save signature [default: ./data/signature.txt]
```

Рис.3 Модуль генерації підпису

Модуль генерації підпису створює підпис до повідомлення, асоційований з набором публічних ключів учасників кільця та приватним ключем одного з учасників кільця. Модуль генерації підпису також зберігає підпис у файл. Докладніше про аргументи:

а) -h – показати довідку по параметрам;

б) -i – номер приватного ключа підписувача у списку ключів;

в) -k – шлях для завантаження публічних ключів, за замовчуванням ключі шукаються в папці «data» в поточній директорії у файлі з назвою «publics.txt»;

г) -l – шлях для завантаження приватного ключа, за замовчуванням ключ шукається в папці «data» в поточній директорії у файлі з назвою «secreti.txt», де *i* – параметр, який позначає номер приватного ключа підписувача у кільці з пункту б);

д) -m – повідомлення, яке необхідно підписати;

е) -s – шлях для збереження підпису, за замовчуванням підпис зберігається в папку «data» в поточній директорії у файл «signature.txt».

```
usage: ring_signature.py verify_signature [-h] [-k [K]] [-m M] [-s [S]]

optional arguments:
  -h, --help  show this help message and exit
  -k [K]      path to public keys [default: ./data/publics.txt]
  -m M        message
  -s [S]      path to signature [default: ./data/signature.txt]
```

Рис.4 Модуль верифікації підпису

Модуль верифікації підпису перевіряє, чи є підпис дійсним підписом повідомлення стосовно набору публічних ключів. Докладніше про аргументи:

а) -h – показати довідку по параметрам;

б) -k – шлях для завантаження ключів, за замовчуванням ключі шукаються в папці «data» в поточній директорії у файлі з назвою «publics.txt»;

в) -m – повідомлення;

г) -s – шлях для завантаження підпису, за замовчуванням підпис зберігається в папці «data» в поточній директорії у файлі «signature.txt».

```
usage: ring_signature.py linked [-h] [-s [S]] [-d [D]]

optional arguments:
  -h, --help  show this help message and exit
  -s [S]      path to first signature [default: ./data/signature1.txt]
  -d [D]      path to second signature [default: ./data/signature2.txt]
```

Рис.4 Модуль перевірки зв’язності підписів

Модуль верифікації підпису перевіряє, чи є підписи згенерованими одним учасником кільця. Докладніше про аргументи:

а) -h – показати довідку по параметрам;

б) -s – шлях для завантаження першого підпису, за замовчуванням підпис шукається в папці «data» в поточній директорії у файлі «signature1.txt»;

в) -d – шлях для завантаження другого підпису, за замовчуванням підпис шукається в папці «data» в поточній директорії у файлі «signature2.txt».

У Додатку А наведений код методів генерації ключів, підписання повідомлення та перевірки підпису.

3.4 Робота програми

Розглянемо роботу усіх програмних модулів з стандартними параметрами. За умови існування файлів з ключами та підписами, вони можуть запускатись у довільному порядку.

Запуск та результати роботи модулів генерації ключів, генерації підпису, верифікації підпису відповідно представлені на рис.5, рис.6, рис.7. На рис.8 зображено результат роботи модуля перевірки зв’язності підпису для підписів до

двох різних повідомлень, створених одним і тим же користувачем одного і того ж кільця.

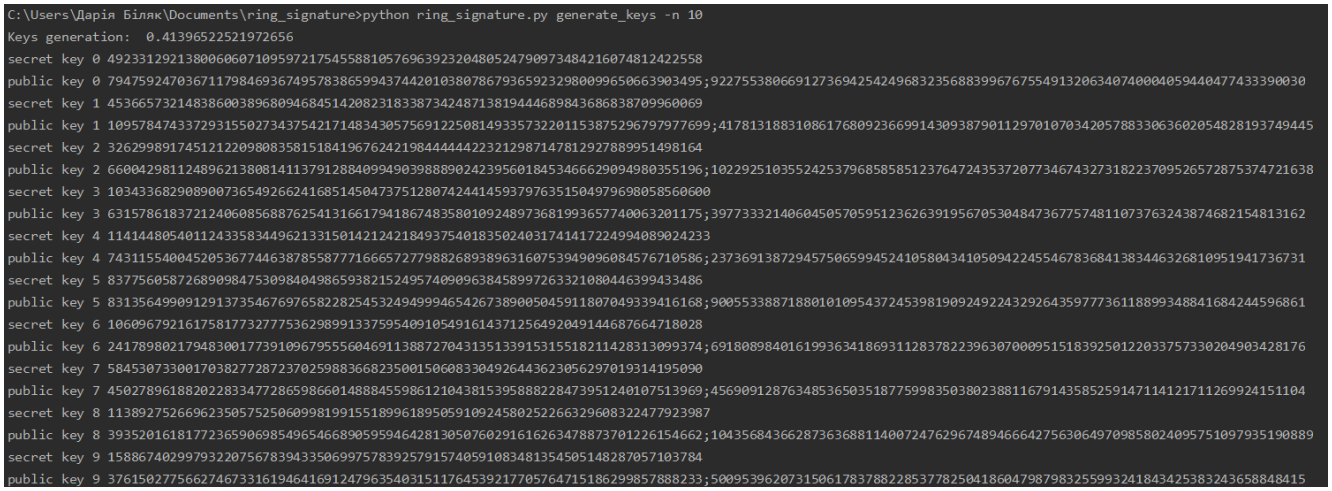

Рис.5 Робота модуля генерації ключів

Модуль генерації ключів генерує ключі, виводить результат генерації ключів у консоль та записує згенеровані ключі у відповідні файли. Окрім цього, модуль генерації ключів обчислює час, за який ключі були створені.

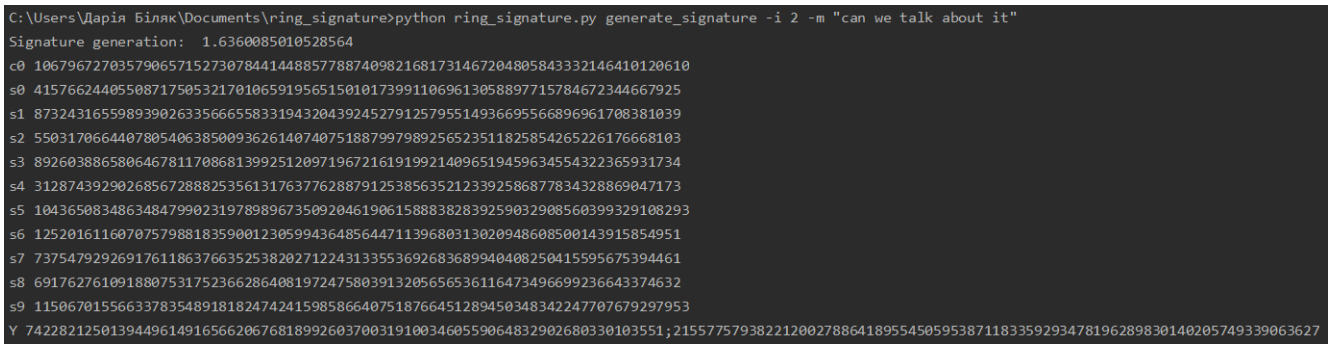

Рис.6 Робота модуля генерації підпису

Модуль генерації підпису створює підпис до заданого повідомлення стосовно заданого приватного ключа та кільця публічних ключів, а також обчислює власний час генерації підпису та виводить складові отриманого підпису на екран.

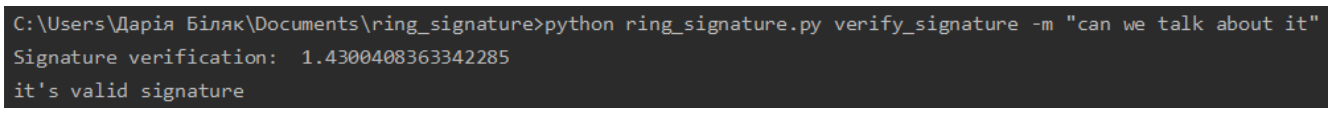

Рис.7 Робота модуля генерації підпису

Модуль верифікації підпису перевіряє, чи є заданий підпис до повідомлення дійсним, обчислює час, за який була проведена верифікація підпису, та виводить

на екран одне з двох повідомлень: «it`s valid signature» – якщо підпис дійсний та «it`s not valid signature» – в іншому випадку.

```
C:\Users\Дарія Біляк\Documents\ring_signature>python ring_signature.py linked
Signatures are linked
```

Рис.7 Робота модуля перевірки зв’язності підпису

Модуль перевірки зв’язності підпису перевіряє, чи є підписи зв’язними та виводить на екран одне з двох повідомлень: «Signatures are linked» – якщо підписи зв’язні та «Signatures are not linked» – якщо підписи не зв’язні.

3.5 Підсумки реалізації

В ході написання кваліфікаційної роботи бакалавра було реалізовано повноцінну та готову до використання схему зв’язного цифрового кільцевого підпису. Вона може використовуватись для дослідження та тестування подальших вдосконалень та реалізує інструменти для використання усіх її можливостей. З повною реалізацією програмного рішення можна ознайомитись за посиланням на гітхаб [25].

ВИСНОВКИ

У даній кваліфікаційній роботі бакалавра було проведено дослідження та аналіз цифрових підписів та реалізовано схему зв’язного кільцевого цифрового підпису на основі алгоритму цифрового підпису Шнорра.

а) Проведено дослідження та аналіз безпеки схем цифрового підпису, їх будови, розглянуто проблеми схем цифрового підпису та можливі варіанти їх вирішення, в тому числі способи розширення обмежених по довжині та обмежених за кількістю використань схем цифрового підпису до загальних. Описана атака з вибором повідомлення та основні вимоги до безпеки схем цифрового підпису, а також основні властивості та види кільцевих схем цифрового підпису.

б) Розглянуто найбільш розповсюджені алгоритми схем цифрового підпису, такі, як схема RSA, схема Рабіна, одноразова схема Лампорта, схеми Ель-Гамаля, DSA, ECDSA, BLS, а також схема цифрового підпису Шнорра та похідна від неї схема кільцевого підпису LSAG. Описані переваги та недоліки наведених алгоритмів, можливі сфери їх застосування.

в) Виконано реалізацію зв’язної схеми кільцевого підпису на основі схеми цифрового підпису Шнорра LSAG, яка широко застосовується у анонімних криптовалютах, які майже унеможливлюють встановлення особистості користувача криптовалюти за історією його транзакцій. Реалізація написана мовою програмування Python, яка є досить зручною для написання та використання коду та дозволяє оцінити основні параметри швидкості роботи алгоритму, хоча і не є швидкою мовою. Проведено аналіз програми та встановлено, що вона виконує поставлені перед нею задачі та повністю реалізує властивості схеми цифрового підпису LSAG.

Алгоритми цифрового підпису є дуже актуальними у наш час, оскільки навіть на побутовому рівне усе більш гостро стоїть питання про безпеку передачі даних. Схеми, що були створені раніше, стають менш актуальними зараз, оскільки

ростуть вимоги до безпеки та швидкості роботи алгоритмів, а також розширюється сфера застосування цифрових підписів.

Особливий потенціал мають схеми кільцевого підпису, оскільки вони дозволяють реалізувати цілковиту анонімність в сучасних криптовалютах, а також мають перспективи для застосування у процедурах голосування. На мою думку, схеми кільцевого цифрового підпису на основі схеми підпису Шнорра в майбутньому стануть ще більш застосованими, оскільки вони є досить швидкими у варіанті з еліптичними кривими та не вимагають значних обчислюваних витрат.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is Digital Signature – How it works, Benefits, Objectives, Concept [Електронний ресурс] – Режим доступу до ресурсу: https://www.hubspire.com/resources/general/what-are-digital-signatures-how-they-work/.

2. Goldreich O. Foundations of Cryptography II Basic Applications / Oded Goldreich., 2004. – 798 с.

3. Katz J. Introduction to Modern Cryptography / J. Katz, L. Yehuda., 2007.

4. Katz J. Digital Signatures / Katz., 2010.

5. R. L. Rivest, A. Shamir, Y. Tauman. How to Leak a Secret, 2001.

6. R. L. Rivest, A. Shamir, Y. Tauman. How to Leak a Secret: Theory and Applications of Ring Signatures, 2006.

7. Menezes A. Handbook of Applied Cryptography / A. Menezes, P. van Oorschot, S. Vanstone., 1996.

8. Lamport L. Constructing Digital Signatures from a One Way Function, 1979.

9. R. L. Rivest, A. Shamir, L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, 1977.

10. C. Dwork and M. Naor. An efficient existentially unforgeable signature scheme and its applications. Journal of Cryptology, 11(3):187–208, 1998.

11. R. Cramer and I. Damg°ard. New generation of secure and practical RSA-based signatures. In Advances in Cryptology – Crypto ’96, volume 1109 of LNCS, pages 173–185. Springer, 1996.

12. S. Hohenberger and B. Waters. Short and stateless signatures from the RSA assumption. In Advances in Cryptology – Crypto 2009, volume 5677 of LNCS, pages 654–670. Springer, 2009.

13. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. ACM Transactions on Information and System Security, 3(3):161–185, 2000.

14. M. Fischlin. The Cramer-Shoup strong-RSA signature scheme revisited. In 6th Intl. Workshop on Theory and Practice in Public Key Cryptography(PKC 2003), volume 2567 of LNCS, pages 116–129. Springer, 2003.

15. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In Advances in Cryptology – Eurocrypt ’99, volume 1592 of LNCS, pages 123–139. Springer, 1999.

16. Rabin M. O. Digitalized Signatures and Public Key Functions as Intractable as Factorization, 1979.

17. ElGamal T. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, 1985.

18. Digital Signature Standard (DSS). Federal Information Processing Standards publication #186-2. U. S. Department of Commerce, National Institute of Standards and Technology, 2000.

19. Schnorr C.P. Efficient Signature Generation by Smart Cards. – J. Cryptology, 1991. – C. 161 – 174.

20. S. Vanstone, “Responses to NIST’s Proposal”, Communications of the ACM, 35, July 1992, 50-52.

21. Boneh D. Short Signatures from the Weil Pairing, 2004.

22. M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In Proc. ASIACRYPT 2002, pages 415–432. Springer-Verlag, 2002. LNCS Vol. 2501.

23. Liu J. K. Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups / J. K. Liu, V. K. Wei, D. S. Wong., 2004.

24. Bassam El Khoury Seguias. Monero’s Building Blocks Part 6 of 10 – Linkable Spontaneous Anonymous Group (LSAG) signature scheme.

25. Реалізація алгоритму зв’язного кільцевого цифрового підпису на еліптичних кривих на основі алгоритму цифрового підпису Шнорра https://github.com/avigeris/ring_signatures

ДОДАТОК А

```
def generate_keys(number_participants):
    start = time.time()
    x = [randrange(curve.order()) for i in range(number_participants)]
    y = list(map(lambda xi: curve.generator() * xi, x))
    export_private_keys(x)
    end = time.time()
    print("Keys generation: ", end - start)
    return x, y
```

Рис.1 Метод генерації ключів

```
def sign(siging_key, key_idx, M, y, G=curve.generator(), hash_func=hashlib.sha3_256):
    start = time.time()
    n = len(y)
    c = [0] * n
    s = [0] * n
    h = H2(y, hash_func=hash_func)
    Y = h * siging_key
    u = randrange(curve.order())
    c[(key_idx + 1) % n] = H([y, Y, M, G * u, h * u], hash_func=hash_func)
    for i in [i for i in range(key_idx + 1, n)] + [i for i in range(key_idx)]:
        s[i] = randrange(curve.order())
        z_1 = (G * s[i]) + (y[i] * c[i])
        z_2 = (h * s[i]) + (Y * c[i])
        c[(i + 1) % n] = H([y, Y, M, z_1, z_2], hash_func=hash_func)
    s[key_idx] = (u - siging_key * c[key_idx]) % curve.order()
    end = time.time()
    print("Signature generation: ", end - start)
    return (c[0], s, Y)
```

Рис.2 Метод підписання повідомлення

```python
def verify(message, y, c_0, s, Y, G=curve.generator(), hash_func=hashlib.sha3_256):
    start = time.time()
    n = len(y)
    c = [c_0] + [0] * (n - 1)
    h = H2(y, hash_func=hash_func)
    for i in range(n):
        z_1 = (G * s[i]) + (y[i] * c[i])
        z_2 = (h * s[i]) + (Y * c[i])
        if i < n - 1:
            c[i + 1] = H([y, Y, message, z_1, z_2], hash_func=hash_func)
        else:
            end = time.time()
            print("Signature verification: ", end - start)
            return c_0 == H([y, Y, message, z_1, z_2], hash_func=hash_func)
    end = time.time()
    print("Signature verification: ", end - start)
    return False
```

Рис.3 Метод верифікації підпису