КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики
Кафедра математичної інформатики

Кваліфікаційна робота

на здобуття ступеню бакалавру за
освітньо-професійною програмою
"Інформатика" спеціальності 122 Комп'ютерні
науки на тему:

Огляд алгоритмів ідентифікації типу «свій-чужий»

Виконав студент 4-го курсу
П'ЯТИГОРСЬКИЙ Нікіта Дмитрович ____ __
(підпис)

Науковий керівник: професор
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

«____» _____ 2023 р.,
протокол № ___

Завідувач кафедри професор, доктор фіз.-мат. наук

ТЕРЕЩЕНКО В. М _____

РЕФЕРАТ

Об’єктом дослідження і розробки є протоколи ідентифікації з нульовим розголошенням та порівняння їх продуктивності.

Метою даної роботи є визначення найбільш продуктивного алгоритму з точки зору малопотужних систем.

Методом дослідження є вимірювання і порівняння часу роботи та використаної пам’яті програмних реалізацій основних протоколів ідентифікації, таких як протокол Шнорра, Фіата-Шаміра. Гіллоу — Куіскуотера, GPS, ECDSA та ШАЗ+. Інструментами дослідження є компілятор мови С, стандартна бібліотека мови С, бібліотека довгої арифметики GNUMP, та середа розробки і вимірювання на базі x86_64 PC.

Результати роботи: в результаті вдалося оцінити, що розглянутий протокол ШАЗ+, має найкращий показник часу автентифікації загалом - тобто розглядаючи одночасно тривалість часу підпису та верифікації, також протокол має найменшу кількість використаної пам’яті серед інших розглянутих алгоритмів. Додатковим результатом є створені реалізації розглянутих протоколів.

Результати даної роботи можуть бути використані як для здійснення більш ширшої оцінки продуктивності протоколу ШАЗ+ в порівнянні з іншими протоколами, так і використання розробленого коду для розробки застосунків для малопотужних систем.

Взаємозв’язок з іншими роботами: ця робота спирається на розробку професора Анісімова Анатолія Васильовича, а саме протоколу автентифікації ШАЗ+ [4].

Як напрям подальшого дослідження можна виділити необхідність точної оцінки витрат енергії алгоритму, також порівняння з більш широким колом протоколів заснованих на геш-функціях та оцінкою при цьому окремих реалізацій ША3+ для декількох різних геш-функцій. Іншим можливим напрямком є апаратна реалізація протоколу з подальшим аналізом її властивостей.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

A - стверджуючий. Особа, що доводить свою ідентичність і(або) передає зашифроване чи підписане **A** повідомлення.

B - перевіряючий. Особа, що проводить перевірку ідентичності іншої особи(як правило A, але не виключаючи інших учасників), або приймає зашифроване чи підписане повідомлення.

T – довірена особистість або контролер коаліційної групи. Більш детальне визначення контролеру та такої групи можна знайти в роботі [4].

H - одностороння криптографічно-стійка геш-функція. В практичній частини цієї роботі в якості **H** використовується алгоритм гешування sha256.

ВСТУП

Протоколи ідентифікації мають фундаментальне значення для створення безпечних і надійних механізмів автентифікації в різних сферах. Зі стрімким зростанням цифрових транзакцій та онлайн-взаємодій протоколи ідентифікації відіграють ключову роль у захисті конфіденційної інформації та запобіганні несанкціонованому доступу до неї. Сфери застосування протоколів ідентифікації різноманітні - від фінансових систем і охорони здоров'я до державних послуг, блокчейну і біометричної автентифікації. Поточні дослідження зосереджені на підвищенні ефективності, масштабованості та конфіденційності протоколів ідентифікації, щоб відповідати викликам і вимогам різних сфер застосування, що постійно змінюються. Дослідники також розглядають вплив протоколів ідентифікації на продуктивність недорогих пристроїв, включаючи Інтернет речей (IoT), де пристрої з обмеженими ресурсами потребують легких і ефективних рішень для автентифікації. Постійно вдосконалюючи ці протоколи, науковці прагнуть розробити ефективні та безпечні ідентифікаційні рішення для широкого спектру реальних сценаріїв.

Підставою для виконання цієї роботи є актуальність пошуку оптимальних способів ідентифікації для малопотужних пристроїв, що забезпечують швидке обчислення, загальні низькі вимоги до оперативної пам’яті пристрою та розміру повідомлень на інтерактивних кроках, а також залишають можливість апаратнох реалізації. З іншої сторони для таких протоколів зберігаються сильні вимоги щодо їх ьезпеки, оскільки для них актуальні всі типи атак, що і для потужних систем.

Метою є визначення найбільш ефективного, з точки зору використання часу та ресурсів алгоритму ідентифікації, який при цьому задовольняє основні безпекові вимоги, як наприклад, нульове розголошення та стійкість до атак типу “людина посередині” та атак з повторенням(replay attacks). Для цього було

розглянуто низку відомих стійких протоколів ідентифікації з нульовим розголошенням, які при цьому визнані застосовними для малопотужних систем.

Методом дослідження є практична реалізація всіх запропонованих до розгляду протоколів, з подальшим виміром характеристик їх роботи. Можливим застосуванням, як результатів оцінок так і програмних реалізації є розробки у сфері IOT, вбудованих пристроїв, малопотужних чіпів для ідентифікації, а також подальші дослідження на цю тему.

Ця робота спирається на розробку професора Анісімова Анатолія Васильовича, а саме - новий протокол автентифікації типу “свій-чужий” ША3+ [4], та ставить за основну задачу порівняння його з іншими протоколами ідентифікації в межах, визначених метою роботи.

РОЗДІЛ 1. ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Поняття ідентифікації та аутентифікації

Ідентифікація це процес декларування особистості або автентичності учасника. Він передбачає обмін доказами або підтвердженням знань між тим, хто підтверджує, і тим, хто перевіряє, для встановлення достовірності заявленої ідентичності. Стверджуючий представляє докази, щоб переконати верифікатора, який застосовує заздалегідь визначені перевірки або правила для оцінки достовірності доказів.

Автентифікація - це процес перевірки заявленої особи або легітимності учасника. Вона передбачає перевірку повноважень або доказів, наданих учасником для встановлення його автентичності. Основна мета автентифікації - переконатися, що заявлена особа відповідає дійсній, запобігти несанкціонованому доступу або видачі себе за іншу особу.

На відміну від автентифікації, ідентифікація відноситься до більш широкого процесу демонстрації ідентичності або автентичності учасника в рамках виконання протоколу. Він охоплює представлення переконливих доказів або доказів знань для встановлення достовірності заявленої особи. Протоколи ідентифікації зазвичай передбачають інтерактивні кроки між тим, хто підтверджує, і тим, хто перевіряє, де той, хто підтверджує, надає докази, а той, хто перевіряє, перевіряє їхню достовірність.

Щоб проілюструвати різницю між автентифікацією та ідентифікацією, розглянемо два поширені протоколи:

- Автентифікація на основі пароля: У цьому протоколі користувач надає ім'я користувача та пароль для підтвердження своєї особи. Сервер перевіряє надані облікові дані за збереженими записами і надає доступ,

якщо автентифікація пройшла успішно. Тут основна увага приділяється перевірці заявленої ідентичності (автентифікації) шляхом перевірки правильності пароля.

- Протокол Zero-Knowledge Proof (ZKP): ZKP є прикладом протоколу ідентифікації. Він дозволяє підтверджуючому продемонструвати знання секрету без розкриття самого секрету. У цьому протоколі підтверджуючий взаємодіє з верифікатором за допомогою серії інтерактивних кроків, надаючи докази, щоб переконати верифікатора у своєму знанні секрету. Мета полягає у встановленні особи або автентичності стверджуючого без розкриття конфіденційної інформації.

Поняття "Ідентичність", застосоване раніше в цьому розділі можна розуміти як унікальне представлення або характеристику, що відрізняє фізичну або юридичну особу від інших. У контексті протоколів ідентифікації ідентифікатор слугує засобом встановлення довіри та гарантування того, що учасники можуть достовірно пов'язувати дії та інформацію з конкретними суб'єктами. На практиці представлення ідентичності в розглянутих далі протоколах має вигляд числа, побудованого на основі певної приватної або публічної інформації особи **A**, як наприклад повне ім’я, пошта, номер телефону.

Протоколи ідентифікації ретельно розробляються для того, щоб учасники могли довіряти ідентифікаційним даним, представленим під час виконання протоколу, одночасно забезпечуючи конфіденційність і безпеку. Завдяки використанню криптографічних методів і математичних конструкцій, ці протоколи є стійкими до різних типів атак, включаючи атаки з відтворенням(replay attacks), атаки з видаванням себе за інших осіб і атаки "людини посередині".

1.2 Sigma-протокол

Одним з прикладів протоколів ідентифікації є Σ-протоколи. Це сімейство протоколів доказу з нульовим розголошенням, для яких доведено такі основні властивості, як досконала повнота, достовірність знань, та нульове розголошення за умови чесного верифікатора(HVZK - honest verifier zero knowledge). Достовірність знань - це гарантія того, що якщо доказувач нечесний і не має необхідних знань, то верифікатор з високою ймовірністю відхилить його твердження. Досконала повнота означає гарантію того, що чесний доказувач, який володіє необхідними знаннями, завжди переконає верифікатора в істинності заявленого твердження.

Хід Σ-протоколу представляється трьома раундами:

1. Внесок(commitment) - особа **A** передає **B** випадкове значення **n**
2. **B** відповідає **A** випадковим випробуванням c.
3. **A** робчислює випробування на основі його приватного числа, та чисел **c**, **n** та передає його **B.** На цьому раунді, на основі отриманого розв’язку **B** приймає рішення, відхилити або прийняти правдивість особистості **A**.

Більшість протоколів, у цій роботі будуть представлені у загальному вигляді Σ-протоколу, тобто є трьох раундовими та мають доведені основні властивості Σ-протоколу.

Детальне визначення Σ-протоколу та доказ основгих його властивостей наведено в роботі [8].

РОЗДІЛ 2. КЛАСИЧНІ ПРОТОКОЛИ ІДЕНТИФІКАЦІЇ

2.1 Протокол ідентифікації Шнорра

Протокол підпису та ідентифікації Шнорра, запропонований у робот [1], є найбільш цітуємим та досліджуваним Σ-протоколом. Він використовує проблему дискретного логарифму в підгрупі порядку q за модулем p. яка є обчислювально не розв'язною за великих p та q.

Опишемо основні доменні параметри протоколу, що задають властивості його підгрупи:

- **p** - модуль групи
- **q** - порядок групи
- **g** - генератор підгрупи Z_p^* порядку **q**.

Протокол 2.1.1

Процедура 2.1.1.1 - ініціалізація доменних параметрів особою **T**:

1. Оберемо прості числа **p** та **q**: **p - 1** mod **q** = 0.
2. Оберемо g: $g^q = 1 \, mod \, p, \, g \asymp 1$
   $T \to A, B$: $g, \; q, \; p$

Процедура 2.1.1.2 - генерація пари ключів особи **A**:

1. Генеруємо число **s** як випадкове за модулем q, s > 0.
2. Обчислимо $v = g^{-s} mod \, p$

Після отримання ключової пари, довірена особа T має підписати (I,v), де I - публічний ідентифікатор особи A(див. теоретичні відомості). Надалі підписана пара (I, v) надається всім, хто зацікавлений в перевірці ідентичності I за публічним ключем v.

Процедура 2.1.1.3 - обмін повідомленнями

1. Внесок: А обирає випадкове r за модулем q, r > 1. Та обчислює $x = g^r mod\ p$. $A \rightarrow B: x$
2. В обирає випробування $e \in \{0,..., 2^t - 1\}$, $B \rightarrow A: e$
3. А обчислює розв'язок $y = r + se\ mod\ q$, $A \rightarrow B: y$

Отримавши y, В здійснює перевірку $x = g^y v^e \bmod p$. Якщо рівність виконується приймається рішення ПРИЙНЯТИ інакше ВІДХИЛИТИ. Дійсно, легко показати властивість повноти протоколу: $g^y v^e = g^{r+se} g^{-se}$; $g^r g^{se-se} = g^r = x$

Таким чином особа А надала доказ володіння приватним ключем s, пов'язаним з $s \in (I, v)$ співвідношенням 3.1.1.2 (2).

2.2 Протокол ідентифікації Фейге — Фіата — Шаміра

Протокол ідентифікації Фейге-Фіата-Шаміра [2], є ще одним прикладом схеми з нульовим розголошенням. Протокол ґрунтується на складності розв'язання задач на квадратичні залишки в модульній арифметиці, зокрема, на квадратичні лишки за модулем складеного числа та є одним з найперших запропонованих протоколів, що вирішує проблему ідентифікації з нульовим розголошенням.

У протоколі Фейге-Фіата-Шаміра, стверджувач **A** спочатку генерує внесок на основі квадрату випадкового числа за модулем. Верифікатор, в свою чергу, генерує випадкові виклики, на які особа **A** відповідає, надаючи добуток обраних елементів його приватного ключа в залежності від секретного внеску та фінально маскує їх додатково домноживши на випадковий внесок R. Фінально верифікатор виконує множення квадрату розв'язку А на публічний ключ А, і в результаті залишається публічна частина внеску X, у випадку валідної автентифікації.

Безпека протоколу Фейге-Фіата-Шаміра ґрунтується на припущенні про обчислювальну неможливість ефективного розв'язання проблеми квадратичних лишків. Протокол є відносно обчислювально простий, і на відміну від Шнорра не потребує експоненціацій у велику степінь, в чому ми переконуємося на практиці в експериментальній частині роботи.

Домені параметри:

- прості числа p та q, належать довіреному центру **T**, та не розповсюджуються
- **n** - публічний модуль, що обчислюється та розповсюджується довіреним центром **T**
- **k** - додатковий публічний параметр безпеки
- **t** - додатковий публічний параметр безпеки

Протокол 2.2.1

Процедура 2.2.1.1 - ініціалізація доменних параметрів особою **T**:

1. Оберемо два великі прості числа **p** та **q**
2. Обчислимо n = pq
3. Обчислимо додаткові параметри k = [log(log(n)] + 1, t = [log(n)] + 1

$T \rightarrow A, B: n,\ k,\ t$

Процедура 2.2.1.2: - генерація пари ключів особи **A**:

1. Оберемо **k** випадкових чисел $S_1 \ldots S_k$ як випадкові за модулем n
2. Обчислимо **k** значень I, де $\pm S^{-2} mod\, n$, де знак обирається випадково для кожного I

$A \to B: I_{1..k}$

Процедура 2.2.1.2 - обмін повідомленнями

Кроки 1-3 виконуються **t** разів

1. А обирає R випадковим чином та обчислює $X = \pm R^2 mod\ n$, $A \to B: X$
2. В обирає випадкову бітову строку E довжини k біт, $B \to A: E$
3. А обчислює $Y = R \cdot \prod_{Ej=1} S_j mod\ n$, $A \to B: Y$

B здійснює перевірку, що $X = Y^2 \cdot \prod_{Ej=1} I_j mod\ n$, або $X = -Y^2 \cdot \prod_{Ej=1} I_j mod\ n$. Якщо одна з рівностей виконується, приймається рішення ПРИЙНЯТИ інакше ВІДХИЛИТИ.

Покажемо доведення повноти протоколу 3.2.1

$$Y^2 \cdot \prod_{E_j=1} I_j = (R \cdot \prod_{E_j=1} S_j)^2 \cdot \prod_{E_j=1} I_j = R^2 \cdot \prod_{E_j=1} (S^2 I_j) = \pm R^2 = \pm X$$

2.3 Протокол ідентифікації Гіллоу — Куіскуотера

Протокол ідентифікації Гіллоу-Куіскуотера [1], що був опублікований через деякий час після протоколу Фіата-Шаміра та незадовго до ідентифікації Шнорра побудований беручи за основу криптосистему RSA: стверджувач доводить, що знає корінь порядку **e** за модулем **n.** Таким чином, крок ініціалізації цього протоколу є дуже схожим з процедурою генерації ключів RSA. На кроці ініціалізації покажемо, як отримати публічну трійку **(n, e, y)** та

закриту основу екпоненціації **x**, що визначаэ обобистість **A**. Однією з зручних переваг протоколу над попередниками є відсутність потреби повтору раундів автентифікації **t** разів, що значно зменшує загальний розмір даних, необхідних для передачі.

Нагадаємо про додаткове позначення:

λ(n) - функція Кармайкла. Використовуючи, що n = pq можемо обчислити λ(n) = НСК(λ(p), λ(q)). Оскільки **p** та **q** прості, то використовуючи теорему Кармайкла це можна виразити як НСК(φ(p), φ(q)) = НСК(p-1, q-1), де **φ** - функція Ейлера.

Протокол 2.3.1

Процедура 2.3.1.1 - ініціалізація доменних параметрів довіреним центром **T**:

1. Оберемо q, p, обчислимо n
2. Обчислимо λ(n) = lcm(p − 1, q − 1)
3. Оберемо e: 2 < e < λ(n), НСД(e, λ(n)) = 1

$T \rightarrow A, B: n, e$

Процедура 2.3.1.2: - генерація пара ключів особи **A**:

1. Оберемо секретне випадкове значення x з Z_n^*
2. Обчислимо $y = x^e mod\ n$

Примітка: зазвичай e можливо обрати наперед як невелике(відносно n) просте число і не виконувати кроки 1-3. В подальшому, при реалізації приймемо e = 65537.

Процедура 2.3.1.3 - обмін повідомленнями:

1. A обирає r, як випадкове з Z_n^* та обчислює $a = r^e mod\, n$, $A \to B$: a
2. B обирає c, як випадкове з Z_n^*, $B \to A$: c
3. A обчислює $z = rx^c$, $A \to B$: z

   B здійснює перевірку, що $z^e = ay^c mod\, n$. Якщо рівність виконується приймається рішення ПРИЙНЯТИ інакше ВІДХИЛИТИ.

Покажемо доведення повноти алгоритму 3.3.1:

$$z^e = (rx^c)^e = r^e(x^e)^c = ay^c mod\, n$$

2.4 Протокол ідентифікації Girault Poupard Stern

Після того, як протокол ідентифікації Шнорра був запропонований у 1989 році, багато робіт було виконанопо його модифікації, доведенню основних властивостей та обґрунтування вибору безпечних параметрів. Однією з найвідоміших таких розробок стосовно покращення продуктивності алгоритму для малопотужних систем є схема GPS [7], описана в 1991 році Марком Жиро(Girault) та перевірена на коректність пізніше, у 1997 році, Гійомом Пупардом та Жаком Стерном.

Запропонована вченими схема є дуже схожою до схеми Шнорра, за виключенням того, що на кроці **3.1.1.3** (3) здійснюється множення без редукції за модулем, що значно спрощує обчислення на боці особи **A**. Також пізніше була запропонована ще одна модифікація протоколу з пропозицією обчислити внесок заздалегідь, у якості набору одноразових купонів, оскільки обчислення на першому кроці потребує піднесення в велику ступінь на боці **A**, що є важкою операцією для малопотужної системи. Альтернатио, купони A може надавати довірений центр T, користуючись захищеним каналом. В цій роботі ми

розглядаємо протокол GPS, оскільки він є особливо актуальним для малопотужний систем і основною його властивістю є саме перевага в швидкості над іншими Σ- протоколами, і оцінка цього є одним з об’єктів дослідження.

РОЗДІЛ 3. ВИКОРИСТАННЯ СХЕМ ПІДПИСУ ДЛЯ ІДЕНТИФІКАЦІЇ

У цьому розділі ми розглянемо алгоритм цифрового підпису над еліптичними кривими (ECDSA) та його застосування в контексті задач ідентифікації. Покажемо відомий спосіб того, як схема ECDSA може бути використана як будівельний блок для побудови протоколів ідентифікації у загальній формі сигма-протоколів.

3.1 Метод побудови протоколів ідентифікації на базі схем підпису

Для ідентифікації можуть бути використані інші відомі схеми підпису. Для цього вони мають бути приведені до загального Σ-протоколу.

Припустимо $S_A(m)$ представляє підпис особою A тексту m; certA - публічний ключ A, за яким можна перевірити валідність сигнатури. Тоді спираючись на цей механізм підпису, можна навести протокол ідентифікації особою B особи A:

Протокол 3.1.1

1. A обирає випадкове число r_A
2. B обирає випробування r_B, $B \to A$: r_B, B
3. A виконує $S_A(r_A, r_B, B)$, $A \to B$: $S_A(r_A, r_B, B)$, r_A

B перевіряє:

- підпис $S_A(r_A, r_B, B)$ є валідним за certA
- r_Bотримане на кроці (1) відповідає отриманому на кроці (3)
- перевіряє що текст **B** дійсно представляє його (тобто особи **B**) ідентичність

Якщо всі три умови виконуються приймається рішення ПРИЙНЯТИ інакше ВІДХИЛИТИ.

3.2 Перетворення схем ідентифікації в схеми підпису

Варто відмітити що зворотне перетворення з схеми Σ-ідентифікації в схему підпису теж можливе та добре вивчена. Здійснюється допомогою використання випадкового оракула з заданими розмірами вхідних та вихідних параметрів (на практиці може бути використана геш-функція). Такий метод був вперше запропонований Фіатом-Шаміром в їх роботі [2] і названий відповідно трансформацією Фіата-Шаміра.

Отже, ми можемо зробити висновок що будь яка схема підпису з може бути конвертована в протокол ідентифікації і навпаки, зі збереженням безпекових властивостей та за умови, що відповідні початкові схеми є безпечними.

3.3 Алгоритм Elliptic Curve Digital Signature Algorithm

Алгоритм DSA, що описується в стандарті DSS був розроблений Національним інститутом стандартів і технологій (NIST) у США в 1991 році і опублікований як Федеральний стандарт обробки інформації США (FIPS). DSA є варіантом алгоритму підпису Шнорра, і відповідно, як і схема Шнорра, використовує проблему дискретного алгоритму як обчислювально важку задачу.

Алгоритм ECDSA [6] зі стандарту ECC, представляє собою варіант базового DSA, що спирається на криптографію над еліптичними кривими, а саме добуток точок кривої, та неможливість обчислення зворотної операції за значенням добутку(тобто точкою) та тільки однією з точок множників.

Еліптична крива є множиною точок $P = (x_p, y_p)$, що задовольняють певне рівняння а також точка не нескінченності **O**. Наприклад рівняння визначене стандартом для поля цілих чисел F_p має вигляд:

$$y_p^2 = x_p^3 + ax_p + b$$

Зазначимо, що можливе використання і інших кривих, що проте вимагає додаткового розгляду їх властивостей та може вплинути на безпечність схеми.

Для точок такої кривої задаються три основні операції: додавання, піднесення до квадрату та різниця. Множення обчислюється з використанням додавання, та більш ефективних методів, комбінуючи додавання з піднесенням до квадрату. Точка на нескінченності **O** має особливу властивість при додаванні:

- O + P = P
- O + O = O

,де P будь яка точка. що належить кривій. Тепер, розглянувши загальні теоретичні відомості, щодо еліптичних кривих, ми можемо навести опис алгоритму цифрового підпису ECDSA.

Домені параметри:

P - модуль, за яким проводиться обчислення

N - порядок групи, іншими словами $N * G = O$

A, B - коефіцієнти рівняння кривої

G - початкова точка еліптичної кривої, що визначається координатами G_x та G_y

Протокол 3.2.1

Процедура 3.2.1.1 - генерація ключової пари особи A

1. Обирається випадкове ціле ціле d, d > 0, d < N
2. Обчислюється скалярний добуток Gd = Q

   Q - публічний ключ, d - приватний

Зворотне обчислання d за відомим Q, тобто логарифм в адитивній групі, є обчислювально нерозв'язною проблемою для великих N.

Процедура 3.2.1.1 - підпис повідомлення e:

1. Оберемо випадкове число k з інтервалу [0, N-1].
2. Обчислимо точку на кривій $(x_1, y_1) = k \times G$
3. Обрахуємо $r = x_1 mod\ n$, якщо r == 1, перейдемо на крок 1
4. Обрахуємо $s = k^{-1}(z + rd_A)$ mod n

   Сигнатура є парою (r, s)

Процедура 3.2.1.1 - верифікація підпису:

1. Виконаємо обчислення $u_1 = es^{-1}\ u_2 = rs^{-2}$
2. Обчислимо координати точки $(x_1, y_1) = u_1 G + u_2 Q_A$, за умови (x_1, y_1) є точкою **O** сигнатура відхиляється
3. Якщо $r = x_1 mod\ n$, то приймемо сигнатуру, інакше відхилимо

Наведену схему підпису ECDSA можна легко використати для ідентифікації за протоколом 4.1.1. Більш детальний опис алгоритму підпису та можливих його варіантів, як і рекомендацій щодо вибору параметрів можна знайти у стандарті [6].

РОЗДІЛ 4. ПРОТОКОЛ ШВИДКА АВТЕНТИФІКАЦІЯ 3+

Іншим способом побудови схем автентифікації, дещо відмінних від загального вигляду та властивостей Σ-протоколів, що були розглянуті є використання швидких, засновані на геш функціях, що є стійкими до колізій, схем підпису. В якості прикладу таких протоколів, можна навести одноразові підписи однобітних повідомлень з використанням геш-функцій - як найпростіший варіант, схеми Леслі Лемпорта - як більш досконалі та багаторазові. Пізніше, Лемпортом була розроблена та опублікована багаторазова схема автентифікації, що спирається на геш-ланцюги.

Визначення 1. Нехая $H: X \to X$ є функцією. Для додатного цілого числа j, приймемо $H^j(x)$ як позначення j-тої ітерації функції H: $H^j(x) = H(H(\dots H(x)\dots))$, де застосування H повторюється j разів.

В цьому розділі розглянемо нову схему автентифікації ША3+ типу свій-чужий, що була запропонована Анісімовим Анатолієм Васильовичем в його роботі [4]. Вона спирається на використання швидких геш-функцій, має структуру, подібну до автентифікації Лемпорта, але на відміну від останньої задовольняє більш сильні безпекові властивості, в тому числі нульове розголошення. Протокол ША3+ дозволяє здійснити автентифікацію(та взаємну автентифікацію), за умови що особи знаходяться в одній коаліційній групі, іншими словами особа в середині групи розпізнається іншою особою з групи як “свій”, інакше - “чужий”.

Визначення 2. Коаліційна група – це угруповання користувачів, що об’єднанні довірчими відносинами та взаємодіють між собою в кібер-просторі, що наражуваний до зловмисних атак.

Також, наведемо окремі вимоги та властивості, що має задовольняти розглянутий в цьому розділі протокол:

1. Автентифікація виконується за два швидких комунікаційних раундів «запит-відповідь».
2. Сторона **B** нічого не знає і не може обчислити таємні ідентифікаційні параметри (ключи) **A** і навпаки.
3. Система повинна бути стійкою до атак типу «людина посередині».
4. A може бути обчислювально відносно «слабким» в порівнянні з **B**.
5. Тільки авторизований перевіряючий **B** може автентифікувати **A**. Спосіб авторизації **B** визначає контролер коаліції **T**.
6. При взаємній автентифікації (B автентифікує A, при цьому A автентифікує B) вимоги симетричні при заміні A на B і B на A.

Повне обгрунтування виконуваності цих властивостей, а також більш повне визначення поняття коаліційної групи та її учасників можна знайти в роботі [4].
Опишемо протокол 5.1 для випадку двох учасників та контролеру:
Нехай **n** - кількість учасників коаліції. Наприклад **n** = 2 у випадку підтримки участі **A** та **B**.

Протокол 4.1 ША3+

Процедура 4.1.1 - ініціалізація контролером **T**:

1. Оберемо випадкове ціле число **N**: N > n. Оберемо також випадкові i, j < N
2. Оберемо випадкове секретне число x.
3. Обчислимо секретні ідентифікаційні значення $idA = H^i(x) idB = H^j(x)$

$T \rightrightarrows A$: IdA, N ; $T \rightrightarrows B$: IdB, N, де $T \rightrightarrows A$ позначає передачу захищеним каналом від особи **T** до особи **A**

Процедура 4.1.2 - автентифікація

1. Особа **B** обирає випадкове vr, $B \to A: B, vr$. Тут B, як повідомлення, позначає текст, що публічно ідентифікує B.
2. Особа **A** обирає випадкове pr, обчислює свій прихований ідентифікатор $IdA^* = H^{i+N_A}(x)$, $N_A -$ випадкове число, $0 < N_A < N$. Та остаточно обчислює $r = H(vr, pr, B, A)$; $in1 = H\left(IdA^*, r\right); in2 = H\left(H^N\left(IdA^*\right), r\right)$; $A \to B: A, r, pr, in1, in2$

Особа **B** здійснює перевірку $vid = IdB$; якщо $r = H(vr, pr, B, A)$, то викликаємо $Auth(N, r, vid, in1, in2)$, інакше робимо висновок про похибку в каналі.

Процедура 4.1.2 - процедура Auth(N, r, vid, in1, in2)

для $k = 0$ крок 1 до $2N$ РОБИТИ

якщо $(in1 = H(vid, r))$?

то *результат* = (свій,k); $vid = H(vid)$; $k = k + 1$;

якщо$(in2 = H(vid, r))$?

то *результат* = (свій, k)}

результат = «чужий»

РОЗДІЛ 5. ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА

5.1 Методи вимірювань

В якості основних критеріїв оцінки продуктивності криптографічних протоколів в цій роботі розглядаються час та витрачена пам’ять. Обидва ці параметри є критичними для малопотужних систем, як і витрачена на обчислення енергія. Достатньо точні заміри останньої важко оцінити без обчислювальної техніки спорядженої спеціальними апаратрими розширеннями, призначеної для таких вимірювань. Тому, за відсутності доступу до такого обладнання будемо розглядати тільки час та задієну оперативну пам’ять RAM. Надалі, в цьому розділі, будемо називати час та використану алгоритмом пам’ять параметрами вимірювання, а для звернення до процесу вимірювання будемо користуватись поняттям бенчмарк(від англ benchmark).

Всі вимірювання проводилися на платформі x86_64 PC. Для реалізація протоколів та логіки бенчмарку була використана мова програмування C, стандартна бібліотека C, GMP, та Win32 API. В тих місцях. де це можливо та необхідно були використані спеціальні інструкції з розширень архітектури Intel, так, наприклад RDSEED, як надійне джерело випадковості, яке потребують більшість криптоалгоритмів, або SHA256RND та векторні інструкції з розширення VEX 2 для реалізації швидкого обрахування ген-функції SHA256.

5.1.1 Оцінка часу роботи

Час роботи інтерактивних протоколів, розглянутих цьому розділі, вимірювався окремо для стверджувача та верифікатора, оскільки малопотужною системою на практиці часто виступає тільки один із них і виділення алгоритмів кращих за одним із параметрів: загальний час підпису та загальний час верифікації - є

цікавим питанням. Так, окремі алгоритми, як наприклад GPS обіцяють понижену складність обчислень особи **A,** представляючи її як RFID чіп.

Метод оцінки часу роботи алгоритмів використовує на бібліотечну функцію C clock_gettime, що надає можливість виміру із точністю до 100 нс, що є достатньою точністю для оцінених протоколів. Середній час вимірювання будемо обчислювати, як правило, після 100 раундів запуску алгоритму. Виміри показали, що цього достатньо, щоб розкид середнього значення досяг значення похибки.

5.1.2 Оцінка використаної пам’яті

Для оцінки оперативної пам’яті використаної алгоритмом, була використана функція Win32 api GetProcessMemoryInfo, що дозволяє отримати низку параметрів пам’яті процесу в тому числі кількість виділеної приватної пам’яті.

5.2 Реалізації протоколу Гіллоу — Куіскуотера

Реалізація протоколу представлена модулем gq/gq.c, що надає такі основні функції як: ініціалізація, підпис, генерація випробування та верифікація. Основні домені параметри протоколу представляються двома структурами gq_params - для параметрів контролера T та member_keys - представляє ключову пару стверджувача. Для ініціалізація параметрів контролеру використовувались заздалегідь отримана RSA-трійка n, p, q.

Розглянемо функцію генерації ключів.

```
void gq_keys_create(struct gq_params *params, struct member_keys *mparams)
{
        mpz_init(mparams->x);
        mpz_init(mparams->y);
        mpz_init_set(mparams->e, params->e);
        mpz_init_set(mparams->n, params->n);

        gmpt_rndmod(mparams->x, mparams->n);
        mpz_powm(mparams->y, mparams->x, params->e, params->n);
}
```

На рисунку можна побачити, реалізацію обчислення публічного ключа x^e, за модулем n, для довільного максимального розміру параметрів. Отримавши ключову пару можна здійснити підпис(розв'язок) випробування за допомогою функції gq_solve:

```
void gq_solve(const struct member_keys *params, mpz_t z, const mpz_t r, const mpz_t c)
{
        mpz_powm(z, params->x, c, params->n);
        mpz_mul(z, z, r);
        mpz_mod(z, z, params->n);
}
```

Остаточно, особа B перевіряє підпис z для випробування c користуючись функцією gq_verify і отримує результат true або false.

```
_Bool gq_verify(const mpz_t z, const mpz_t e, const mpz_t a, mpz_t y, const mpz_t c, const mpz_t n)
{
        _Bool result = false;
        mpz_t z_powe;
        mpz_t ay_powc;
        mpz_init(z_powe);
        mpz_init(ay_powc);

        mpz_powm(z_powe, z, e, n);
        mpz_powm(ay_powc, y, c, n);
        mpz_mul(ay_powc, ay_powc, a);
        mpz_mod(ay_powc, ay_powc, n);
        result = mpz_cmp(ay_powc, z_powe) == 0;

        mpz_clear(z_powe);
        mpz_clear(ay_powc);
        return result;
}
```

5.3 Реалізації протоколу Фейге — Фіата — Шаміра

Реалізація протоколу ідентифікації Фейге — Фіата — Шаміра наведена в модулі ffs/ffs.c. Загальна структура інтерфейсу відповідає вже розглянутому протоколу GQ, і є спільною для всіх Σ-протоколів.

Наведемо основну функцію верифікації підпису ffs_solve.

```
void ffs_solve(struct ffs_member_params *params, mpz_t Y, mpz_t E, mpz_t R, mpz_t n)
{
        mpz_set(Y, R);
        for (int i = 0; i < FFS_KMAX; i++) {
                if (mpz_tstbit(E, i))
                        mpz_mul(Y, Y, params->s_arr[i]);
        }
        mpz_mod(Y, Y, n);
}
```

На рисунку можна побачити реалізацію кроку (3) Процедури 3.2.1.3. FFS_KMAX представляє значення **k** - тобто довжину випробування. В результаті, в змінну розв’язку Y ми записуємо спочатку R, а потім домножуємо її на елементи масиву приватного ключа A. згідно стану випробування E. Для перевірки розв’язку Y слугує функція ffs_verify:

```
_Bool ffs_verify(mpz_t i_arr[FFS_KMAX], mpz_t X, mpz_t Y, mpz_t E, mpz_t n)
{
        _Bool result = true;
        mpz_t real_x;
        mpz_init(real_x);
        mpz_pow_ui(real_x, Y, 2);

        for (int i = 0; i < FFS_KMAX; i++) {
                if (mpz_tstbit(E, i))
                        mpz_mul(real_x, real_x, i_arr[i]);
        }
        mpz_mod(real_x, real_x, n);

        if (mpz_cmpabs(X, real_x) != 0)
                result = false;

        mpz_clear(real_x);
        return result;
}
```

5.4 Реалізації протоколу Girault Poupard Stern

Реалізація ідентифікації GPS майже ідентична Шнорра, за виключенням спеціальної процедури для завантаження обчисленого наперед внеску (купону):

```
void gps_load_coupon(struct schnorr_params *params, mpz_t x, mpz_t r)
{
        switch (params->bitsize) {
        case GPS_L_1024:
                mpz_set_str(x, coupon_x_1024, 10);
                mpz_set_str(r, coupon_r_1024, 10);
                break;

        case GPS_L_2048:
                mpz_set_str(x, coupon_x_2048, 10);
                mpz_set_str(r, coupon_r_2048, 10);
                break;

        case GPS_L_3072:
                mpz_set_str(x, coupon_x_3072, 10);
                mpz_set_str(r, coupon_r_3072, 10);
                break;
        default:
                perror("invalid key length");
                break;
        }
}
```

Бачимо - для трьох можливих передбачених довжин ключа протоколу, підвантажується окрема пара x, r обрахована попередньо. Зрозуміло, що в практичній реалізації купон має бути різним кожного разу, але у випадку бенчмарку використання одного купону не впливає на оцінювані параметри. Також зміни зазнала функція підпису - тут прибрано крок редукції за модулем.

```
void gps_sign(mpz_t y, mpz_t r, mpz_t s, mpz_t e)
{
        mpz_mul(y, s, e);
        mpz_add(y, y, r);
}
```

5.5 Реалізації протоколу Шнорра

Наведемо функцію створення ключової пари стверджувача

```
void schnorr_user_keys(mpz_t s, mpz_t v, struct schnorr_params *params)
{
        gmpt_rndmod(s, params->q);
        /* v = g ^ -s mod p */
        mpz_sub(v, params->q, s);
        mpz_powm(v, params->g, v, params->p);
}
```

Згенерувавши випадковий приватний ключ s за модулем q - обчислюємо крок (2) процедури 2.1.1.2, отримуючи публічний ключ v. Здійснити підпис випробування **e** можна за допомогою функції schnorr_sign

```
void schnorr_sign(struct schnorr_params *params, mpz_t y, mpz_t r, mpz_t s, mpz_t e)
{
        mpz_mul(y, s, e);
        mpz_add(y, y, r);
        mpz_mod(y, y, params->q);
}
```

Можна помітити, що на відміну від GPS, тут присутня модулярна редукція результату.

5.6 Реалізації протоколу Elliptic Curve Digital Signature Algorithm

Реалізацію протоколу ECDSA можна знайти в модулі ecdsa/ecdsa.c. Наведемо функцію підпису ecdsa_sign, шо при ідентифікації має бути використана для підпису випробування **e.**

```
void ecdsa_sign(const struct ecdsa_params *params, const struct member_keys *keys, const mpz_t e, mpz_t r, mpz_t s)
{
        mpz_t k;
        struct curve_point kG;
        mpz_init(k);
        curve_point_init(&kG);

        do {
                gmpt_rndmod(k, params->n);

                ecdsa_mul(params, params->G, &kG, k);
                mpz_mod(r, kG.x, params->n);
        } while (mpz_cmp_ui(r, 0) == 0);

        mpz_invert(k, k, params->n);
        mpz_mul(s, keys->d, r);
        mpz_add(s, s, e);
        mpz_mul(s, s, k);
        mpz_mod(s, s, params->n);

        curve_point_clear(&kG);
        mpz_clear(k);
}
```

Як бачимо крім функцій GNUMP, тут ми використовуємо додаткові процедури для операціями над точками еліптичних кривих: множення, додавання та квадрат. Ці функції є відносно великими. тому приведені в додатках 1-3.

Генерація приватного та публічного ключа представлена відносно простою процедурою, що використовує тільки операцію множення точки на скаляр

```
void ecdsa_keys_create(const struct ecdsa_params *params, struct member_keys *keys)
{
        mpz_init(keys->d);
        curve_point_init(&keys->Q);
        gmpt_rndmod(keys->d, params->n);
        ecdsa_mul(params, params->G, &keys->Q, keys->d);
}
```

6.7 Реалізації протоколу ША3+

Реалізація протоколу ША3+ наведена у модулі fa3x. На відміну від розглянутих сігма протоколів, ключі члену коаліції обчислюються контролером. Цю операцію представляє процедура fa3x_controller_add_memeber, що має виконуватися лише на контролері коаліції та оперує його приватними параметрами.

```
_Bool fa3x_controller_add_memeber(struct controller_params *cparams, struct member_keys *mparams)
{
        if (cparams->member_count + 1 < cparams->N) {
                mparams->N = cparams->N;
                cparams->member_count += 1;
                fa3x_hash_superpos(mparams->id, cparams->x, cparams->member_count);
                return true;
        }
        return false;
}
```

Процедура fa3x_hash_superpos є основною для цього протоколу та обчислює ланцюг гешів.

```
static void fa3x_hash_superpos(fa3x_key_t result, const fa3x_block x, size_t power)
{
        struct sha256_state state = {0};
        fa3x_block block;
        memcpy(block, x, sizeof(fa3x_block));
        for (size_t i = 0; i < power; i++) {
                sha256_state_init(&state);
                sha256_process_x86(&state, block, sizeof(fa3x_block));
                memset(block, 0, sizeof(fa3x_block));
                memcpy(block, state.hash_data, sizeof(fa3x_key_t));
        }
        memcpy(result, block, sizeof(fa3x_key_t));
}
```

Як було зазначено, використаним генем є sha256. Наведемо також функцію member_auth:

```
_Bool fa3x_member_auth(size_t N, const fa3x_key_t r, const fa3x_key_t vid,
const fa3x_key_t in1, const fa3x_key_t in2)
{
        fa3x_block block = {0};
        fa3x_key_t vid_temp, hvidr;
        memcpy(vid_temp, vid, sizeof(fa3x_key_t));
        for (size_t i = 1; i < 2 * N; i++) {
                fa3x_member_hash(hvidr, vid_temp, r);
                if (memcmp(hvidr, in1, sizeof(fa3x_key_t)) == 0)
                        return true;
                memcpy(block, vid_temp, sizeof(fa3x_key_t));
                fa3x_hash_superpos(vid_temp, block, 1);
                fa3x_member_hash(hvidr, vid_temp, r);
                if (memcmp(hvidr, in2, sizeof(fa3x_key_t)) == 0)
                        return true;
        }
        return false;
}
```

6. АНАЛІЗ РЕЗУЛЬТАТІВ

6.1 Аналіз результатів роботи протоколу Гіллоу — Куіскуотера

Таблиця 6.1

Розмір Ключа біт	Час доведення мкс	Час перевірки мкс	Час загалом мкс
1024	17.3	13.8	31.1
2048	46.9	47.2	94.0
3072	91.2	91.0	182.2

Використання пам’яті RAM: 40.6 КБ

6.2 Аналіз результатів роботи протоколу Фейге — Фіата — Шаміра

Таблиця 6.2

Розмір Ключа біт	Час доведення мкс	Час перевірки мкс	Час загалом мкс
1024	20.0	24.4	44.4
2048	53.7	66.0	119.7
3072	111.8	133.7	245.5

Використання пам’яті RAM: 42.1 КБ

6.3 Аналіз результатів роботи протоколу Girault Poupard Stern

Таблиця 6.3

Розмір Ключа біт	Час доведення мкс	Час перевірки мкс	Час загалом мкс
1024	1.3	141.1	142.4

2048	1.8	561.5	563.3
3072	2.9	1271.0	1273.9

Використання пам’яті RAM: 40.9 КБ

6.4 Аналіз результатів роботи протоколу Шнорра

Таблиця 6.4

Розмір Ключа біт	Час доведення мкс	Час перевірки мкс	Час загалом мкс
1024	63.5	122.1	185.6
2048	277.9	556.5	834.4
3072	656.1	1293.2	1949.3

Використання пам’яті RAM: 40.4 КБ

6.5 Аналіз результатів роботи протоколу Elliptic Curve Digital Signature Algorithm

Таблиця 6.5

Розмір Ключа біт	Час доведення мкс	Час перевірки мкс	Час загалом мкс
256	770.0	1535.9	2306.0
384	1591.4	3180.8	4772.2
521	2869.9	5724.0	8594.0

Використання пам’яті RAM: 46.2 КБ

6.6 Аналіз результатів роботи протоколу Швидка Автентифікація 3+

Таблиця 6.6

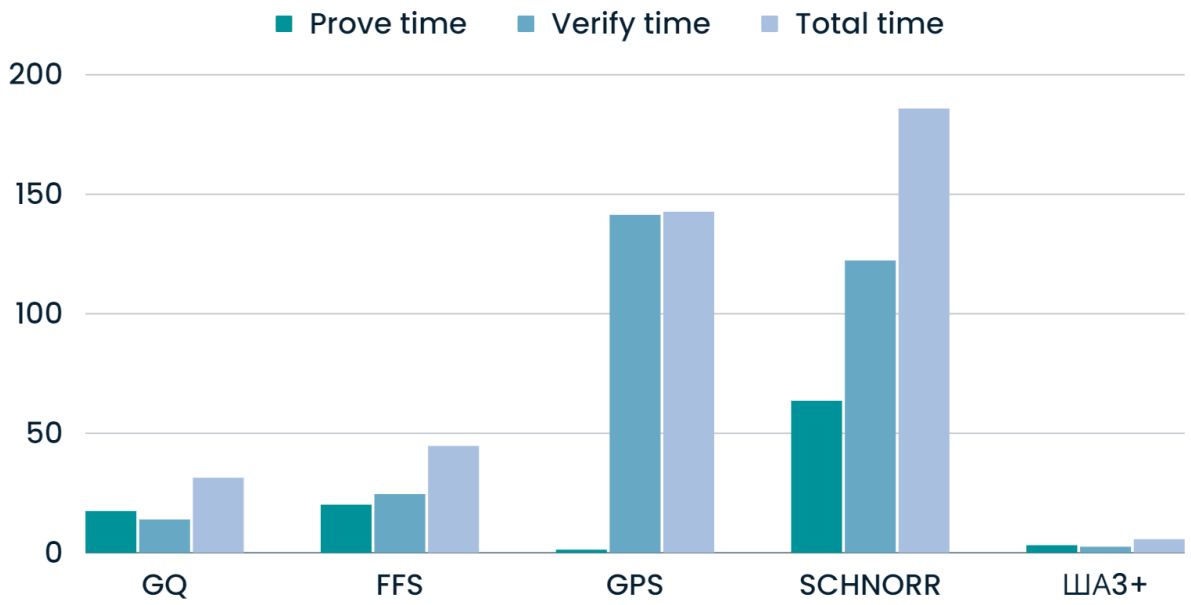
Максимальний ро групи	Час доведення мкс	Час перевірки мкс	Час загалом мкс
10	3.0	2.4	5.4
100	7.8	6.9	14.7
1000	61.7	54.2	115.9
10000	576.1	495.7	1071.8
100000	5968.0	5622.1	11590.1

Використання пам’яті RAM: 1 КБ

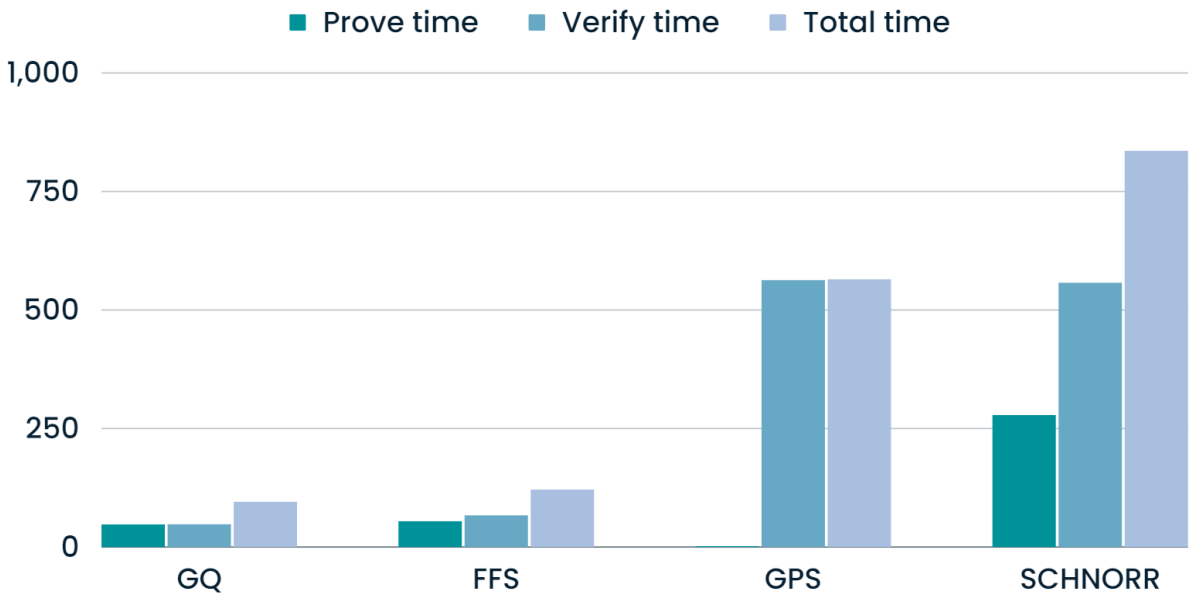
6.7 Зведені результати

На діаграмах 7.7.1-3 представлено час основних операцій для рекомендованої мінімальної, середньої та максимальної довжини ключа. Результати алгоритму ECDSA на наведені на цих трьох діаграмах, оскільки вони на порядок більші, ніж в інших алгоритмах і тому будуть

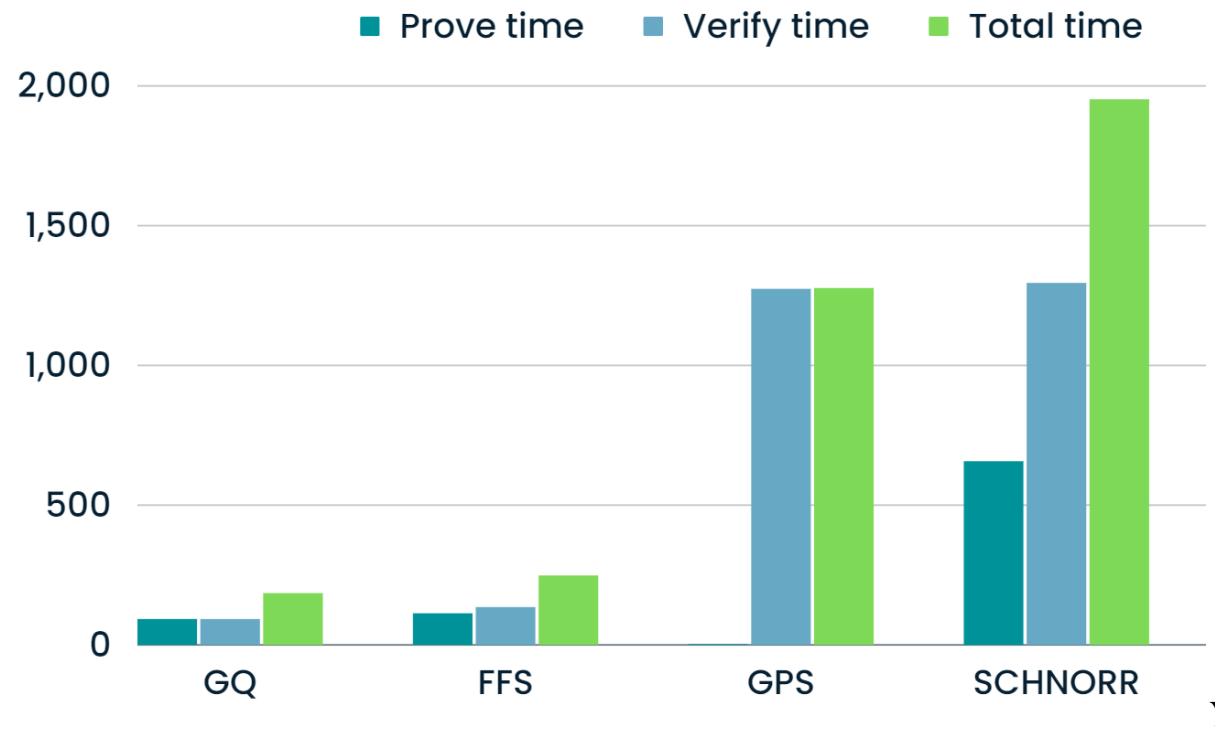
Діаграма 6.7.1



Діаграма 6.7.2

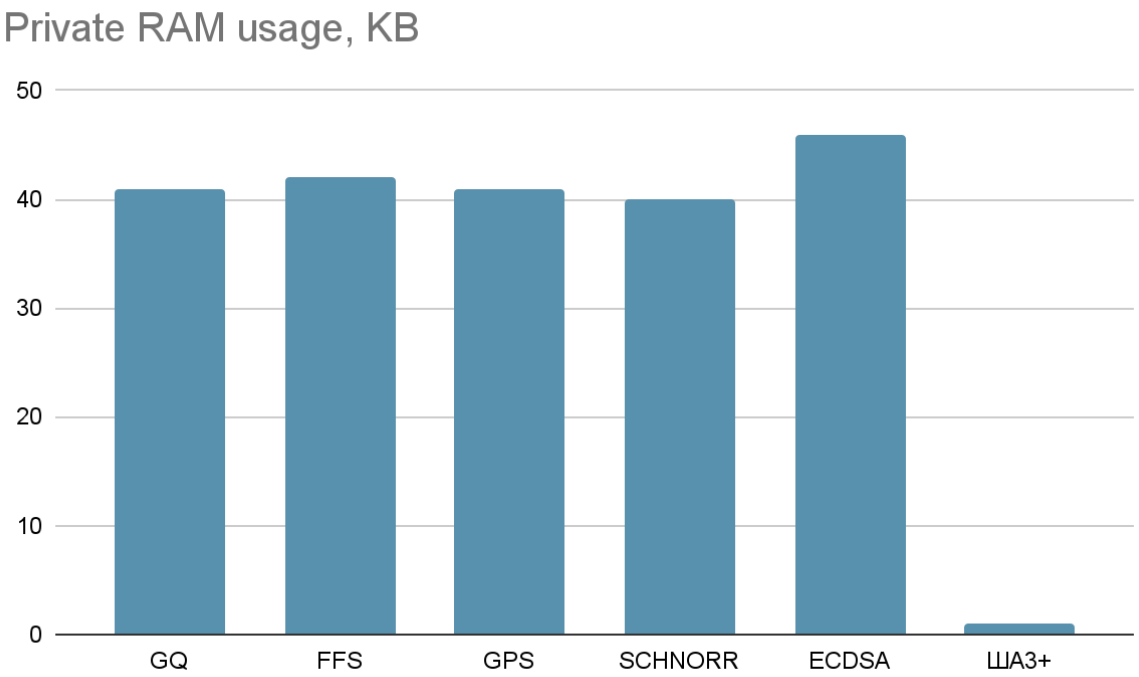


Діаграма 6.7.3



У діаграмі 7.7.4 неведомо середнє значення використаної пам’яті RAM для розглянутих протоколів.

Діаграма 6.7.4


Private RAM usage, KB
50
40
30
20
10
0
GQ
FFS
GPS
SCHNORR
ECDSA
ША3+

ВИСНОВКИ

В результаті виконання роботи було створено реалізації протоколів ідентифікації Шнорра, Фейге — Фіата — Шаміра, Гіллоу — Куіскуотера, GPS, ECDSA та ША3+, а також додаткові механізми для оцінки їх продуктивності. Проаналізувавши дані про час роботи та використану оперативну пам’ять можна зробити наступні висновки.

Протокол ША3+, має найкращий показник часу автентифікації загалом, а найменшу кількість використаної пам’яті. В той же час, схема GPS, демонтруючи дуже близький(практично однаковий) час підпису випробування, але має набагато більший час верифікації, на рівні базового варіанту, а саме протоколу Шнорра. Протокол GPS має очікуване значне прискорення, з боку стверджувача, за рахунок використання обчислених заздалегідь купонів - але варто зазначити, що це також створює додаткові складнощі, у вигляді проблеми зберігання достатньої кількості цих купонів або їх захищеної передачі по мірі використання.

Протоколи Фейге — Фіата — Шаміра та Гіллоу — Куіскуотера демонструють дуже близький результат, як з точки зору часу, так і використаної пам’яті, при цьому з точки зору пам’яті також подібні до GPS та ідентифікації Шнорра. Можна стверджувати, що ця близькість обумовлена операціями з використанням довгої арифметики з однаковим модулем для цих алгоритмів, оскільки основні запити додаткової пам’яті в реалізації здійснює саме бібліотека довгої арифметики.

Найбільші значення по часу та пам’яті продемонструвала реалізація ECDSA, при чому майже на порядок перевершивши інші схеми, з точки зору параметра часу. Це може бути обумовлене загальною обчислювальною складністю адитивних операцій над еліптичними кривими, що робить подібні алгоритми

підпису з використанням властивостей еліптичних кривих менш застосовними для малопотужних пристроїв.

Підсумовуючи, за результатами роботи я можу стверджувати, що серед розглянутих алгоритмів найбільше задовольняють умовам поставленим в меті алгоритми ША3+ та GPS.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Guillou L. Self-Certified Public Keys. / L. Guillou, J. Quisquater. – Davos, 1988. – 473 c. – (Advances in Cryptology – EUROCRYPT '88).
2. Fiat A., Shamir A. How to Prove Yourself: Practical Solutions to Identification and Signature Problems // Advances in Cryptology — CRYPTO ’86: 6th Annual International Cryptology Conference, Santa Barbara, California, USA, 1986, Proceedings / A. M. Odlyzko — Springer Berlin Heidelberg, 1987. — P. 186–194. — 490 p. — (Lecture Notes in Computer Science; Vol. 263).
3. Schneier B. Applied Cryptography : Protocols, Algorithms, and Source Code in C / Bruce Schneier. – New York: Wiley, 1996. – 784 c.
4. Анісімов А. В. Приватна комунікація. Київ, 2023. 23с.
5. Schnorr C. Efficient signature generation by smart cards / Claus-Peter Schnorr. – Berlin: Springer-Verlag, 1991. – (Journal of Cryptology; вип. 4).
6. Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm [Електронний ресурс] // Accredited Standards Committee X9. – 1998. – Режим доступу до ресурсу: https://safecurves.cr.yp.to/grouper.ieee.org/groups/1363/private/x9-62-09-20-98.pdf.
7. Girault M. Self-Certified Public Keys. / Mark Girault. – Brighton, 1991. – 556 c. – (Conference: Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques).
8. Damg˚ard I. On Σ-protocols / Ivan Damg˚ard. – Aarhus, 2010. – 22 c. – (Cryptologic Protocol Theory).

ДОДАТКИ

Додаток 1. Процедура множення над еліптичними кривими.

```
void ecdsa_mul(const struct ecdsa_params *params, struct curve_point p, struct curve_point *r, mpz_t m)
{
        struct curve_point Q, T;
        mpz_init(Q.x); mpz_init(Q.y);
        mpz_init(T.x); mpz_init(T.y);
        long no_of_bits, loop;

        no_of_bits = mpz_sizeinbase(m, 2);
        mpz_set_ui(r->x, 0);
        mpz_set_ui(r->y, 0);
        if(mpz_cmp_ui(m, 0) == 0)
                return;

        mpz_set(Q.x, p.x);
        mpz_set(Q.y, p.y);
        if(mpz_tstbit(m, 0) == 1){
                mpz_set(r->x, p.x);
                mpz_set(r->y, p.y);
        }

        for(loop = 1; loop < no_of_bits; loop++) {
                mpz_set_ui(T.x, 0);
                mpz_set_ui(T.y, 0);
                ecdsa_double(params, Q, &T);

                mpz_set(Q.x, T.x);
                mpz_set(Q.y, T.y);
                mpz_set(T.x, r->x);
                mpz_set(T.y, r->y);
                if(mpz_tstbit(m, loop))
                        ecdsa_add(params, T, Q, r);
        }

        mpz_clear(Q.x); mpz_clear(Q.y);
        mpz_clear(T.x); mpz_clear(T.y);
}
```

Додаток 2. Сума над еліптичними кривими.

```
void ecdsa_add(const struct ecdsa_params *params, struct curve_point p, struct curve_point q, struct curve_point *r)
{
        mpz_mod(p.x, p.x, params->p);
        mpz_mod(p.y, p.y, params->p);
        mpz_mod(q.x, q.x, params->p);
        mpz_mod(q.y, q.y, params->p);

        if(mpz_cmp_ui(p.x, 0) == 0 && mpz_cmp_ui(p.y, 0) == 0) {
                mpz_set(r->x, q.x);
                mpz_set(r->y, q.y);
                return;
        }

        if(mpz_cmp_ui(q.x, 0) == 0 && mpz_cmp_ui(q.y, 0) == 0) {
                mpz_set(r->x, p.x);
                mpz_set(r->y, p.y);
                return;
        }

        mpz_t temp;
        mpz_init(temp);

        if (mpz_cmp_ui(q.y, 0) != 0) {
                mpz_sub(temp, params->p, q.y);
                mpz_mod(temp, temp, params->p);
        } else
                mpz_set_ui(temp, 0);

        if (mpz_cmp(p.y, temp) == 0 && mpz_cmp(p.x, q.x) == 0) {
                mpz_set_ui(r->x, 0);
                mpz_set_ui(r->y, 0);
                mpz_clear(temp);
                return;
        }

        if(mpz_cmp(p.x, q.x) == 0 && mpz_cmp(p.y, q.y) == 0)    {
                ecdsa_double(params, p, r);

                mpz_clear(temp);
                return;
        } else {
                mpz_t slope;
                mpz_init_set_ui(slope, 0);

                mpz_sub(temp, p.x, q.x);
                mpz_mod(temp, temp, params->p);
                mpz_invert(temp, temp, params->p);
                mpz_sub(slope, p.y, q.y);
                mpz_mul(slope, slope, temp);
                mpz_mod(slope, slope, params->p);
                mpz_mul(r->x, slope, slope);
                mpz_sub(r->x, r->x, p.x);
                mpz_sub(r->x, r->x, q.x);
                mpz_mod(r->x, r->x, params->p);
                mpz_sub(temp, p.x, r->x);
                mpz_mul(r->y, slope, temp);
                mpz_sub(r->y, r->y, p.y);
                mpz_mod(r->y, r->y, params->p);

                mpz_clear(temp);
                mpz_clear(slope);
                return;
        }
}
```

Додаток 3. Квадрат над еліптичними кривими

```
void ecdsa_double(const struct ecdsa_params *params, struct curve_point p, struct curve_point *r)
{
        mpz_t slope, temp;
        mpz_init(temp);
        mpz_init(slope);

        if(mpz_cmp_ui(p.y, 0) != 0) {
                mpz_mul_ui(temp, p.y, 2);
                mpz_invert(temp, temp, params->p);
                mpz_mul(slope, p.x, p.x);
                mpz_mul_ui(slope, slope, 3);
                mpz_add(slope, slope, params->a);
                mpz_mul(slope, slope, temp);
                mpz_mod(slope, slope, params->p);
                mpz_mul(r->x, slope, slope);
                mpz_sub(r->x, r->x, p.x);
                mpz_sub(r->x, r->x, p.x);
                mpz_mod(r->x, r->x, params->p);
                mpz_sub(temp, p.x, r->x);
                mpz_mul(r->y, slope, temp);
                mpz_sub(r->y, r->y, p.y);
                mpz_mod(r->y, r->y, params->p);
        } else {
                mpz_set_ui(r->x, 0);
                mpz_set_ui(r->y, 0);
        }
        mpz_clear(temp);
        mpz_clear(slope);
}
```