Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

Бакалаврська кваліфікаційна робота на тему:
"Система захищеного електронного документообігу"

Виконав: студент 4 курсу групи 1БКС-21б
спеціальності 125 Кібербезпека
_____ Дмитро ЛОБАЙ

Керівник: к. т. н., доцент каф. ЗІ
_____ Юрій БАРИШЕВ
"16" червня 2025 р

Рецензент: к. т. н., доц., доц. каф. ПЗ
_____ Вікторія ВОЙТКО
"16" червня 2025 р

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
"16" червня 2025 р

Вінниця ВНТУ – 2025 року

Міністерство освіти і науки України
Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти І (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 125 Кібербезпека
Освітньо-професійна програма – Кібербезпека критичних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ,
д.т.н., проф.
_____ Володимир ЛУЖЕЦЬКИЙ
"21" березня 2025 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Дмитру ЛОБАЮ

1. Тема роботи: "Система захищеного електронного документообігу"
   керівник роботи: Юрій БАРИШЕВ, к.т.н., доцент кафедри ЗІ,
   затверджені наказом ректора ВНТУ від 20 березня 2025 року № 97.
2. Строк подання студентом роботи 16 червня 2025 р.
3. Вихідні дані до роботи:
- підтримка форматів документів doc, docx, pdf;
- максимальна кількість підписантів – не менше 10;
- алгоритму формування індивідуальних підписів – ECDSA.
4. Зміст текстової частини: Вступ. 1. Аналіз систем електронного документообігу. 2. Архітектура системи. 3. Алгоритм роботи системи. 4. Тестування системи. Висновки. Список використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу: порівняльний аналіз систем електронного документообігу, методи групового підпису, узагальнена архітектура, блок групового підпису, узагальнений алгоритм, алгоритм формування групового підпису, алгоритм перевірки групового підпису, алгоритм ЕЦП, результати модульного тестування, результати статичного тестування безпеки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Ю. БАРИШЕВ, к.т.н., доц. каф.ЗІ	21.03	31.0
2	Ю. БАРИШЕВ, к.т.н., доц. каф.ЗІ	21.03	17.
3	Ю. БАРИШЕВ, к.т.н., доц. каф.ЗІ	21.03	05
4	Ю. БАРИШЕВ, к.т.н., доц. каф.ЗІ	21.03	19.

7. Дата видачі завдання 21.03 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Приміт
1	Аналіз завдання. Вступ	24.03.25 – 30.03.25	
2	Аналіз інформаційних джерел за напрямком бакалаврської кваліфікаційної роботи	31.03.25 – 13.04.25	
3	Розробка рішень, моделей, алгоритмів	14.04.25 – 04.05.25	
4	Практична реалізація, моделювання, експериментування, результати	05.05.25 – 14.05.25	
5	Аналіз виконання БКР, висновки	15.05.25 – 18.05.25	
6	Оформлення пояснювальної записки	19.05.25 – 26.05.25	
7	Попередній захист БКР	27.05.25 – 30.05.25	
8	Виправлення зауважень, перевірка на наявність текстових запозичень, підготовка ілюстративного матеріалу	31.05.25 – 10.06.25	
9	Представлення БКР до захисту, рецензування	11.06.25 – 13.06.25	

Студент _____ Дмитро ЛОБАЙ
(підпис)

Керівник роботи _____ Юрій БАРИШЕВ
(підпис)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 68 сторінок формату А4, на яких є 36 рисунків, 3 таблиці, список використаних джерел містить 50 найменувань.

Бакалаврська кваліфікаційна робота присвячена розробці системи захищеного електронного документообігу. В результаті аналізу систем електронного документообігу було виявлено відсутність будь-яких альтернатив звичайних електронних підписів та печаток у таких системах. Як наслідок було прийнято рішення, щодо розробки системи електронного документообігу із можливістю використання групового підпису, у його ролі буде виступати схема Short Group Signature та ECDSA для звичайного електронного підпису. Наведено результати розробки архітектури та алгоритмів функціонування засобу. Здійснено програмну реалізацію системи у вигляді веб-застосунку. Проведено модульне та інтеграційне тестування системи, у результаті яких було підтверджено коректність роботи системи. Наведено результати статичного тестування безпеки, яке підтвердило відсутність вразливостей у реалізації.

Ключові слова: системи електронного документообігу, груповий підпис, електронно-цифровий підпис, конфіденційність, приватність користувачів, ідентичність документів, тестування, веб-застосунок.

ABSTRACT

The bachelor's thesis consists of 68 pages of A4 format, which includes 36 figures, 3 tables, and a list of references containing 50 titles.

The bachelor's thesis is devoted to the development of a secure electronic document management system. The analysis of electronic document management systems revealed the absence of any alternatives to conventional electronic signatures and stamps in such systems. Therefore, a decision was made to develop an electronic document management system with the possibility of using a group signature, which will be based on the Short Group Signature scheme and ECDSA for an electronic signature. The study presents the results of the development of the architecture and algorithms of the system. The system was implemented as a web application. Unit and integration testing confirmed the system's correct functionality. The results of static security testing are also provided, which confirmed the absence of vulnerabilities in the implementation.

Keywords: electronic document management systems, group signature, electronic digital signature, confidentiality, user privacy, document identity, testing, web application.

ЗМІСТ

ВСТУП

Рух документів в установі є ключовою складовою управління процесами її роботи. Сьогодні для цього використовують системи електронного документообігу, вони забезпечують систематизацію, збереження та швидкий доступ до документів. Однак одночасно із позитивністю їх використання постає питання безпеки приватності таких систем та їх користувачів. Одним із методів вирішення такої проблеми є використання додаткових засобів забезпечення ідентичності документів. Зазвичай в уже існуючих системах використовують ті чи інші реалізації цифрових печаток та/або цифрових підписів, однак було б доречно розширити такий функціонал додатковими методами захисту. Існує велика кількість подібних засобів, однак серед них чудово було б виділити саме групові підписи. Цей метод є альтернативою звичайних цифрових підписів, відділяється своєю високою різноманітністю у функціоналі, в залежності від реалізованої схеми підпису, із головною особливістю у вигляді забезпечення часткової або повної анонімності користувача. Як приклад у роботі самих систем документообігу такі підписи можна використати у випадках коли необхідно приховати особу підписанта документа – це може бути певна звітність результатів роботи у компанії, оцінка роботи над якимись процесами тощо. Було б чудово згадати, що існують методи групового підпису із можливістю розкриття підпису файлу, що надає змогу ідентифікації підписанта при крайній необхідності, у випадку систем документообігу така функціональна особливість була б доречною, так як повна анонімність підписанта у більшості випадків буде тільки перепоною у процесі роботи системи. Взявши до уваги потребу додаткового захисту приватності та переваги такого методу підписання документів можна стверджувати, що розробка системи документообігу із можливістю підписання документів саме груповим підписом є цілком актуальною.

Об’єктом бакалаврської кваліфікаційної роботи є система електронного документообігу.

Предметом бакалаврської кваліфікаційної роботи є методи та засоби захисту приватності у системах електронного документообігу.

Метою роботи є покращення захисту приватності у системах електронного документообігу на основі використання групових підписів.

Для досягнення мети потрібно виконати такий перелік завдань:

- аналіз завдань використання систем електронного документообігу;
- порівняльний аналіз відомих рішень;
- аналіз методів формування електронно-цифрових підписів;
- розробка архітектури системи електронного документообігу;
- розробка алгоритмів роботи відповідно до архітектури;
- обґрунтування засобі розробки;
- тестування розробленої системи.

Результати отримані у бакалаврській кваліфікаційній роботі апробована на Всеукраїнській науково-технічній конференції факультету інформаційних технологій та комп’ютерної інженерії (2025) [1].

1 АНАЛІЗ СИСТЕМ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

1.1 Аналіз завдань використання систем електронного документообігу

Процес документообігу в будь-якій організації є її критичним процесом без якого її функціонування є неможливим, за їх допомогою виконується передача суттєвої частини всієї інформації, що необхідна для її коректної роботи.

Відповідно до типів інформації, що циркулює на будь-якому підприємстві, можна вирізнити внутрішній та зовнішній документообіг [2, 3].

Як можна здогадатися із назви, внутрішній документообіг – це обмін документами безпосередньо всередині підприємства між різним підрозділами та робітниками однієї і тієї ж організації. З іншого боку зовнішній документообіг – це рух документів між компанією та її клієнтами, бізнес партнерами, державними організаціями тощо.

Існує дві головні методики обігу документів. Паперовий документообіг є більш традиційним форматом роботи із документами, відповідно до його назви головною його особливістю є те, що всі процеси, які проводяться із документами виконуються у фізичному паперовому вигляді. Як противага до нього у електронному документообігові всі процеси виконуються у електронному форматі [4]. Обидва методи мають як свої переваги так і недоліки. Таким чином електронний документообіг має набагато більшу швидкість та зручність виконання процесів над документами, є менш витратним та при дотриманні необхідних умов безпеки, є більш безпечнішим; паперовий документообіг є набагато простішим у своїй реалізації, підтримці його роботи та забезпеченні безпеки його функціонування. Виходячи із таких їх властивостей, для забезпечення найкращих результатів зазвичай використовують комбіновані варіанти таких методів документообігу із більшою частиною, що припадає на електронний спосіб.

Системи електронного документообігу (СЕД) дозволяють прискорити майже всі процеси, що позв’язані із роботою над документами [5]. Автоматизація документообігу дозволяє створювати та реєструвати документи за лічені хвилини із можливістю їх маршрутизації у середині системи, контролю роботи над ними

іншими співробітниками та можливістю швидкого архівування. СЕД надають можливість швидкої передачі документів між працівниками та відділами об’єкта, бачити статус роботи на певним документом у реальному часі тощо.

Для кращого розуміння роботи СЕД доречно розглянути задачі для виконання яких вони використовуються. Головні задачі цих систем [6]:

- приймання, оброблення, передавання усіх документів, що циркулюють у системі (зовнішні та внутрішні);
- створення нових екземплярів документів;
- контроль руху документів;
- ведення електронного архівування;
- розмежування доступу та забезпечення безпеки документів;
- можливість спільної роботи між користувачами;
- автоматизація процесів;
- інтегрування із іншими системами;
- розмежування прав доступу.

Таким чином система в своєму функціоналі повинно мати можливість отримання документів із різних джерел як з внутрішніх, так із зовнішніх, та зберігати їх у сховищі, самі документи мають бути класифіковані за їх походженням. До них потрібно прив’язати такі атрибути як дата, автор, тип документа, має бути забезпеченим можливість подальшої їх передавання усередині або ззовні системи. Повинна бути присутня можливість створення нових документів за допомогою заздалегідь створених шаблонів для подальшої їх модифікації та обробки. У самій системі має бути наявною можливість логування дій над документами та їх руху у системі. Для забезпечення безпечного користування системою є необхідність використання розмежування доступу до самих документів та певних функцій у системі.

1.2 Аналіз нормативно-правового регулювання

Оскільки сфера електронного документообігу передбачає створення та обробку документів, які мають юридичне значення, то СЕД має відповідати певним державним нормам. Було проаналізовано низку українських та закордонних нормативно-правових актів, які регулюють питання електронного документообігу.

Закон України “Про електронні документи та електронний документообіг” № 851-IV від 22 травня 2003 року [7] визначає поняття електронного документу, в ньому зазначено, що склад та порядок розміщення обов’язкових реквізитів електронних документів визначається законодавством. Для ідентифікації автора електронного документа може використовуватися електронний підпис (ЕП) та/або електронна печатка, накладанням цього методу ідентифікації завершується створення електронного документа. Суб’єкти документообігу використовують такі засоби у випадках, що були встановлені законодавством або за їх домовленістю. Перевірка цілісності електронного документа проводиться шляхом підтвердження ЕП чи печатки із застосуванням інших засобів і методів захисту інформації з дотриманням вимог законодавства у сфері захисту інформації. Документи повинні бути збережені на електронних носіях у формі, що дозволяє перевірити їх цілісність на термін не менший від строку, встановленого законодавством для відповідних документів на папері, при неможливості збереження на такий термін необхідне проведення дублювання таких даних на інші носії для подальшого збереження. Також закон визначає питання стосовно оригінальності електронного документа, правового статусу, термін електронного документообігу, відправлення, передавання та одержання таких документів, загальну інформацію про організацію електронного документообігу.

У законі України “Про електронну комерцію” від 03 вересня 2015 року № 675-VIII [8] зазначено, що даний закон регулює правові відносини у сфері електронної комерції під час проведення електронних правочинів, окрім таких випадків коли один із сторін договору є іноземець, у цьому випадку норми цього закону застосовуються з урахуванням положень Закону України “Про міжнародне

приватне право” [9]. Зазначено головні особливості, що стосуються учасників відносин у сфері електронної комерції, порядку вчинення електронних правочинів, вирішення спорів та відповідальність у сфері електронної комерції.

Закон України “Про електронну ідентифікацію та електронні довірчі послуги” від 01 грудня 2022 року № 2801- IX [10] регулює взаємодію між юридичними, фізичним особами, суб’єктами владних повноважень у процесі надання і отримання послуг електронної ідентифікації. Визначає принципи забезпечення юридичної значимості електронних документів, захисту персональних даних, інтероперабельності та технологічної нейтральності технічних рішень. Відповідно до цього закону зазначено, що електронна ідентифікація здійснюється з використанням засобів електронної ідентифікації та процедури автентифікації відповідно до схем електронної ідентифікації. Так існує три рівня довіри до засобів електронної ідентифікації, а саме низький, середній та високий рівні. Кваліфікований електронний підпис (КЕП) або печатка забезпечує високий рівень довіри, а використання удосконалених електронних підписів та печаток, які базуються на кваліфікованих сертифікатах відкритих ключів, забезпечує середній рівень довіри. Визначаються вимоги до надавачів електронних довірчих послуг, регламентує порядок використання електронних довірчих послуг, таких як електронні підписи та печатки, описує кваліфіковані електронні довірчі послуги створення, формування, перевірки та підтвердження КЕП чи печатки.

Типова інструкція з документування управлінської інформації в електронній формі та організації роботи з електронними документами в діловодстві, електронного міжвідомчого обміну, затвердженої постановою Кабінету Міністрів України від 17 січня 2018 року № 55 [11], у цьому документі зазначено, що установи повинні використовувати переважно у електронній формі з застосуванням КЕП або удосконаленого ЕП, що базується на кваліфікованому сертифікаті ЕП, кваліфікованої електронної печатки або удосконаленої електронної печатки, що базується на кваліфікованому сертифікаті електронної печатки, та кваліфікованої електронної позначки часу, крім випадків наявності обґрунтованих підстав для документування управлінської інформації у паперовій

формі. До електронних документів підписаних одним із методів ідентифікації підписанта необхідне відтворення візуального підпису або відбитка печатки незалежно від особливостей оформлення документів не допускається.

Порядок роботи з електронними документами у діловодстві та їх підготовки до передавання на архівне зберігання (Наказ Міністерства юстиції України 11.11.2014 № 1886/5) [12] порядок обов'язковий для органів державної влади, місцевого самоврядування, підприємств, установ та організацій незалежно від форми власності. Електронні документи повинні бути підписані КЕП або кваліфікованою електронною печаткою, що забезпечує їхню юридичну силу та автентичність. У разі, коли для документа, на який накладено удосконалений чи інший вид ЕП, за результатами проведеної експертизи цінності визначено постійний або тривалий (понад 10 років) строк зберігання, працівник архівного підрозділу або особа, відповідальна за ведення архіву установи, засвідчує його КЕП. Структура імені файлу електронного документа повинна складатись із таких елементів: ідентифікатор країни походження; ідентифікатор установи; дата електронного документа; реєстраційний індекс електронного документа; версія конвертування; розширення імені файлу. Передбачено процедури підготовки електронних документів до передавання на архівне зберігання, включаючи формування електронних примірників описів справ постійного зберігання та обкладинок архівних електронних справ. Встановлено порядок внесення змін до XML-документів у разі створення нових версій електронних документів, а також процедури знищення конвертованих електронних документів, що не підлягають обліку та зберіганню. Забезпечується автоматичне протоколювання роботи системи та користувачів на автоматизованому робочому місці працівника архіву, а також проведення технічної перевірки електронних документів перед їх передаванням на архівне зберігання.

General Data Protection Regulation [13] – це юридична база, головна мета якої встановити умови для збирання та обробки персональних даних осіб, котрі проживають у межах Європейського союзу та поза його кордонами. Однією із головних задач цього юридичного документу є надати звичайним користувачам

можливість контролю їхньою приватною інформацією, шляхом впровадження відповідальності за дії над їх інформацією, які вчиняє компанія. Вважається одним із найжорсткіших законів, що стосується захисту персональних даних, при порушення умов закону на підприємство порушник будуть накладено штрафи у розмірі до декількох мільйонів євро. У цьому документі йдеться про принципи захисту даних, відповідальність, процедури забезпечення захисту інформації, концепція ”Захист даних за проєктом та за замовчуванням”, при розробці проєкту нового продукту потрібно враховувати захист даних, зазначено випадки, коли дозволено проводити операції над інформацією тощо.

Серія стандартів ISO 30300 (ДСТУ ISO 30300:2015) [14] пропонує узгоджену із цілями та стратегією організації методологію для створення та керування документацією. Встановлює вимоги до політик, процедур, визначення ролей та обов’язків, розробки та впровадження СЕД, оцінювання її результативності та шляхів покращення. Документ ISO 30301 (ДСТУ ISO 30301:2015) [15] містить вимоги до встановлення, підтримки та покращення СЕД. Пропонує методики для полегшення прийняття рішень та розподілення ресурсів. ISO 30302 (ДСТУ ISO 30302:2018) [16] безпосередньо пропонує інструкції для імплементації таких систем у роботу підприємства згідно до умов зазначених у документі ISO 30301.

1.3 Аналіз методів формування електронно-цифрових підписів

Оскільки під час роботи СЕД через неї проходить велика кількість документів різноманітної важливості та юридичної цінності, постає необхідність забезпечення їх ідентичності, через потенційні небезпеки із збоку потенційних порушників безпеки системи так із різноманітних програмних збоїв роботи системи. Для того щоб забезпечити такі характеристики цих документів доцільним є використання електронно-цифрових підписів.

Існує велика кількість різноманітних методів формування електронно-цифрових підписів кожен із яких має свої особливості використання, переваги та

недоліки. Такі методи описують як спосіб підписання файлів, так і шляхи створення ключів для їх роботи. Серед них для більш детального аналізу було обрано найпоширеніші методи створення підписів: методи Rivest–Shamir–Adleman (RSA) [17], Digital Signature Algorithm (DSA) [18], Elliptic Curve DSA (ECDSA) [19], Boneh–Lynn–Shacham (BLS) [20] та ДСТУ 9212:2023 [21].

RSA є один із найбільш популярних асиметричних криптографічних алгоритмів, його зазвичай використовують для шифрування різноманітних повідомлень, окрім цього широко застосовується у роботі електронно-цифрових підписів. Його робота має такий вигляд, спочатку виконується генерування публічного та приватного ключів, обирається два великих простих числа a та b, далі обчислюється їх добуток n, після чого застосовується функція Ойлера від числа n, що має такий вигляд $\varphi(n) = (p - 1) * (q - 1)$, обирається відкрита експонента e така, що $1 < e < \varphi(n)$ та e взаємно просте із $\varphi(n)$, обчислюється показник приватної експоненти d, знаходиться як обернене за модулем $\varphi(n)$ до числа e, таким чином було створено публічний ключ (n, e) та приватний ключ (n, d). Наступним кроком є підписання та перевірка цього документа. Спочатку підписант обчислює геш-код повідомлення, який ще називають дайджест повідомлення, далі цей дайджест шифрується за допомогою приватного ключа, таким чином було створено цифровий підпис, далі для подальшої верифікації підписант передає оригінальний файл та підпис, верифікатор обчислює геш-код оригінального повідомлення та із використання публічного ключа підписанта розшифровує підпис, як результат він отримує оригінальний дайджест повідомлення, якщо він сходиться із геш-кодом оригінального файлом то його оригінальність підтверджується у протилежному випадку оригінальність спростовується [22].

DSA використовується як функція для цифрового підпису, не може бути використаним для шифрування чи безпечної передачі ключів. На початку своєї роботи цей алгоритм створює геш-код файлу, що потрібно підписати, після чого цей код поєднується із випадковим числом k і використовується як вхідні дані для функції підпису, сама функція залежить від публічного, що складається із 3 значень p, q, g, та приватного, має бути випадковим цілим числом у діапазоні $0 < x < q$,

ключів. На виході цієї функції буде підпис із двох елементів s та r. А для перевірки ідентичності цього файлу буде необхідно об’єднати геш-код повідомлення із підписом і занести у функцію перевірки, на виході буде отримано значення рівне до r, з цього можна зазначити, що оригінальність цього файлу підтверджено і навпаки, якщо це значення не задовольняє цю умову [23].

ECDSA – це криптографічний алгоритм відкритим ключем, що використовується для створення ЦП файлів для подальшої їх перевірки на автентичність. За своєю будовою є аналогічним до DSA, однак вирізняється тим, що для генерування ключів використовуються точки еліптичної кривої, криві Вейєрштрасса. Робота самого алгоритму розпочинається із вибору випадкової точки на кривій, після чого генерується випадкове число, що буде слугувати приватним ключем, далі генерується точка на кривій, що буде слугувати відкритим ключем, за допомогою раніше обраних приватного ключа та точки на кривій. Далі виконується процес підпису, обчислюється геш-код файлу, поєднується раніше згенерована точка із випадковим числом k, для створення підпису отримані значення заносить у функцію підпису, разом із приватним ключем, на виході буде отримано підпис, що складається із елементів s та r. Для верифікації необхідно занести значення публічного ключа, точки кривої, підпис та геш-коду повідомлення у функцію перевірки, на виході буде отримано значення певної точки, якщо координата x цієї точки рівна r, це означає, що підпис дійсний, інакше – ні [24]. Схожим чином працює ще одна відома схема ЕП EdDSA [25], головна відмінність якої полягає у використанні кривих Едвардса.

BLS – це криптографічна схема цифрового підпису, яка базується на властивостях пар в еліптичних кривих. Вирізняється можливістю створення підпису короткої довжини та можливістю агрегування кількох підписів в один. Ця схема працює схожим чином, користувач випадковим чином вибирає секретний ключ із заздалегідь визначеного поля і обчислює відповідний відкритий ключ як точку на заздалегідь визначеній еліптичній кривій. Щоб підписати, користувач множить хеш повідомлення на секретний ключ . Отриманий підпис також є точкою кривої. Для перевірки підпису використовується білінійне сполучення, що має

вигляд рівняння, якщо обидві сторони рівні — підпис дійсний, у іншому випадку він недійсний [26].

ДСТУ 9212:2023 [21] стандарт установлює криптографічні алгоритми електронного підпису на алгебраїчних решітках, використовується для забезпечення різноманітних послуг безпеки та є основою для електронної ідентифікації, електронної автентифікації, вдосконалених і кваліфікованих електронного підпису, печатки та створення електронного документа. У стандарті описано алгоритми ЕП, які використовують криптографічні перетворення на основі перетворення "Fiat-Shamir з відхилами" на алгебраїчних решітках. Як прототип застосовують алгоритм, розширений для забезпечення більшої криптостійкості в умовах застосування квантових комп'ютерів. Криптографічні перетворення типу ЕП виконують у кільцях поліномів Z [X] / (Xn + 1) та Zq [X] / (Xn + 1) степеня n для цілого числа q з використанням асиметричних пар ключів: для ЕП з використанням особистого (секретного) ключа підписувача, а для перевіряння ЕП з використанням відкритого ключа перевіряння ЕП [27].

Однак на практиці можна зіштовхнутися із ситуаціями коли звичайного ЦП не вистачає або його безпосереднє використання може бути не найефективнішим чи менш підходящим порівняно із іншими методами. Наприклад у випадках, коли підписання певного документу потребує участь великої кількості підписантів і як наслідок сам підпис документу може зайняти досить велику кількість часу, або коли ідентичність того чи іншого документу потрібно засвідчити без розголошення особистості особи, що його підписала, звичайний ЗП буде не ефективний або зовсім не придатним. Таким чином було б доречно використати такі методи як груповий підпис (ГП) [28], кільцевий підпис (КП) [29] або пороговий підпис в особливості його один із його варіантів мультипідпис [30].

ГП один із методів реалізації ЦП, головна властивість якої полягає в тому, що вона дозволяє члену групи анонімно підписувати повідомлення від імені групи, тобто при перевірці документу перевіряючий може засвідчитися тільки, що документ був підписаний одним із членів групи одна не матиме інформації хто це саме зробив. Важлива особливість такого методу є присутності адміністратора

підпису у процесі його реалізації, він забезпечує є можливість встановлення особи підписанта та додає нових членів у групу.

КП по своїй концепції схожий до ГП, але відрізняється тим, що підписант самостійно обирає склад цієї групи, додаючи туди будь-яку кількість учасників, зокрема і себе. Для створення самого підпису немає необхідності отримання згоди чи будь якої допомоги від інших учасників групи, так як у процесі формування підпису використовуються лише їхні відкриті ключі разом із власним закритим ключем.

Мультипідпис є схемою електронного підпису, для реалізації якої необхідно T ключів з групи N членів. Сам процес формування підпису відбувається таким чином, кожен член генерує приватний та публічний ключі, визначаються значення параметрів T та N, надалі кожен учасник підписує файл своїм приватним ключем, після чого всі підписи групи об'єднуються у один спільний підпис, щоб перевірити підпис необхідно використати потрібно використати відкритий ключ учасників. Найчастіше такий вид підпису використовується у різноманітних крипто гаманцях.

1.4 Порівняльний аналіз відомих систем електронного документообігу

Так на ринку програмного забезпечення існує величезна кількість різних СЕД, кожна із яких має свої особливості було обрано одні зі найпопулярніших системи із великою кількістю інформації про них у відкритому доступі. Обрано такі українські СЕД: FlyDoc [31], M.E.Doc[32], СОТА "Звітність" [33] та зарубіжні box [34], DocuWare [35] та revverdocs [36]. Кожна із цих систем має свої преваги та недоліки, функціональні особливості, що виділяють їх серед інших.

FlyDoc – це додатковий модуль для виконання обміну електронних документів із контрагентами, який встановлюється безпосередньо у систему для автоматизації бізнесу BAS та аналогів [31]. Цей програмний продукт містить можливість використання засобів електронно цифрових підписів, як для накладання так і перевірки документів, виконує шифрування документів для забезпечення їх конфіденційності. Сам модуль сумісний із багатьма

конфігураціями систем автоматизації бізнесу, можливий обмін із іншими СЕД за допомогою платформи “Птах”, серед усіх названих українських СЕД має найкращу інтеграцію із BAS.

M.E.Doc є найбільш використовуваною СЕД на українському ринку таких систем, її можна вважати універсальним ПЗ для налагодження ЕДО на будь-якому підприємстві. і функціональний [32]. Сервіс має зручний та інтуїтивний інтерфейс із широким функціоналом можливостей, окрім роботи з різними видами звітів, він також дає змогу вести облік заробітної плати, обмінюватися податковими й акцизними накладними, рахунками, актами та договорами. Передбачена можливість створення власних шаблонів первинної документації. У своєму функціоналі має велику кількість модулів для виконання широкого кола задач на підприємстві, за допомогою них є можливість забезпечення розрахунку та нарахування заробітної плати та обліку кадрів, обліку ПДВ, акцизного податку можливість зручної інтеграції із різними обліковими системами тощо.

СОТА Електронний документообіг – сучасним хмарним онлайн сервіс, головна задача якого є виконання потреб малого та середнього бізнесу у задачах, що зв’язані із бухгалтерською звітністю та ведення діловодства. У своєму функціоналі має два варіанта СЕД [33]. СОТА.ЕДО є більш зрубним та легким варіантом ЕДО, має спрощену систему входу для цього потрібно лише КЕП та доступ до інтернету, зручний інтерфейс із набором головних функцій таких систем необхідних для повноцінної роботи. Із іншої сторони СОТА.ЗВІТ має значно ширший функціонал, більшу кількість полів для звітності, детальну фільтрацію і можливість імпортування та експорту будь-яких форматів документів. У своєму функціоналі має можливість роботи із різними податковими накладними та розрахунками користувачів, формує декларації на основі книги доходів, має можливість відправки звітів у податкову і тому подібне. Завдяки тому, що цей програмний продукт є онлайн сервісом, є можливість здачі необхідних форми звітів для контролюючих органів без необхідність встановлення та налаштування програмного забезпечення так як всі необхідні функції знаходяться у веб-браузері. Також можна зауважити, високу функціональну схожість цієї системи із M.E.Doc.

Для більш детального аналізу та кращого розуміння, результати зображено на таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз українських СЕД

№	Назва СЕД	Вартість на рік за користувача	Інтеграція	Функціональні особливості
1	FlyDoc	4650 до 9300 грн	BAS та інш.	Має найкращу інтеграцію із BAS, має весь необхідний функціонал який потрібний для виконання задач на підприємстві.
2	M.E.Doc	Від 1900 до 2150 грн, додаткові модулі придбаються окремо.	BAS та інш.	Для кожного виду користувача має окремі рішення, які будуть необхідні для їх роботи, є можливість придбання додаткових модулів, для забезпечення специфічних потреб.
3	COTA	700 до 3300 грн; додаткові модулі придбаються окремо.	BAS та інш.	Своїми можливостями досить схожа до M.E.Doc, однак має менше функцій.

Кожна із перерахованих СЕД є потужним інструментом для забезпечення ефективного ЕДО на підприємствах, головна відмінить цих систем є їх реалізація, M.E.Doc є десктопним ПЗ для операційної системи Windows, COTA онлайн сервіс із можливістю роботи у веб-браузері, FlyDoc модуль, що встановлюється безпосередньо на BAS або подібні системи. Кожна із систем має інтеграцію із платформою “Птах” для обміну даними.

У зв’язку із тим, що закордоні СЕД не мають можливості інтеграції із системами специфічними для українського ринку, вони будуть розглянуті окремо.

Box – є сучасною онлайн платформою для управління файлами та спільною роботою над ними, має легку інтеграцію із великим переліком різних інструментів таких як Microsoft Office, Salesforce і Google Workspace тощо [34]. Завдяки інтуїтивно зрозумілому інтерфейсу досить легкий в освоєні для людей з будь-якою підготовкою. Надає можливість створення, редагування та кооперативної роботи над різними типами цифрового файлів у реальному часу. Має можливість

автоматизації робочих процесів, розширені засоби контролю безпеки та детальне управління дозволами.

DocuWare – це сучасна хмарна платформа для цифрового управління документами та автоматизації бізнес-процесів [35]. Вона працює як хмарне або локальне рішення і дозволяє організаціям зберігати, впорядковувати, обмінюватися та обробляти документи в електронному вигляді. Виділяється тим, що має змогу інтеграції із понад 500 різними бізнес застосунками, що надає можливість вибору для будь-яких бізнесів найбільш фінансово вигідних та функціональних ПЗ. Має можливість виконання розумного пошуку, за допомогою використання штучного інтелекту та оптичного розпізнання символів, що забезпечує можливість пошуку у сканованих копіях паперових документів [37].

Revverdocs – це хмарна платформа для управління документами, що спеціалізується на автоматизації роботи із документами, забезпечує безпечне зберігання, організацію та спільну роботу над документами, має високий рівень інтеграції із Microsoft 365 [36]. Платформа призначена для поліпшення ефективності бізнес-процесів на підприємстві та управління документами у різних галузях його роботи. Однією із найсильніших сторін цієї системи є її можливості у роботі із автоматизацією та робочими процесами без використання коду, це було досягнуто за допомогою зручного та інтуїтивного інтерфейсу із високими функціональними можливостями, що дозволяє легко та швидко виконувати різні повсякденні виснажливі завдання [38]. Порівняльний аналіз раніше перерахованих систем зображено у таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз закордонних СЕД

№	Назва СЕД	Вартість на рік за користувача	Інтеграція	Функціональні особливості
1	Box	Від 18 до 45 євро	понад 1500 застосунків	Має кращі інструменти одночасної роботи та високим рівнем інтеграції із різними бізнес системами
2	DocuWare	Від 25 до 100 доларів	понад 500 застосунків	Вирізняється потужними інструменти автоматизації робочих процесів
3	Revverdocs	Від 15 до 50 доларів	Microsoft Office, Salesforce, Dropbox, Google Drive	Спрямована на зручність у роботі для користувачів, має потужні інструменти автоматизації без використання коду

Кожна перелічена СЕД у своєму функціоналі має тей чи інший метод звичайного ЦП, однак рішень що стосуються таких методів як ГП, КП чи будяка реалізація порогового підпису не було виявлено.

1.5 Постановка завдання

СЕД є критичною частиною роботи інформаційної інфраструктури будь-якої організації чи підприємства. Такі системи виконують роль централізованого зберігання, обробки, передавання, логування дій із документами, забезпечення безпеки документів. Завдяки цьому підвищується ефективність бізнес-процесів, скорочується час ухвалення рішень і зменшуються ризики втрати, несанкціонованого доступу або фальсифікації інформації.

Аналіз відомих систем показав відсутність у них можливості створення альтернативних до ЦП та електронної печатки засобів. Водночас наявність математичних можливостей для створення ГП так КП дозволяє на їх основі створювати анонімізовані підписи документів, їх верифікацію. Тому постає актуальне завдання розробки СЕД, яка підтримуватиме функцію такого роду підписів, разом із стандартними ЕП.

Також такий застосунок повинен задовольняти базові потреби у роботі СЕД та поліпшені можливості забезпечення безпеки документів та самої системи. Зокрема, система підтримуватиме можливість завантаження та вивантаження файлів, підписання документів за допомогою ЦП, ГП тощо.

1.6 Висновки з розділу

Як результат виконання цього розділу було проаналізовано інформація, необхідна для розробки та вдосконалення системи захищеного електронного документообігу. З’ясовано та засвоєно головні терміни, що стосуються документообігу в загальному, досліджено головні властивості паперового та електронного документообігу, головні задачі використання СЕД. Через те, що сфера ЕД має високу фізичну та інтелектуальну цілісність на неї розповсюджуються певні державні норми та закони, для їх дотримання було проаналізовано перелік українських та закордонних нормативно-правових актів, що регулюють питання електронного документообігу. Далі було проаналізовано найбільш популярні ЕЦ та інші методи підпису документів. Проведено аналіз відомих систем електронного документообігу українського та закордонного зразків, кожна з яких має різний рівень інтеграції із застосунками роботи із документами, можливість автоматизації робочих процесів, метод реалізації таких систем тощо.

2 АРХІТЕКТУРА СИСТЕМИ

2.1 Узагальнена архітектура системи

Узагальнена архітектура модуля для СЕД буде складатися із чотирьох головних блоків, зображено на рисунку 2.1.

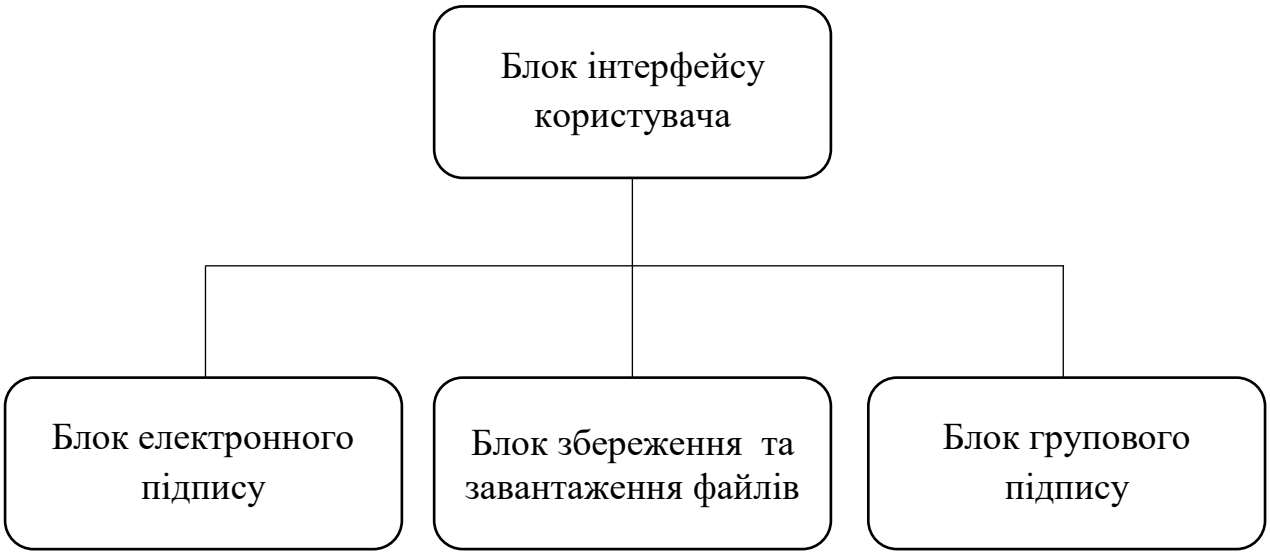


Рисунок 2.1 – Узагальнена архітектура засобу

Як видно із рисунку 2.1 запропонована архітектура засобу має такі блоки:

- блок інтерфейсу користувача, сюди входить графічний інтерфейс усіх функціональних модулів;
- блок електронного підпису відповідає за роботу із звичайним ЕП, включаючи всі необхідні методи для його роботи;
- блок групового підпису відповідає за роботу із ГП та всіма його елементами;
- блок збереження та завантаження файлів виконує роль роботи із файлами у системі.

Згідно із раніше перехованих задач СЕД та проведеного аналізу уже відомих СЕД можна зробити такий висновок, що кожна СЕД для забезпечення основних

потреб користувачів повинна мати певний перелік архітектурних складових, які забезпечуватимуть функціональні особливості СЕД:

- авторизація у систему;
- можливість збереження/завантаження файлів;
- підпис файлу із використанням звичайного ЦП;
- підпис файлу із використанням звичайного ГП;
- доступ до шаблонів документів.

Таким чином для того щоб забезпечити коректну роботу перелічених функцій застосунку необхідно розбити процес розробки на окремі етапи.

Етап 1: провести детальне дослідження тем зв’язаних із раніше названими функціями.

Етап 2: реалізувати початковий варіант графічного інтерфейсу застосунку, а саме сторінки авторизації та головних функцій системи.

Етап 3: провезти аналіз та на його результатах створити модуль для авторизації користувачів.

Етап 4: реалізувати систему для збереження та завантаження документів.

Етап 5: реалізувати можливість завантаження шаблонів документів.

Етап 6: реалізувати можливості підписання документів звичайним ЦП.

Етап 7: реалізувати можливості підписання документів ГП.

Етап 8: реалізувати кінцевий варіант графічного інтерфейсу.

Етап 9: провести тестування та подальше виправлення виявлених помилок.

Оскільки ключовою властивістю цієї системи є груповий підпис доцільно обґрунтувати вибір алгоритму його реалізації.

2.2 Обґрунтування вибору методу групового підпису

Орім ГП у ЦП є певні альтернативні методи для підписання документів, так для кращого розуміння вибору саме ГП для використання у СЕД проаналізовано можливі шляхи використання та функціональні можливості його головних конкурентів КП та мультипідпис.

Підписання за допомогою КП надаю більшу гнучкість у процесі підписання, так як підписант сам обирає групу довірених осіб для створення підпису, та потенційній анонімність особи, що підписує документ. Серед потенційних випадків використання можна виділити анонімне підписання документів із приналежністю до певної групи.

Виходячи із головних властивостей мультипідпису його використання у контексті СЕД зводиться до документу певною групою осіб, без задіяння повної її складу.

Кожен із перерахованих підписів потенційно збільшує швидкість підписання документів при збережені їх ідентичності, однак у порівняні ГП є найшвидшим, так як цей підпис створюється тільки один раз після чого може бути використаний членами групи без необхідності його модифікації чи зміни, окрім випадків збільшення групи. У випадку мультипідпису для підпису залишається потреба взаємодії принаймні мінімальної кількості користувачів, а у випадку КП при зміні складу групи необхідно генерувати новий підпис.

Незважаючи високий потенціал забезпечення анонімності названих підписів, сам факт відсутності можливості ідентифікувати підписанта для СЕД не є позитивним, це спричинене тим, що підписаний такими підписами файл не є легітимним згідно законодавства про електронно цифрові підписи, саме через анонімність підписанта. Проте такі підписи можуть використовуватися у випадках коли документ не повинен нести ніякої юридичної цінності. Як приклад можна навести таку ситуацію, під час перевірки та тестуванні якості написаного програмного коду в організації, під час підписання робочих звітів, було б доречно приховувати особистість людини, що виконувала цю перевірку, задля уникнення упередженості самого підписанта та можливих суперечок серед персоналу підприємства. Також це може бути корисним у випадках проведення певного анонімного голосування або при потребі завантаження файлу у систему із підвищеною безпекою, так як за допомогою цих методів можна підтвердити приналежність підписанта файлу до певної організації [1]. У такому випадку ГП є також більш ефективним, завдяки контролю над групою із сторони адміністратора,

таким чином керування параметрами підпису у СЕД є набагато простішим та ефективнішим. Кількість членів групи завжди є попередньо визначеним, що спрощує процес контролю над групами.

Отже постає необхідність вибору доречної схеми підпису, спираючись на висунуть раніше потенційні приклади використання можна зазначити, що підписання документу ГП не потребує високого рівня захисту та повинен мати коротку довжину підпису та високу загальну оптимізацію його роботи, можливість розкриття підпису є опціонною.

Group Signatures with Selective Linkability [39] – це вдосконалений алгоритм ГП, що покращує традиційні ГП шляхом введення контрольованого зв'язування. Його головна особливість є можливість виявлення приналежності підписів до одного користувача, що дозволяє залишити анонімність користувача однак мати загальне уявлення, що тей чи інший підписант відноситься до двох груп.

Short Group Signatures [40] – це криптографічна схема ГП, оптимізована для зменшення розміру підпису та підвищення ефективності. У своєму функціоналі має можливість відкриття підпису, для ідентифікації підписанта.

Short Randomizable Signatures [41] – це криптографічний алгоритм, призначений для створення цифрових підписів, із можливістю перетворення цього підпису на новий із зберіганням можливості ідентифікації того самого повідомлення. Така "рандомізація" особливо корисна для підвищення конфіденційності, оскільки дозволяє приховати оригінальний підпис, однак залишити можливість його перевірити.

Взявши до уваги наведені вище особливості використання такого типу підписів, для реалізації ГП було обрано схему Short Group Signatures. Завдяки її властивостям відстежуваності, адміністратор групи може визначити особу підписанта, короткі довжині, зазвичай 200 байт, та загальній оптимізації вона найкраще підходить для використання у СЕД.

2.3 Архітектура блоку групового підпису

Схема ГП складається із кроків ініціалізації, генерування необхідних початкових параметрів, генерування ключа, підписання, верифікація та відкриття підпису.

Архітектура блоку підпису скрадатиметься із чотирьох головних процесів: генерування ключа, підписання, верифікації, відкриття підпису. Контроль над процесами створення підпису та відкриття підпису матиме лише адміністратор групи, будь-який користувач може користуватися процесом верифікації та функцією створення підпису. Процес відкриття підпису має бути погоджений із вищим керівництвом або відповідальним відділом організації, що використовує СЕД, лише після цього буде можливе отримання інформації про підписанта повідомлення. Таким чином архітектуру блоку ГП можна розділити на три частини: підписання/верифікація, генерування ключів та відкриття підпису. На рисунку 2.2 наведено блок підписання/верифікації.

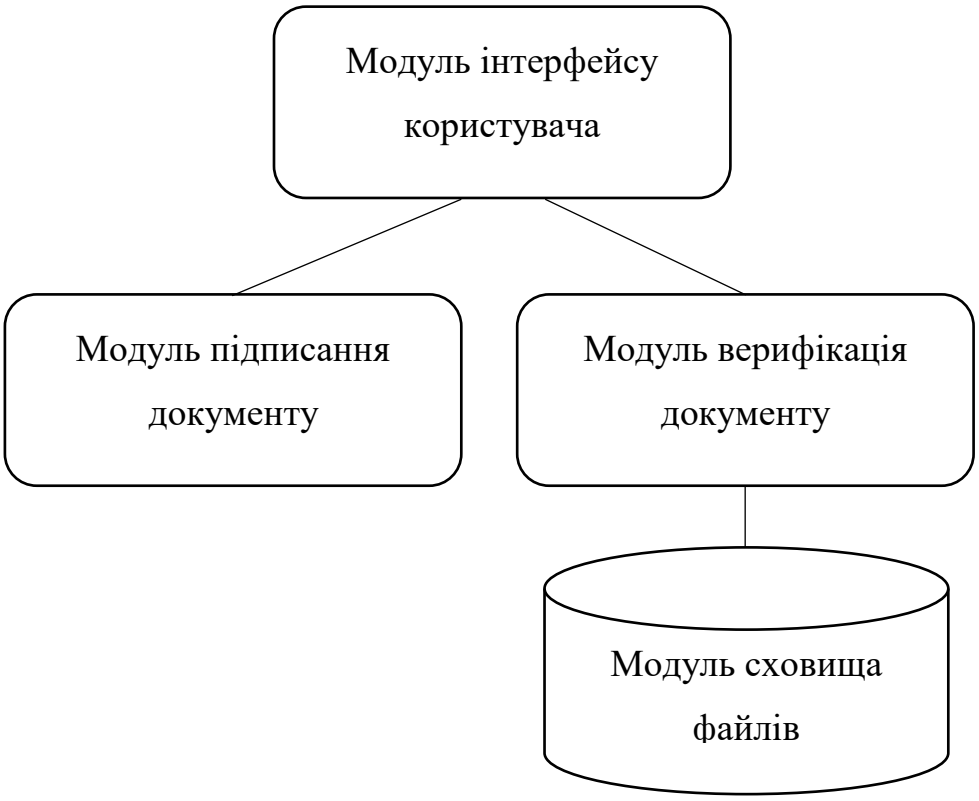


Рисунок 2.2 – Блок підписання/верифікації.

Процес підписання документу починається із взаємодії користувача із модулем графічного інтерфейсу, тобто користувач вибирає необхідну функцію системи, у даному випадку підписання/верифікації документу із використанням ГП, після чого користувач обирає яку дію він хоче виконати. У випадку підписання документу користувач повинен завантажити файл у систему та ввести свій приватний ключ для підпису, після чого проходить процес підписання, на виході користувач отримає підпис файлу, після чого він може або завантажити його на власний пристрій. При роботі із верифікацією документу користувач завантажує файл, підпис цього ж файлу та публічний ключ користувача у систему, після чого проводиться процес верифікації. На виході користувач отримає повідомлення про стан ідентичності необхідного файлу. Далі розроблено блок генерування ключів, зображено на рисунку 2.3.

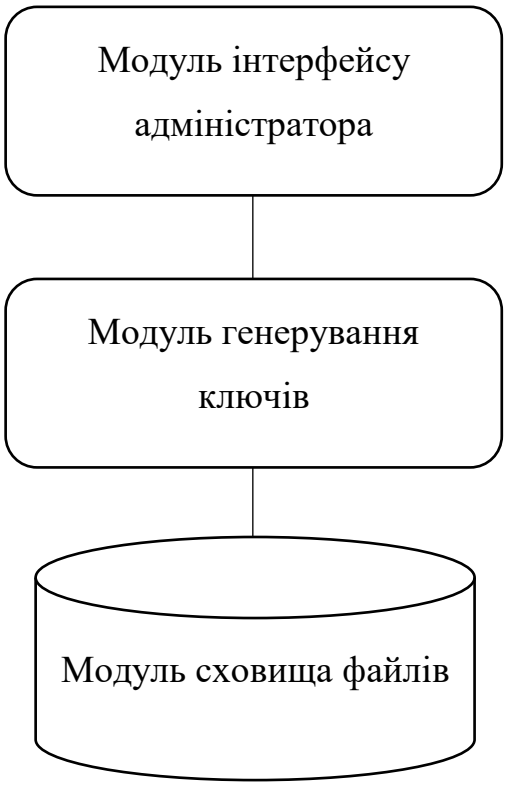


Рисунок 2.3 – Блок генерування ключів.

Генерування ключів виглядає таким чином, група користувачів звертається із проханням із створення ГП, після узгодження всіх необхідних процесів

адміністратор приступає до створення ГП. У результаті буде створено ключів для роботи із підписам, а саме ключ адміністратора, яким буде володіти тільки сам адміністратор, публічний ключ та ключі користувачів, за розповсюдження ключів користувачам має відповідати адміністратор підпису. Розроблено блок розкриття підпису, який наведено на рисунку 2.4.

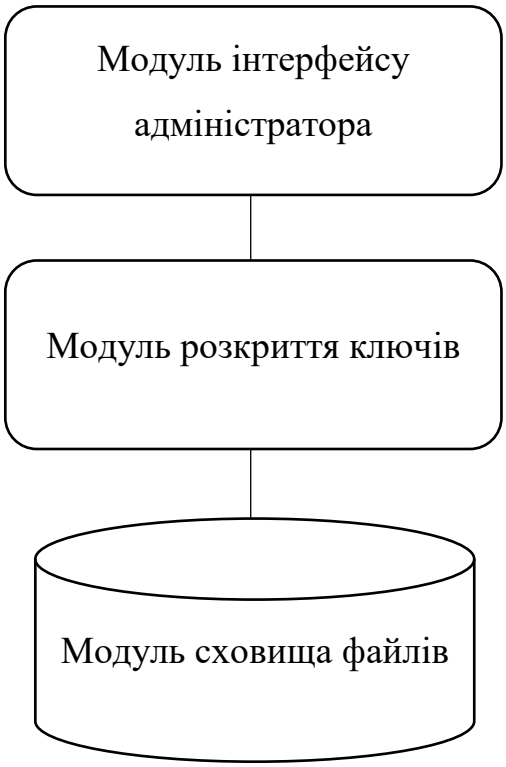


Рисунок 2.4 – Блок розкриття підпису

При необхідності ідентифікації членів групи ГП, керівництво може запросити виконати процес відкриття підпису у адміністратора групи. Адміністратор матиме можливість відкрити підпис, та передати необхідні дані керівництву.

Таким чином можна, при дотримані такої архітектури модуля при розробці рішення можна виконати всі необхідні умови для його роботи та створити функціонал для користування ГП.

2.4 Архітектура блоку авторизації та автентифікації

Для коректної роботи будь-якої СЕД потає необхідність забезпечення функції входу користувача у систему, шляхом проведення автентифікації та подальшої авторизації для надання відповідного рівня доступу. Це дозволить обмежити доступ до конфіденційної інформації та певних функцій системи, покращити рівень безпеки самої системи, надати можливість розподілення прав доступу відповідно до ролей користувачі і надалі вести ефективне журнал подій для подальшого моніторингу. Загальна архітектура цього блоку наведено на рисунку 2.5.

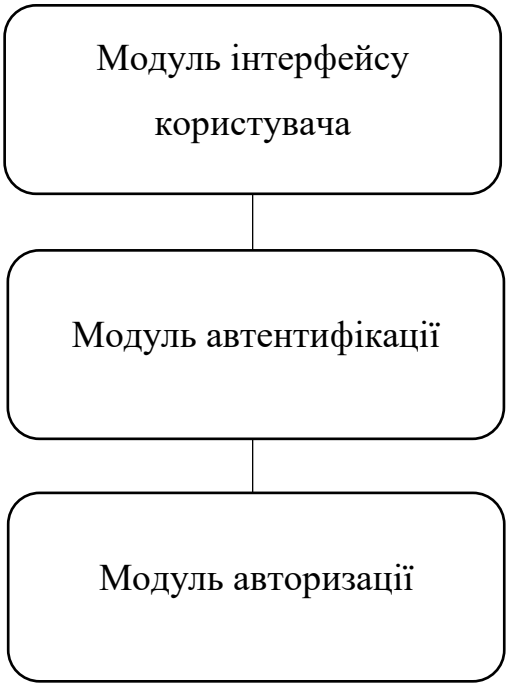


Рисунок 2.5 – Блок авторизації та автентифікації

Відповідно до рисунку 2.5 процес входу у систему відбувається таким чином, користувач переходить на сторінку входу у систему та у відповідних формах вводить свої облікові дані для автентифікації. Далі ці дані надсилаються на сервер системи для перевірки їх справжності. При успішному проходженні перевірки сервер переходить до процесу авторизації та формує токен доступу та передає його для користувача, у протилежному випадку користувач не отримує доступ системи.

Під час авторизації сервер на основі отриманих даних визначає права доступу користувача до ресурсів системи.

2.5 Архітектура блоку збереження та завантаження файлів

Можливість роботи над документами у СЕД є її головною функцією, таким чином постає необхідність забезпечення головних функцій для роботи із документами. Такі інструменти можна розділити на дві головні групи: завантаження та вивантаження файлів із системи, зображено на рисунку 2.6.

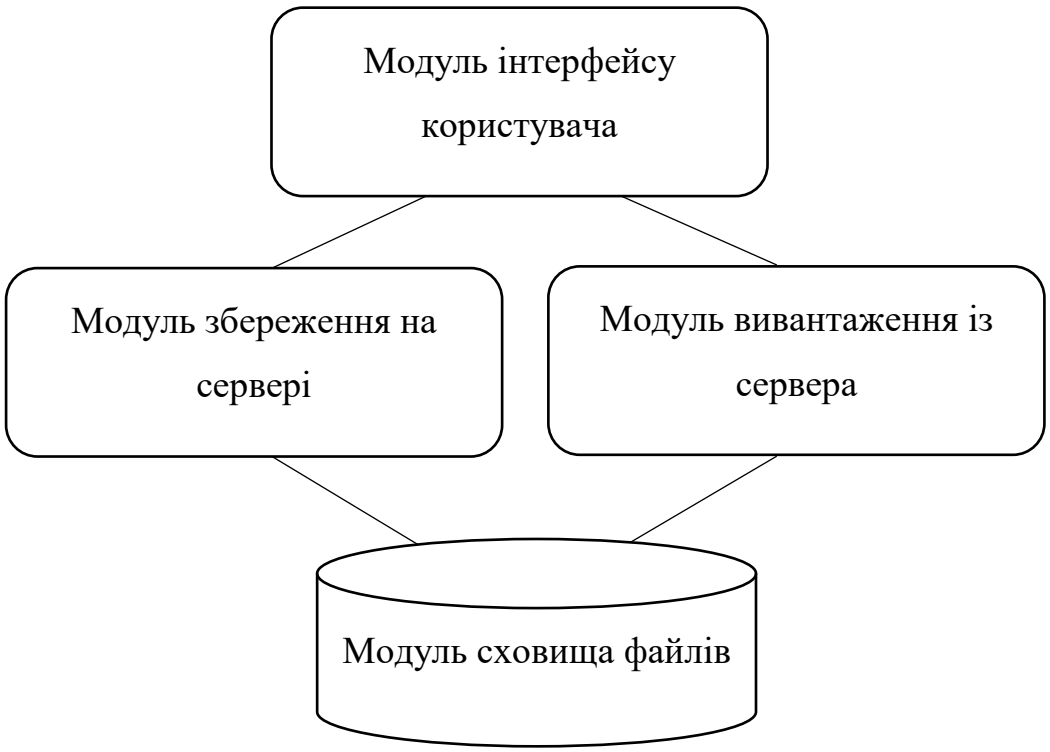


Рисунок 2.6 – Блок збереження та завантаження файлів

Відповідно до рисунку 2.6 користувач матиме такі можливості як завантаження та збереження файлів. Доступ до цих операцій буде відповідати його рівню доступу у системі. Отже спочатку користувач обирає дію яку необхідно йому виконати. При виборі збереження файлу користувач надсилає файл на сервер, після чого відбувається процес збереження файлів у сховищі на сервері. У випадку

вибору завантаження файлу, для користувача буде надано список із всіма файлами до яких він має доступ, після чого він матиме можливість завантажити будь який із цих файлів собі на систему.

2.6 Висновки з розділу

Для реалізації засобу було розроблено узагальнену архітектуру системи, охарактеризовано перелік необхідних для роботи СЕД функцій, для того щоб забезпечити коректну роботу перелічених функцій розділено процес розробки на окремі етапи. Досліджено альтернативи ГП, проведено аналіз їх можливостей стосовно СЕД, на їх основі виконано порівняння із ГП, наведено приклади їх використання, що дало можливість обґрунтувати схему для реалізації ГП. Розроблено архітектуру блоку ГП, вона включає в себе чотири головні процеси та визначено користувачів, які мають до них доступ, сама архітектура складається із трьох частин: підписання/верифікація, генерування ключів та розкриття підпису. Створено архітектуру авторизації та автентифікації для забезпечення функції входу у систему. Розроблено архітектуру блоку роботи із файлами, що надає можливість роботи із файлами у системі. Після розробки архітектури доцільно перейти до розробки алгоритмів роботи.

3 АЛГОРИТМ РОБОТИ СИСТЕМИ

3.1 Узагальнений алгоритм

Узагальнений алгоритм роботи наведений на рисунку 3.1.

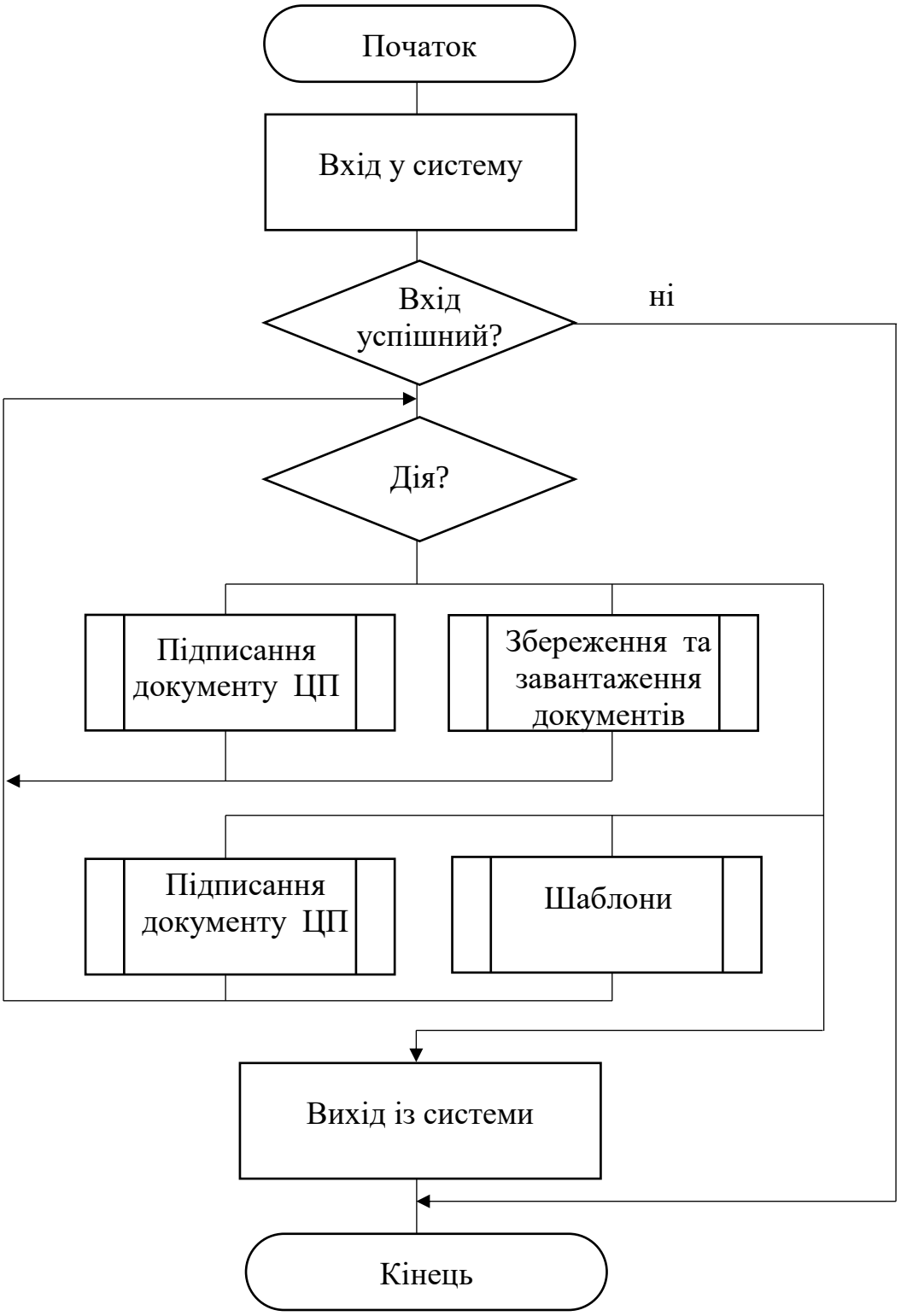


Рисунок 3.1 – Узагальнений алгоритм роботи застосунку

Після побудови всіх необхідних архітектурних блоків системи, враховуючи їх призначення та особливості, було створено узагальнений алгоритм роботи системи, зображено на рисунку 3.1. Його робота починається зі функції входу у системи, після чого, в залежності від його результату, користувачу буде надано доступ до функцій системи або цей доступ буде навпаки відсутній. Далі користувач може використовувати необхідні для його роботи модулі. Модуль підписання документу ЦП відповідає за підписання документів використовую звичайний ЦП. Збереження та завантаження документів відповідає за виконання всіх процесів зв’язаних із завантаження та вивантаження документів у системі. Підписання документу ГП відповідає за підписання документу груповим підписом. Модуль шаблонів відповідає за доступ до шаблонів документів.

3.2 Алгоритм роботи групового підпису

Наступним важливим кроком у виконання цієї роботи є розробка алгоритму роботи ГП, який буде використовувати. Short Group Signatures є загальновідомою схемою ГП, таким чином її алгоритм було розроблено відповідно до оригінальної статті її авторів [40]. Враховуючи ахітектуру ГП він буде складатися із чотирьох головних функцій:

- генерування ключів;
- підпис;
- відкриття підпису;
- верифікація.

Алгоритм генерування ключів зображено на рисунку 3.2.

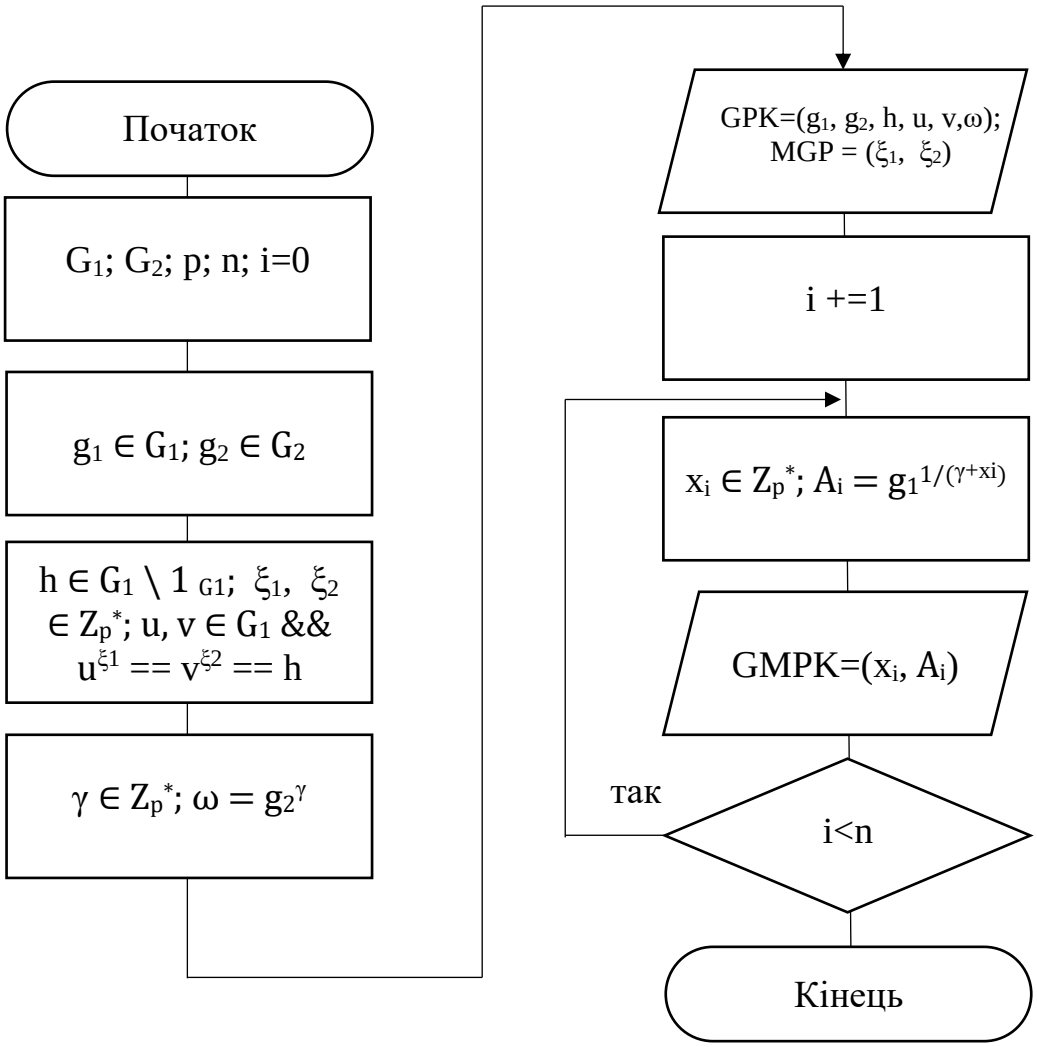


Рисунок 3.2 – Алгоритм генерування ключів

Алгоритм генерування ключів відповідає за процес генерування всіх необхідних ключів для роботи підпису. Сам процес із математичними деталями зображено на рисунку 3.2. Таким чином за його допомогою спершу буде згенеровано ключ адміністратора та публічний ключ групи, у подальшому ці ключі будуть використані для відкриття підпису та верифікації відповідно. Наступним кроком створюються підписи всіх членів групи. Після закінчення процесу генерування, адміністратор підпису розповсюджує приватні ключі до їх власників.

Далі було створено алгоритм підпису документу, зображено на рисунку 3.3.

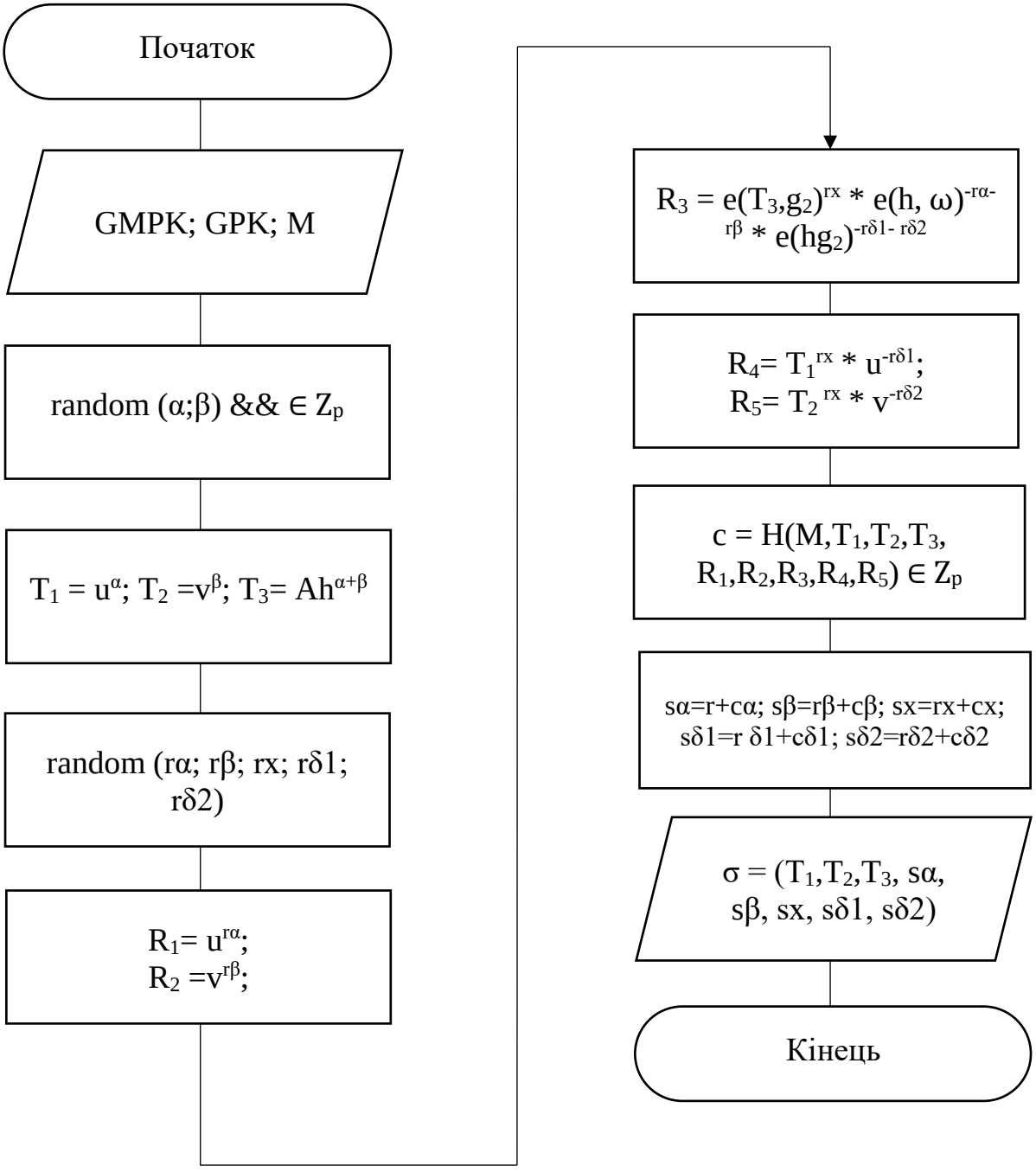


Рисунок 3.3 – Алгоритм підпису документів

Алгоритм підпису, відповідно до його назви, відповідає за підпис документу, на вході він приймає документ для підписання, приватний підпис члена групи, публічний підпис групи. Після чого відповідно до блок-схеми, рисунок 3.3, буде виконано відповідні операції у результаті яких буде отримано підпис.

Наступним кроком було створено алгоритм верифікації документу зображено на рисунку 3.4.

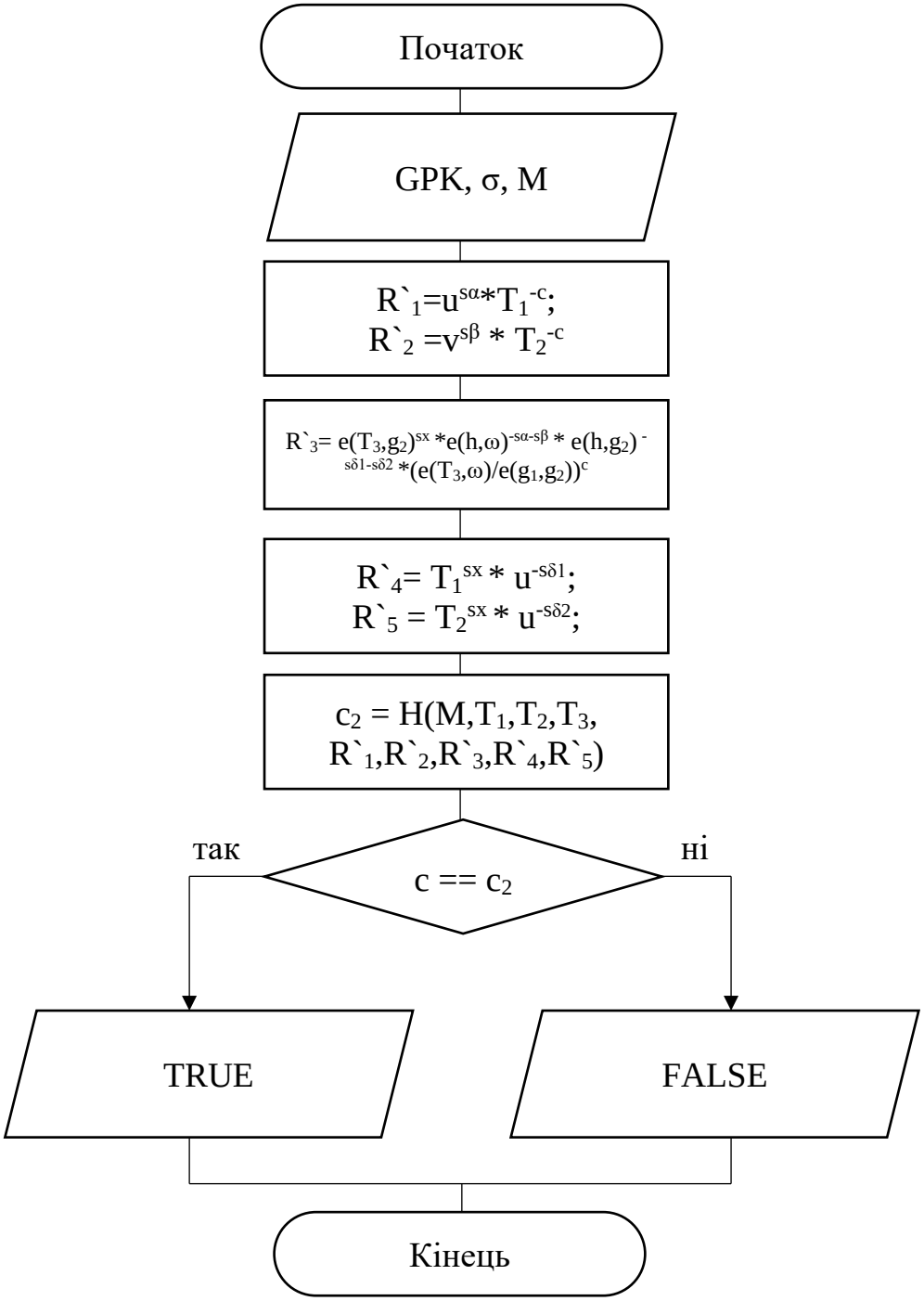


Рисунок 3.4 – Алгоритм верифікації документу

Алгоритм верифікації документу виконує роль ідентифікації документу. На вхід подається підпис, документ, який було підписано, публічний ключ, а на виході буде отримано статус ідентичності файлу.

Реалізовано алгоритм відкриття підпису, зображено на рисунку 3.5.

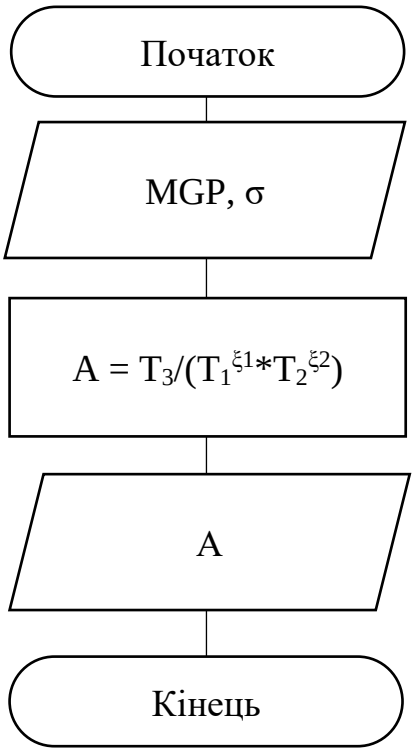


Рисунок 3.5 – Алгоритм відкриття підпису

З допомогою алгоритму, що зображено на рисунку 3.5, описано процес розкриття підпису, що дасть змогу ідентифікувати особу підписанта. Для цього потрібно використати приватний ключ адміністратора та підпис, на виході буде отримано значення A, що є частиною приватного ключа користувача що підписав документ.

Після розробки всіх необхідних алгоритмів для ГП розпочато роботу над розробкою алгоритмів формування електронно цифрового підпису ECDSA.

3.3 Алгоритм формування електронних підписів

Згідно із завдання до бакалаврської кваліфікаційної роботи запроваджено можливість створення ЕП за допомогою методу ECDSA. Для його формування буде використано криву secp256k1 із якої буде використано всі необхідні параметри. Таким чином для подальшої розробки необхідно створити алгоритми для трьох головних його методів :

- створення ключів;
- підпис;
- верифікація.

Алгоритм методу створення ключів зображено на рисунку 3.6.

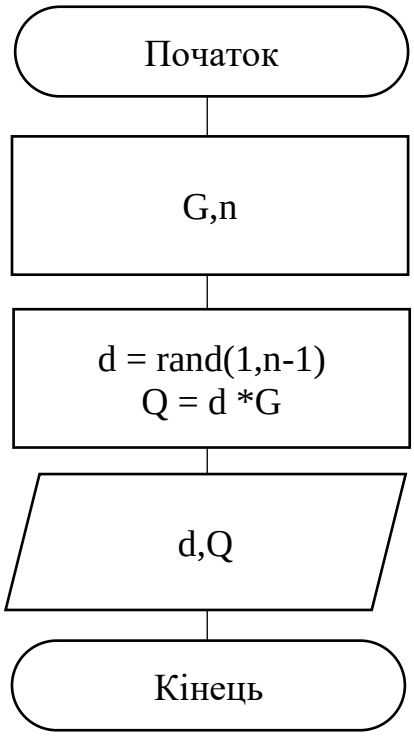


Рисунок 3.6 – Алгоритм створення ключів

Алгоритм створення ключів генерує приватний та відкритий ключі підпису використовуючи точку еліптичної кривої з порядком n.

Далі розроблено алгоритм підпису документу, зображено на рисунку 3.7

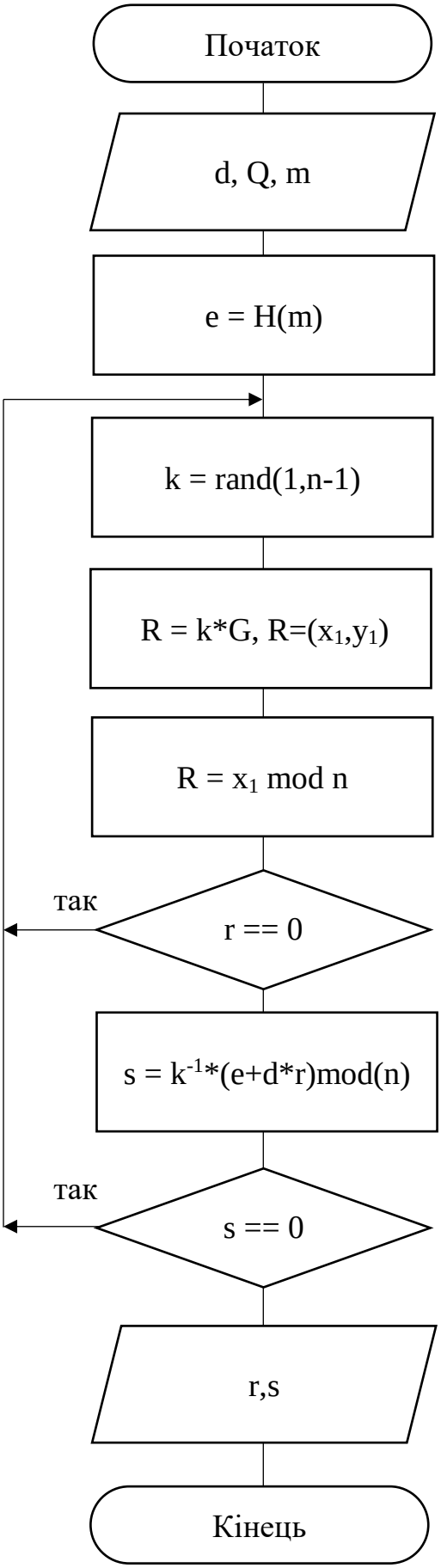


Рисунок 3.7 – Алгоритм підпису документу

На вхід цього алгоритму подається приватний ключ користувача, та повідомлення, після чого виконується процес підпису документу. У результаті буде отримано значення r та s, що будуть слугувати у ролі підпису.

Далі створено алгоритм верифікації підпису (рис. 3.8).

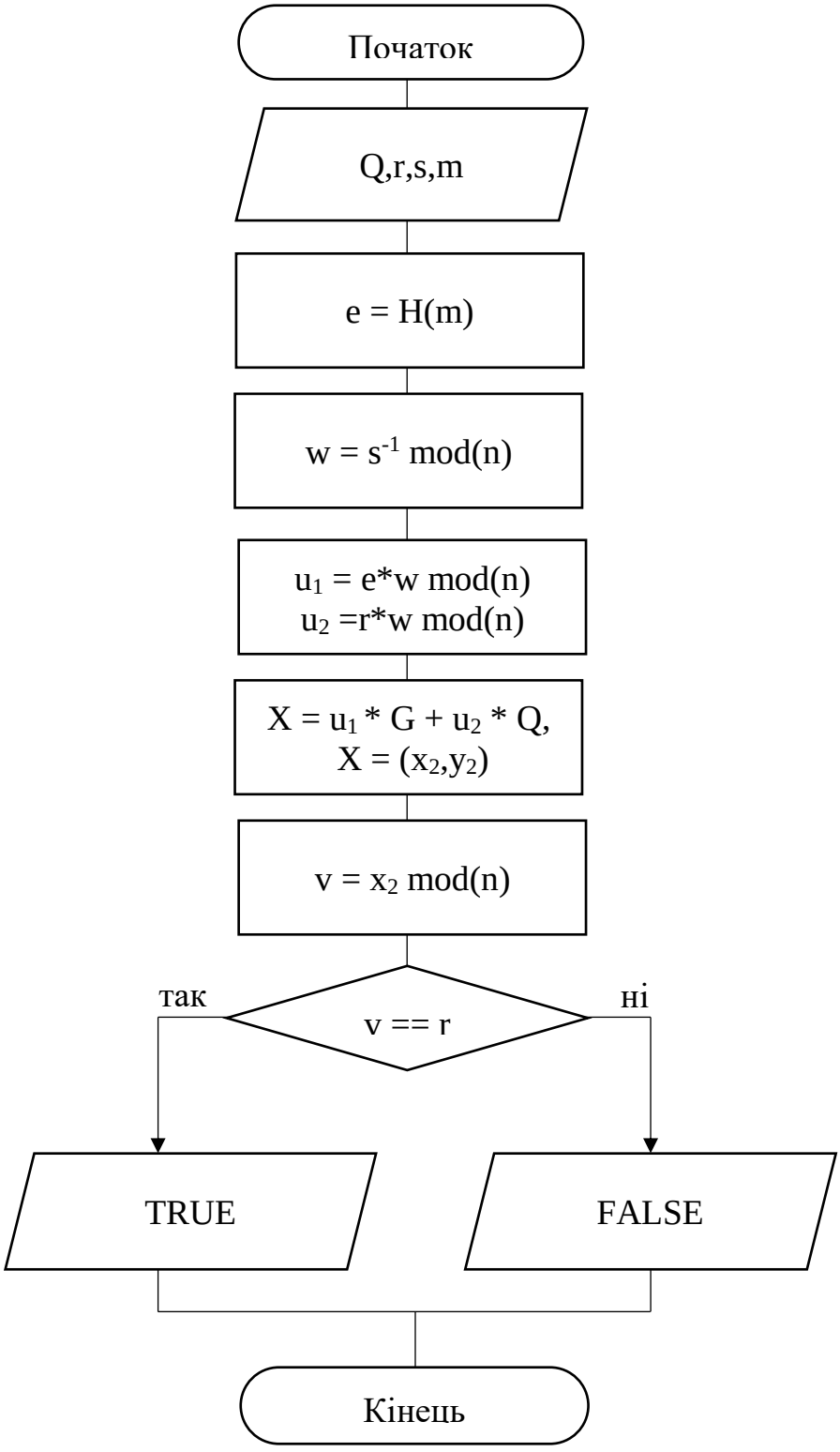


Рисунок 3.8 – Алгоритм верифікації підпису

За допомогою алгоритму верифікації підпису з’явиться змога перевірки ідентичності файлу, на вхід подається файл, публічний ключ та підпис, на виході буде отримано інформацію про перевірку.

3.4 Обґрунтування вибору засобів розробки

Під час підготовки до реалізації СЕД з’явилось питання вибору типу системи для реалізації. Так постав вибір між реалізацією у вигляді програмного застосунку або веб-застосунку.

У випадку програмного застосунку система може бути реалізована у вигляді окремого застосунку, що буде встановлено та буде використовуватися безпосередньо на комп’ютері користувача. Одна у такому випадку постає проблема управління централізованим управлінням даних, оновлення програмного забезпечення, підтримки для різних операційних систем та забезпечення синхронізації між кількома користувачами. Окрім того потрібно враховувати, що система потребує централізованого сховища на сервері, що вимагатиме реалізації певних додаткових рішень.

Натомість у випадку веб-застосунку користувач взаємодіє зі системою через браузер, що відкидає потребу у забезпечення підтримки для різних операційних систем, а основна логіка застосунку та збереження даних виконується на стороні сервера. Таким чином це полегшує доступ до централізованого сховища, забезпечує зручну підтримку й оновлення, дозволяє організувати рольову модель доступу. Також такий підхід є більш гнучким у розгортанні, зокрема в хмарному середовищі або у внутрішній корпоративній мережі.

Таким чином ця система буде розроблена у вигляді веб-застосунку.

Наступник кроком потрібно обрати мову програмування для розробки серверної частини системи, для цього чудово можуть підійти такі мови як Java та C#. Обидві мови у своєму функціоналі маю потужні фреймворки для реалізації веб-застосунків ASP.NET [42] у випадку C# та Spring [43] у випадку Java, окрім цього для реалізації ГП є необхідність у використанні бібліотеки для роботи із

криптографією на основі математичних сполучень [44]. У випадку мови C# бібліотек, що виконують необхідну функцію не було знайдено, а у випадку мови Java знайдено бібліотеку JPBC [45]. Так для розробки буде використано мова Java.

Окрім ГП у роботі СЕД є потреба створення ЦП, у даному випадку підпису ECDSA, для впровадження цього функціоналу потрібно обрати підходящу криптографічну бібліотеку, серед усіх варіантів виділяються Bouncy Castle та JCA. Вибір підходящої системи був досить легким тому, що у порівняні із JCA Bouncy Castle підтримує суттєво більше криптографічних алгоритмів, що значно полегшує майбутнє вдосконалення системи.

Для розробки графічної частини сайту буде використано HTML, CSS та JavaScript.

Для ролі інтегрованого середовище розробки є два головних кандидата: IntelliJ IDEA 2024.2.3 [46] та Eclipse [47]. Кожне із цих середовищ безумовно є потужними інструментами для роботи з кодом, із своїми перевагами та недоліками. Із головних переваг Eclipse можна виділити кращим компілятором коду, має вбудований ініціалізатор проектів на Spring, саме середовище є повністю безкоштовним із відкритим кодом, однак має застарілий та деколи перевантажений інтерфейс, має вбудовану підтримку тільки мови Java, використання інших мов програмування потребує завантаження додаткових плагінів. Із іншої сторони IntelliJ IDEA має кращий інтерфейс користувача та редактор коду, має більшу низку плагінів, без додаткових розширень підтримує такі мови як Java, Groovy та Kotlin, в загальному має кращу надійність під час роботи, однак головним її мінусом є те, що це середовище має безкоштовну та платну версії застосунку, що досить сильно відрізняються у своєму функціоналі. Таким чином для розробки веб-застосунку обрано IntelliJ IDEA, через більшу зручність використанні та більшої кількості вже вбудованих засобів для розробки. Окрім цього можна зауважити, що IntelliJ IDEA у своєму функціоналі має можливість розумного автозаповнення та аналізу коду, потужного засобу рефакторингу, вбудованої підтримки популярних технологій для розробки різних додатків, вбудованої підтримку Git та систем контрою збірки Gradle та Maven, сучасного та зручного інтерфейсу, автоматизації

рутиних задач (підсвічування помилок у реальному часі, генерування коду, шаблони, автоматичне формування проєкту тощо), працює не тільки з Java, а й з Kotlin, Groovy, Scala, JavaScript, SQL, Python тощо.

У ролі системи контролю збірки поміж Gradle [47] та Maven [48] було обрано саме Maven. Головною причиною такого рішення є те, що саме ця система є більш зручнішою та простішою у використані, звісно у порівняні швидкості роботи Gradle є більш швидким, однак у випадку розробки цього проєкту ця властивість не матиме великого впливу.

Після вибору необхідних засобів можна приступати до безпосередньої розробки веб-застосунку та подальшого його тестування.

3.5 Висновки з розділу

Як висновок, у процесі цього розділу було розроблено усі необхідні алгоритми для коректної роботи системи. Самого початку було розроблено узагальнений алгоритм роботи системи, на основі якого, буде розроблено СЕД. Наступник кроком створено алгоритм роботи ГП, він складається із генерування ключів, підпису документа, його верифікації та відкриття підпису, таким чином реалізовано всі функції підпису, які необхідні для його використання. Розроблено алгоритм для підписання звичайним ЦП, у ролі підпису виступає ECDSA із кривою secp256k1, сам алгоритм складається із 3 частин, створення ключів, підписання та верифікації. Далі обрано та обґрунтовано засоби, що будуть використані під час розробки. Так буде розроблено веб-застосунок, використовуючи мову програмування Java та Spring Framework та бібліотеки JPBC та Bouncy Castle. Для розробки графічної частини буде використано HTML, CSS та JavaScript. Для написання коду обрано середовище розробки IntelliJ IDEA. Для контролю збірки було обрано Maven.

4 ТЕСТУВАННЯ СИСТЕМИ

4.1 Тестове середовище

Для проведення якісного тестування створеної системи спочатку потрібно охарактеризувати тестове середовище. Воно включає у себе проведення модульного, статично та інтеграційного тестування. Основними інструментами для тестування виступатимуть JUnit 5 [50] та PMD [51].

JUnit 5 – це фреймворк для проведення модульного тестування він складається із 3 головних компонентів: JUnit Platform, відповідає за запуск тестів та інеграцію із іншими засобами розробки, JUnit Jupiter – основний API для написання тестів, JUnit Vintage – модуль сумісності, що відповідає за роботу із старими версіями тестів.

PMD – інструмент для статичного аналізу коду, який перевіряє тестовану програму на помилки, недоліки стилю коду, дублювання коду та можливі баги без прямого виконання коду.

Для проведення інтеграційного тестування буде використано браузер Google Chrome, локальний веб сервер буде розміщено на 8081 порті.

Всі файли ключів буде згенеровано у процесі тестування, документ який буде використано для необхідних перевірок матиме назву “1.pdf” та за заповнений випадковими даними.

У ролі інтегрованого середовище розробки обрано IntelliJ IDEA 2024.2.3.

Наступними кроками буде проведено модульне та інтеграційнн тестування системи, а також статичне тестування безпеки.

4.2 Модульне тестування

Для того щоб впевнитися коректній роботі системи постає необхідні проведення модульного тестування. Сама перевірка буде виконана за допомогою модулю JUnit 5. Для проведення тестування обрано шість найбільш критичних класів, а саме класи, що забезпечують роботу із ГП та підписом ECDSA,

контролери за допомогою яких виконуєтесь керування відповідними методами цих класів та класів контролера та сервісу завантаження/вивантаження файлів.

Тестування розпочато із класу ГП. Спочатку перевірено генерування публічних, приватних ключів та ключа адміністратора, перевірено чи не генеруються пусті величини. Далі перевірено правильність підписання та верифікації фалів, було створено всі необхідні змінні та підставлено у відповідні методи класів. Останнім тестом виконано перевірку правильність відкриття підпису, значення отримане під час відкриття підпису має співпадати із частиною приватного ключа. Результат тестування наведено на рисунку 4.1.

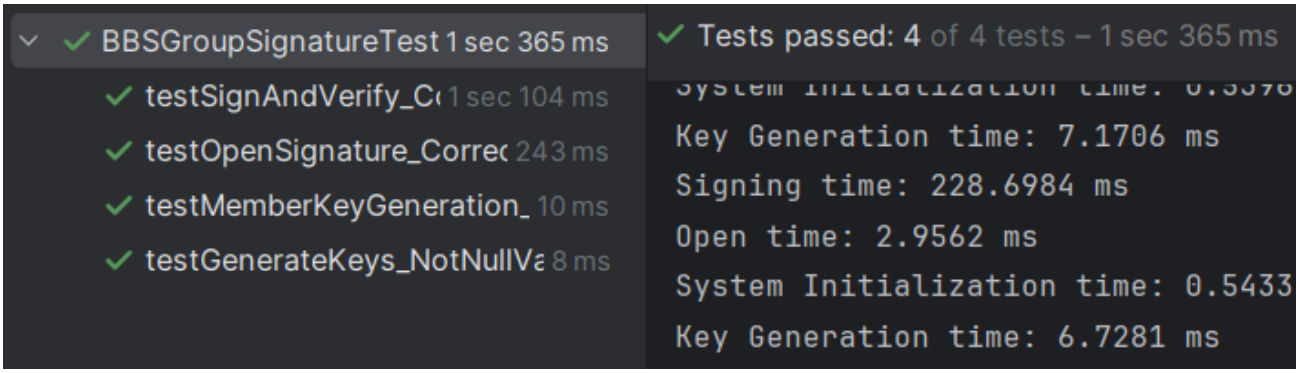


Рисунок 4.1 – Результат тестування класу ГП

Наступним кроком виконано тестування класу, що відповідає за роботу ECDSA. Спочатку перевірено підписання файлу із правильним підписом, на виході не має бути null, пустого значення чи значення false. Далі виконано перевірку верифікації підпису із використанням модифікованого документу, у такому випадку у результаті має бути отримане значення false. На останок створено тест, який перевіряє роботу верифікації із хибним ключем, так як і у минулому тесті для проходження тесту потрібно отримати значення false. По завершені тестування отримано такі результати, що зображено на рисунку 4.2.

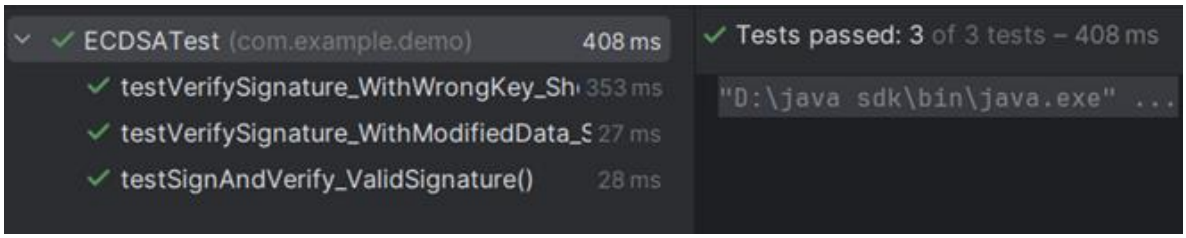


Рисунок 4.2 – Результат тестування класу ECDSA

Далі створено перевірку для класу-контролера роботи із ГП. Так як у цьому класі міститься певні функції, що відповідають за роботу із файлами, перевірено роботу при умові, що файл не знайдено. Для генерування ключів потрібно кількість членів групи ГП, таким чино перевірено випадки, коли вводиться правильне, число більше 10 та менше 50 включно, та не правильне число. Далі створено тест для перевірки повернення правильного статусу та повідомлення, при успішному збереженні файлу. Наступний тест буде навпаки перевіряти чи контролер обробляє виключення й повертає коректну помилку при помилці під час збереження. Після проведення тестування отримано результат, що зображено на рисунку 4.3.

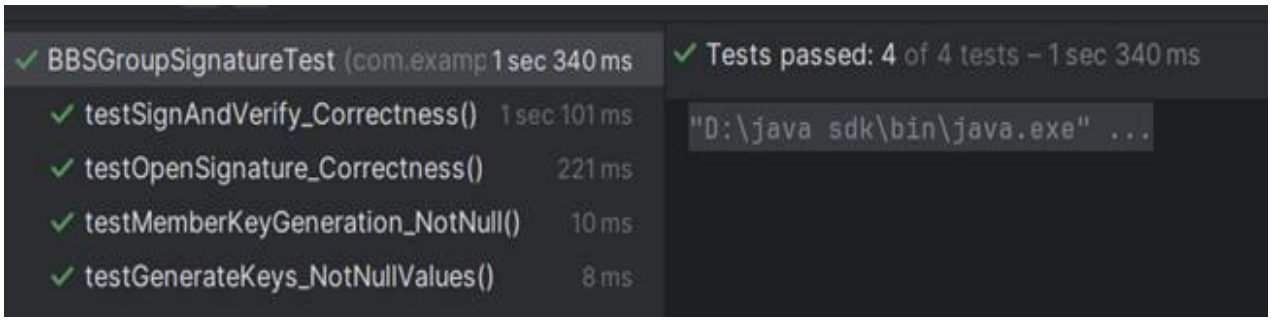


Рисунок 4.3 – Результат тестування класу-контролера ГП

Створено тести для класу-контролера роботи із ECDSA. Було створено тести на перевірку роботи із файлами, а саме на завантаження їх у систему, перевірено коректність при правильному завантаженні та коли виникають певні помилки. Створено перевірки відвантаження фалів генерування ключів та підписання документу. Протестовано чи відбувається відправка коректних значень при перевірці підпису. Результати тестування зображено на рисунку 4.4.

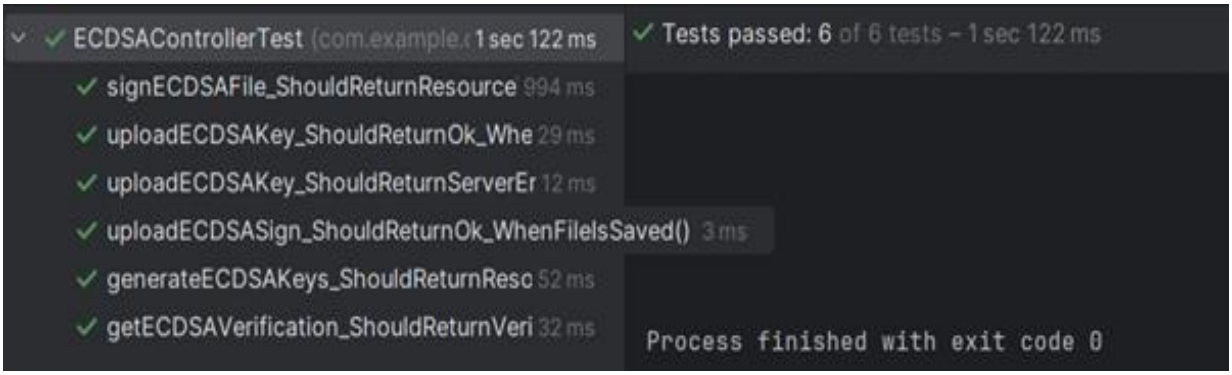


Рисунок 4.4 – Результат тестування класу-контролера ECDSA

Протестовано клас, що відповідає за методи для завантаження та вивантаження файлів. Створено тести для перевірки успішного збереження та отримання файлів та перевірка правильності викликання винятків при передачі null замість файлу. Результат перевірки зображено на рисунку 4.5.

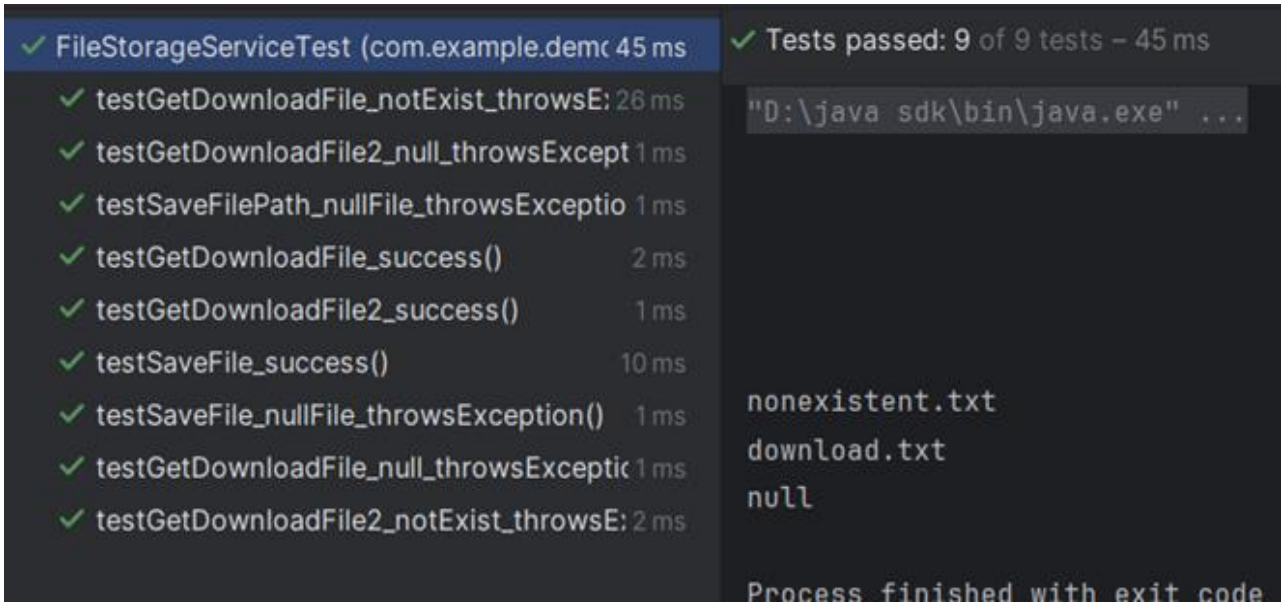


Рисунок 4.5 – Результат тестування класу завантаження/відвантаження файлів

Останнім було розроблено тести для контролера завантаження та відвантаження файлів. Тести виконують функцію перевірки роботи збереження та відвантаження при коректній роботі та при помилках. Таким чином результат тестування зображено на рисунку 4.6.

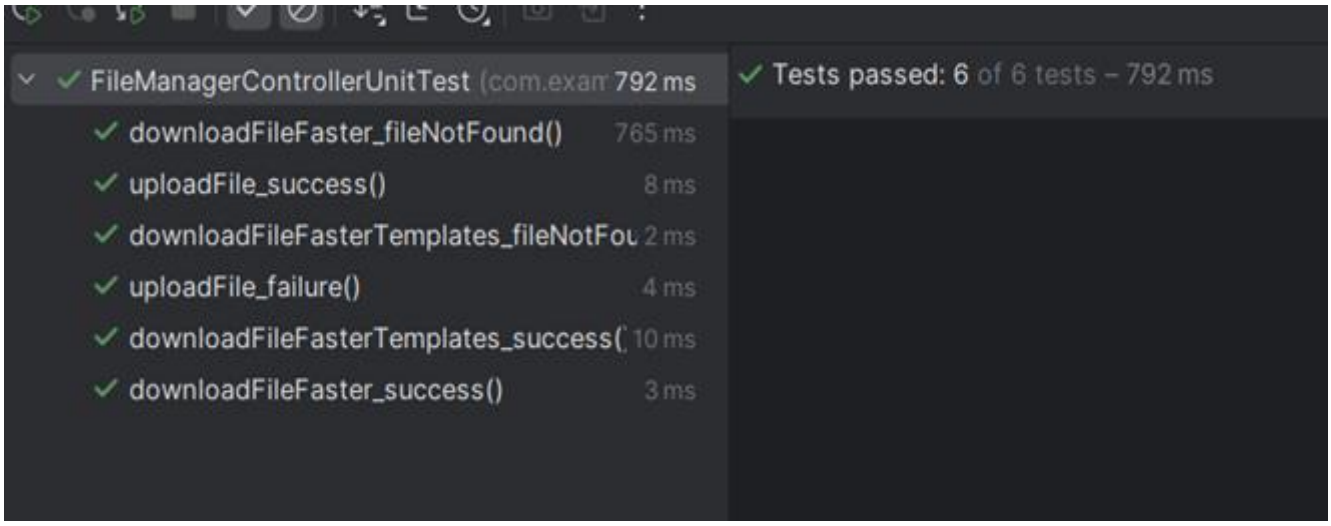


Рисунок 4.6 – Результат тестування класу-контролера завантаження/відвантаження

У результаті модульного тестування, було підтверджено, що всі методи класів працюють коректним чином і не потребують певних критичних змін у своїй роботі. Наступний кроком процесу тестування сервісу буде статичне тестування, виконане у наступному розділі.

4.3 Статичне тестування безпеки системи

Наступним кроком у тестуванні системи буде проведення статичного тестування. Для виконання такої задачі було вирішено використовувати такі інструменти як PMD, він був обраний завдяки свої інтеграції із середовища розроби у вигляді плагіну та його функціоналом у питаннях тестуваня коду.

Спершу проведено тестування засобами PMD, для цього потрібно відкрити командний рядок у середовищі розробки, що використовується та виконати таку команду “mvn pmd:check”, після чого запуститься перевірка коду. Результат перевірки зображено на рисунку 4.7.

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.624 s
[INFO] Finished at: 2025-06-10T08:13:16+03:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-pmd-plugin:3.26.0:chec
k (default-cli) on project demo: PMD 7.7.0 has found 21 violations. For more details
 see: D:\java bach project\main aplication\real\demo\target\pmd.xml -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read th
```

Рисунок 4.7 – Результат тестування коду за допомогою PMD

Як видно із результатів тестування виявлено 21 порушення. Для більш детального дослідження помилок в коді відкрито файл “pmd.xml”, як вказано на отриманій відповіді. Ця інформація зображена на рисунку 4.8.

```
<file name="D:\java bach project\main aplication\real\demo\src\main\java\com\example\demo\ECDSA\E
<violation beginline="5" endline="5" begincolumn="1" endcolumn="24" rule="UnnecessaryImport" rule
Unused import 'java.security.*'
</violation>
</file>
<file name="D:\java bach project\main aplication\real\demo\src\main\java\com\example\demo\ECDSA\E
<violation beginline="20" endline="20" begincolumn="1" endcolumn="18" rule="UnnecessaryImport" ru
Unused import 'java.io.*'
</violation>
<violation beginline="25" endline="25" begincolumn="1" endcolumn="24" rule="UnnecessaryImport" ru
Unused import 'java.security.*'
</violation>
<violation beginline="137" endline="137" begincolumn="52" endcolumn="99" rule="UselessParentheses
Useless parentheses.
```

Рисунок 4.8 – Вміст файлу “pmd.xml”

Після детального аналізу файлу, що було згадано на рисунку 4.8, було з’ясовано, що переважна більшість помилок не несе високої критичності тому, що велика їх кількість пов’язана із імпортом пакетів, що не використовувались у коді, певних частин коду, що не несуть ніяких змін у логіку роботи програми та присутності змінних, що не використовуються. Так згідно до отриманих результатів тестування було виправлено вказані помилки у коді. Однак під час процесу виправлення, виявлення не всі зазначені негаразди у коді є реальними помилками, так як ці частини коду є необхідними для його виконання. Повторне проведення тестування зображено на рисунку 4.9.

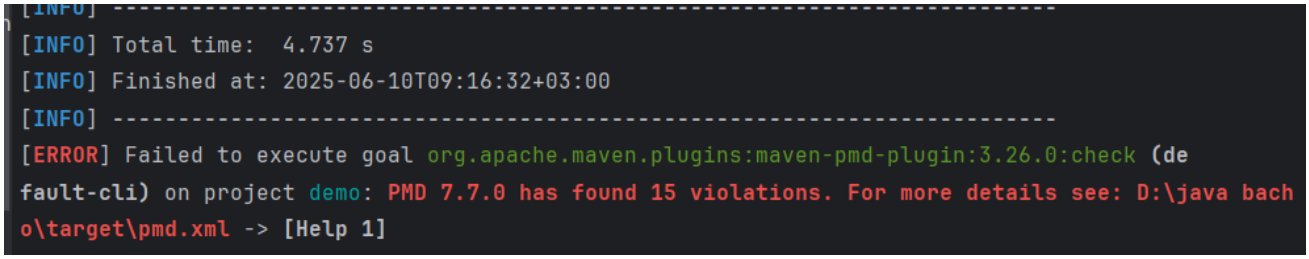


Рисунок 4.9 – Повторний результат тестування коду за допомогою PMD

Як видно із рисунку 4.9 кількість порушень було зменшено до 15, лишилися тільки ті, що було визначені помилково хибними.

Після проведення статичного тестування можна переходити до інтеграційного тестування.

4.4 Інтеграційне тестування модуля

Інтеграційне тестування буде проведено із двох головних частин: тестування функцій адміністрування, за допомогою них також буде згенеровано необхідні значення для подальшого тестування, та тестування функцій звичайного користувача.

Тестування розпочинається із входу у систему, спочатку проведено вхід як адміністратор, для цього введено значення “Username” та “Password”, вікно входу виглядає таким чином, рисунок 4.10

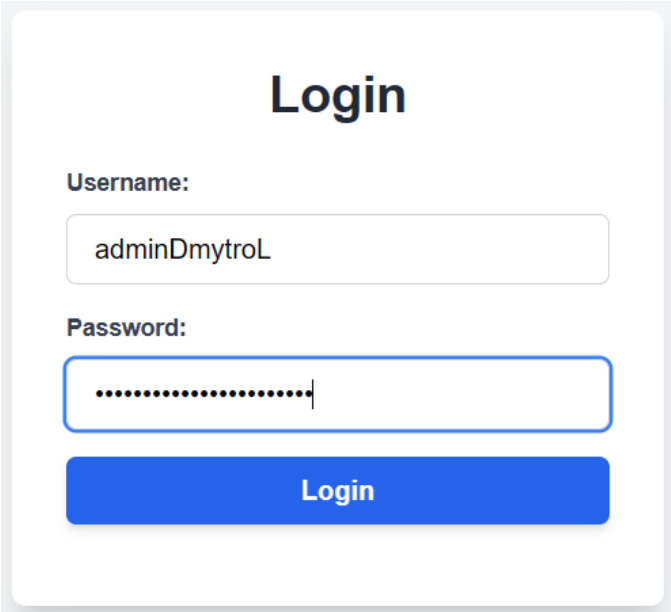


Рисунок 4.10 – Зображення вікна входу у систему

При неправильному вході у систему вікно матиме такий вигляд, рисунок 4.11

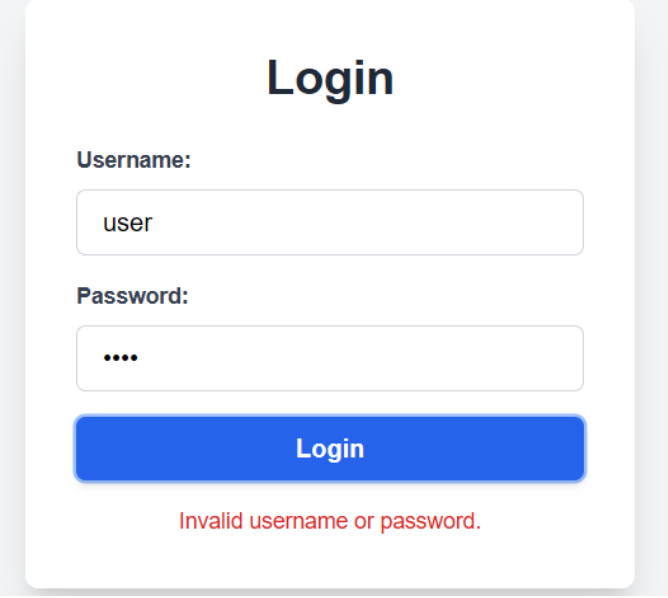


Рисунок 4.11 – Зображення відхилення доступу

Після успішного входу у системи у ролі адміністратора одразу буде відкрите вікно “Generate SGS Key”. Із наповнення воно має можливість навігації між функціями системи за допомогою меню в шапці сайту, а саме такими функціями як

“Generate SGS Key” – генерування ключів ГП, “Open SGS” – відкриття ГП, “Generate ECDSA Key” – генерування ECDSA ключів, та відповідні форми для виконання тих чи інших дій. Меню буде однаковим для всіх сторінок, що доступні адміністратору. Вигляд сторінки зображено на рисунку 4.12.

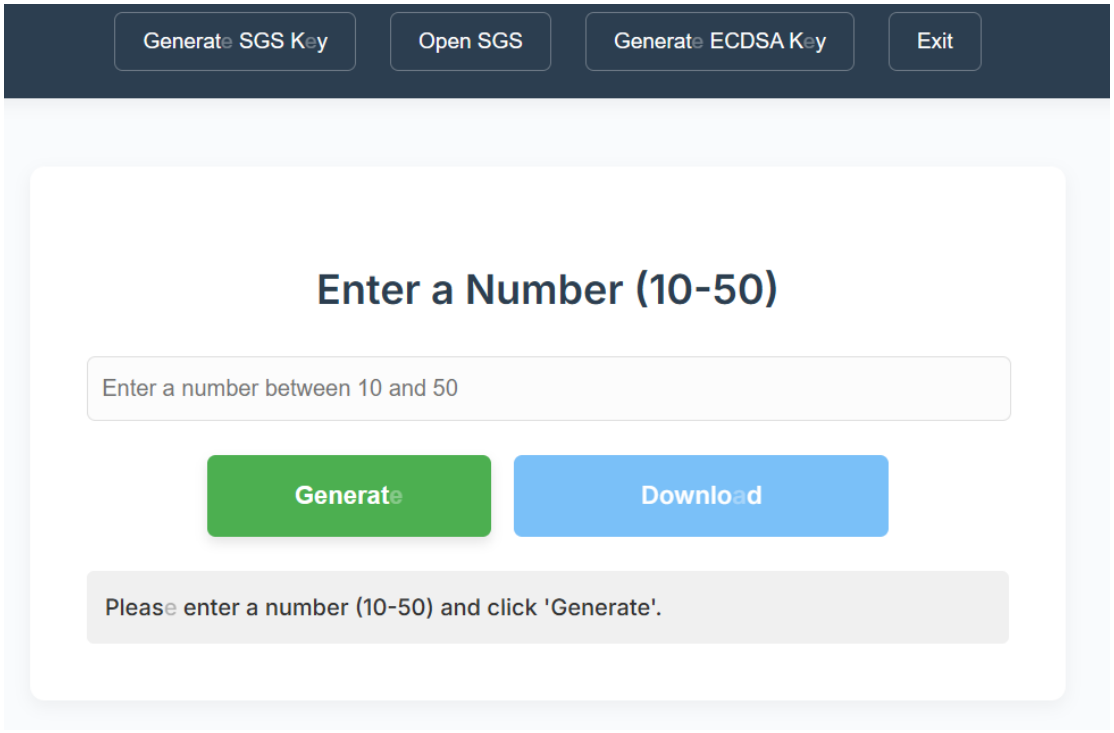


Рисунок 4.12 – Вигляд вікна “Generate SGS Key”

Далі буде виконано генерування ключів для ГП, для цього потрібно заповни всі необхідні форми, у даному випадку ввести кількість членів групи. Після чого натиснути кнопку “Generate” та кнопку “Download” для того щоб завантажити ключі, завантажений файл зображено на рисунку 4.13.

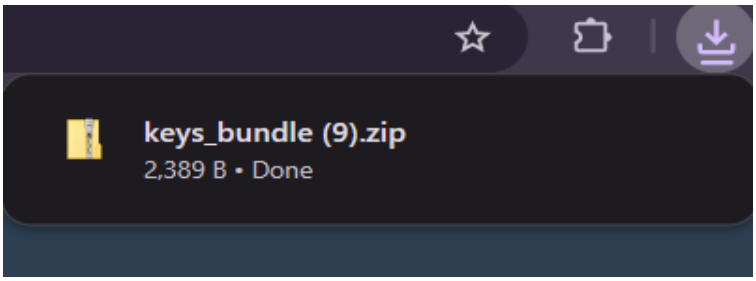


Рисунок 4.13 – Завантажений файл із ключами

Цей файл містить всі необхідні ключі для роботи ГП, вміст файлу зображено на рисунку 4.14

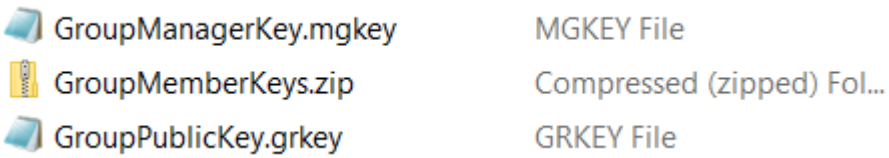


Рисунок 4.14 – Вміст файлу із ключами

При хибному введені кількості учасників групи буде отримано такий результат, рисунок 4.15.

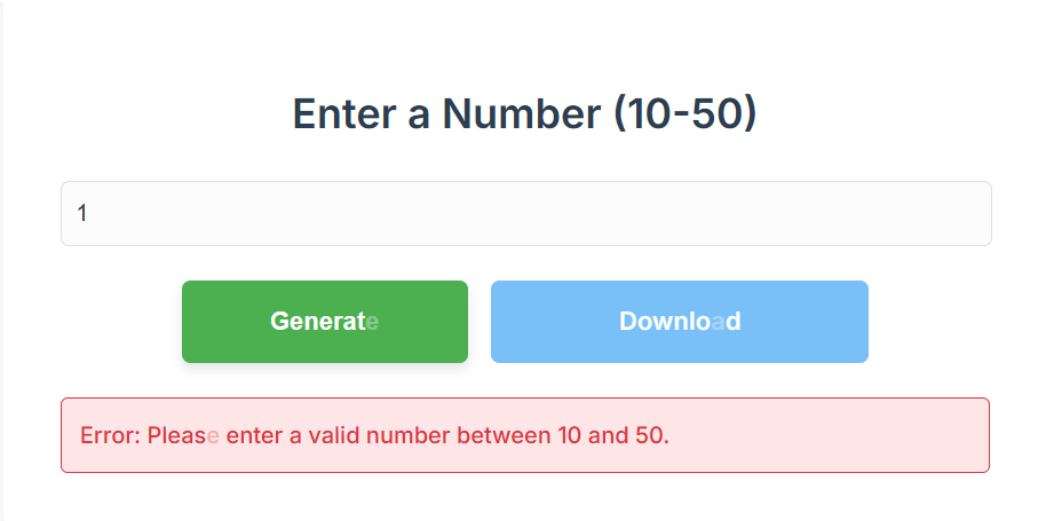


Рисунок 4.15 – Вигляд помилки про хибне введення даних у форму

Далі проведено тестування функції генерування ключів для ECDSA, для цього обрано пункт у меню “Generate ECDSA Key”. У випадку цієї сторінки для створення ключів потрібно натиснути кнопку “Generate ECDSA Keys”, після чого ключі будуть згенеровані та завантажені на комп’ютер у вигляді zip-файлу, зображено на рисунку 4.16.

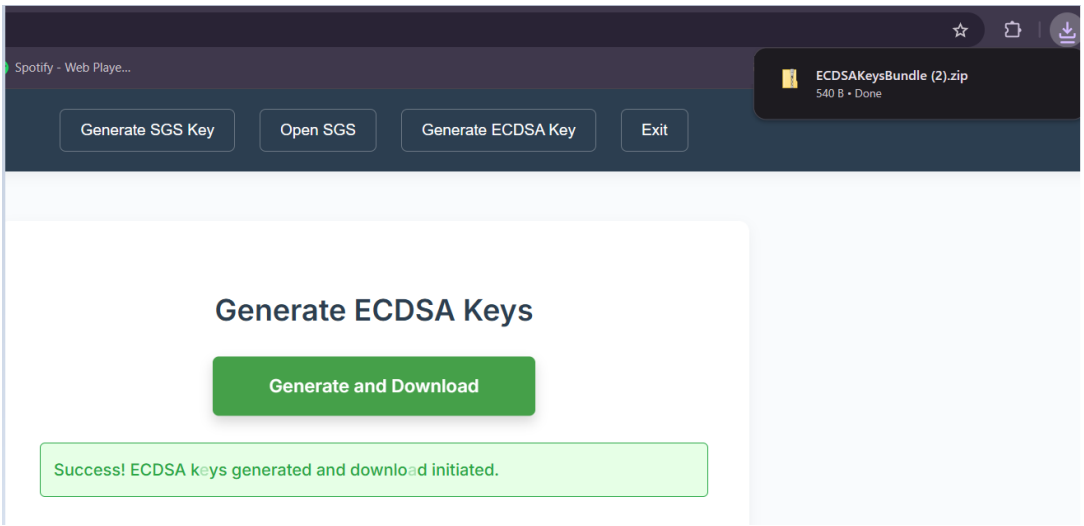


Рисунок 4.16 – Вигляд вікна генерування ключів для ECDSA

Після генерування всіх необхідних ключів можна розпочати тестування функцій доступних для користувача, однак для цього спочатку потрібно зайти від імені користувача у систему. Для цього потрібно натиснути кнопку “Exit” після чого відкриється вікно входу та ввести необхідні параметри входу для користувача.

Кожна сторінка, до якої має доступ користувач, має меню що має такі пункти: “File Download” – вивантаження файлів із системи, “File Upload” – завантаження файлів у систему, “Group Sign” – підписання використовуючи ГП, “Group Verify” – перевірка підписаного ГП файлу, “Templates” – завантаження шаблонів, “Sign” – підписання із використанням ECDSA, “Verify” – перевірка підписаного ECDSA файлу, “Exit” – вихід із системи.

При успішному вході у систему одразу відкриється вікно “File Upload”, для тестування цієї функції обрано файл “1.pdf” та натиснуто на кнопку “Upload File” після чого файл завантажиться у систему, зображено на рисунку 4.17.

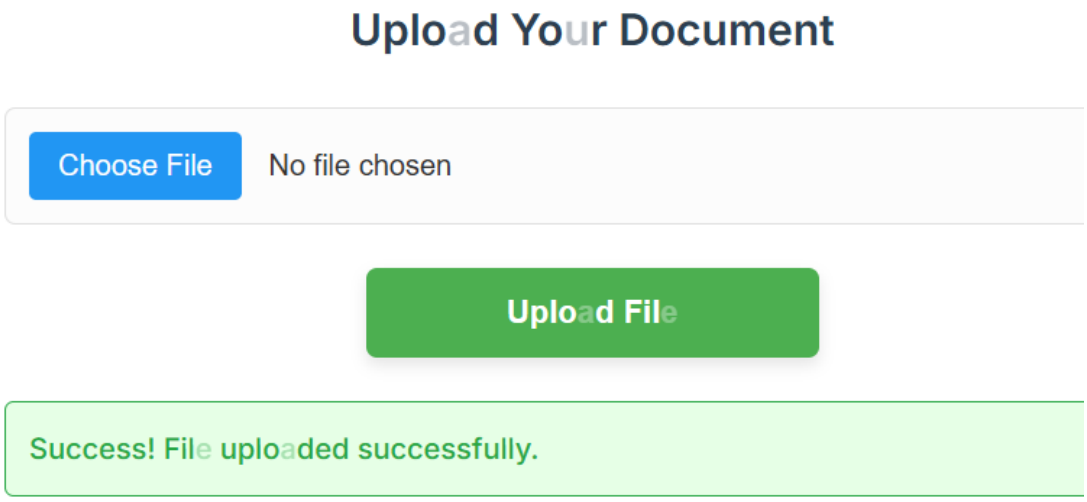


Рисунок 4.17 – Вигляд сторінки “File Upload”

Далі проведено тестування функцію “File Download”, при відкритті цієї сторінки можна відразу побачити весь список файлів доступних для вивантаження із системи, зображено на рисунку 4.18.

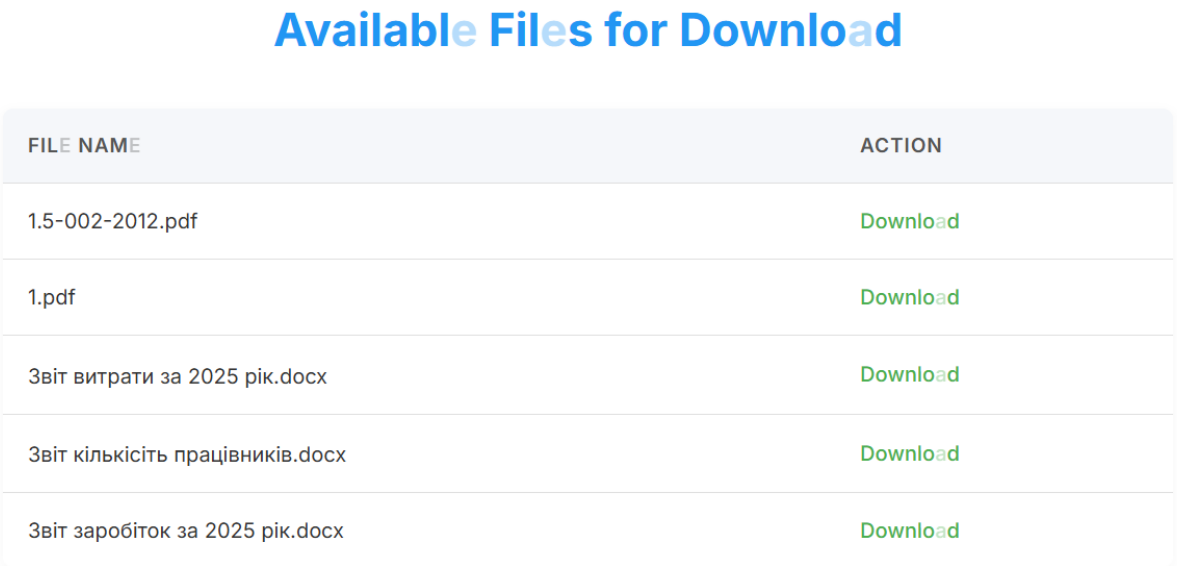


Рисунок 4.18 – Вигляд сторінки “File Download”

Як видно із рисунка 4.18 на сторінці веб-застосунку виведено всі файли, що доступні для завантаження, також тут можна засвідчити, що файл “1.pdf” було

завантажено успішно. Далі проведено тестування безпосередньо вивантаження, натиснувши на кнопку “Download”, для прикладу вивантажено файл “1.pdf”, результат зображено на рисунку 4.19.

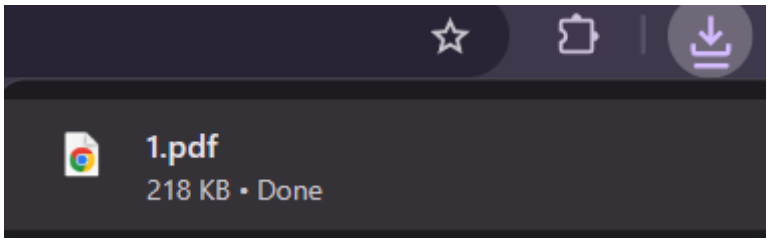


Рисунок 4.19 – Вигляд результатів завантаження файлу

Проведено тестування завантаження шаблонів, для цього потрібно перейти до пункту меню “Templates”, сама сторінка майже ідентична до “File Download”, для тестування завантажено файл, результат зображено на рисунку 4.20.

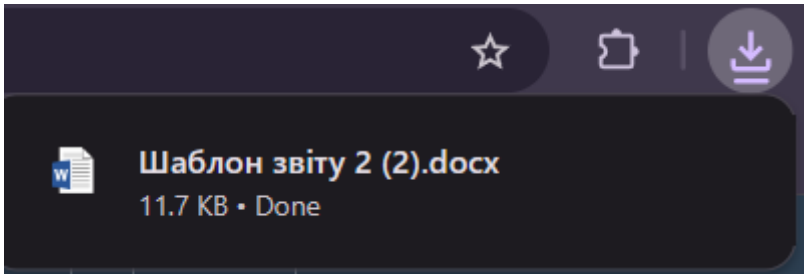


Рисунок 4.20 – Вигляд результатів завантаження шаблону

Далі проведено тестування підписання фалу за допомогою ECDSA. Потрібно відкрити пункт “Sign” після чого завантажити необхідні для підписання файли, файл для підписання та приватний ключ. У ролі приватного ключа виступатиме раніше згенерований ключ, у ролі фала буде виступати “титулка.docx”. Натиснути на кнопку “Sign File” після чого файл буде підписано та завантажено на комп’ютер, результат зображено на рисунку 4.21.

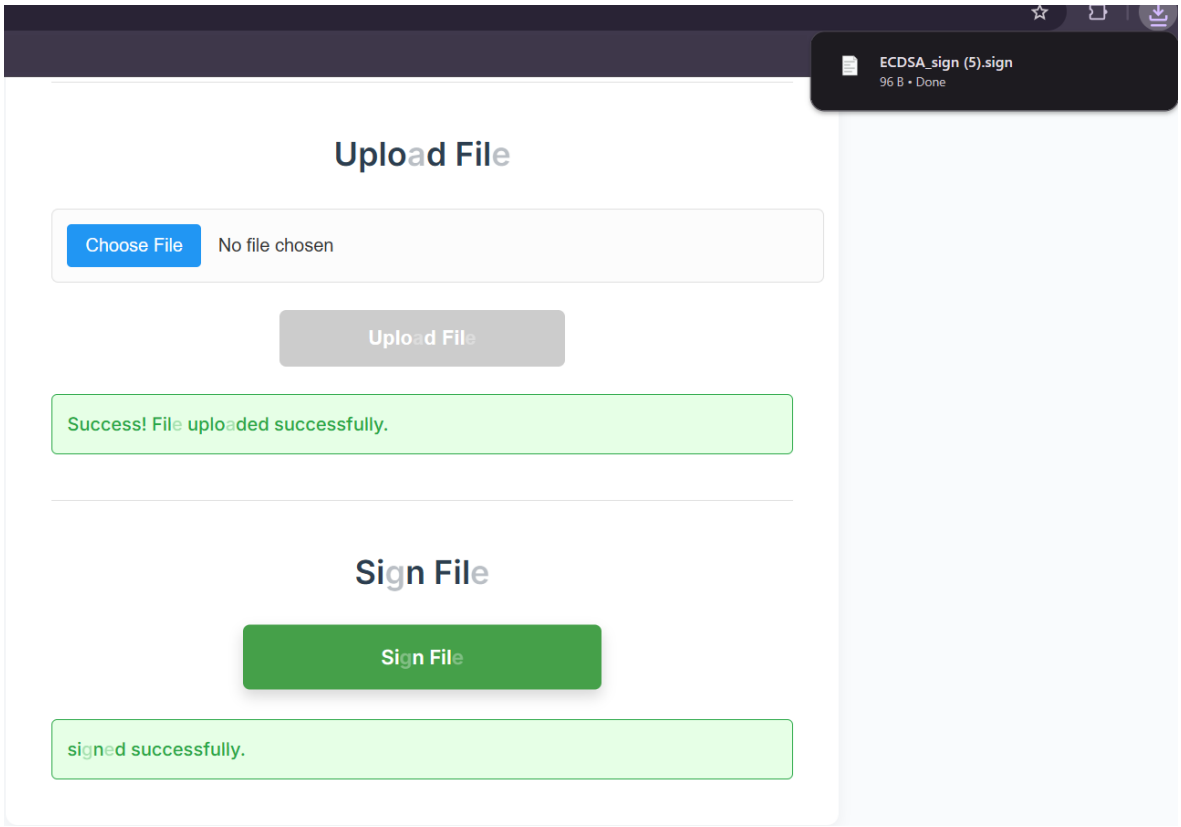


Рисунок 4.21 – Вигляд результатів підписання фалу використовуючи ECDSA

Після отримання підпису його потрібно перевірити, для цього потрібно відкрити сторінку “Verify”. Далі ввезти всі необхідні параметри, а саме підпис, публічний ключ та файл для перевірки. Далі потрібно натиснути кнопку “Verify File”, після чого буде отримано результат верифікації, зображено на рисунку 4.22.

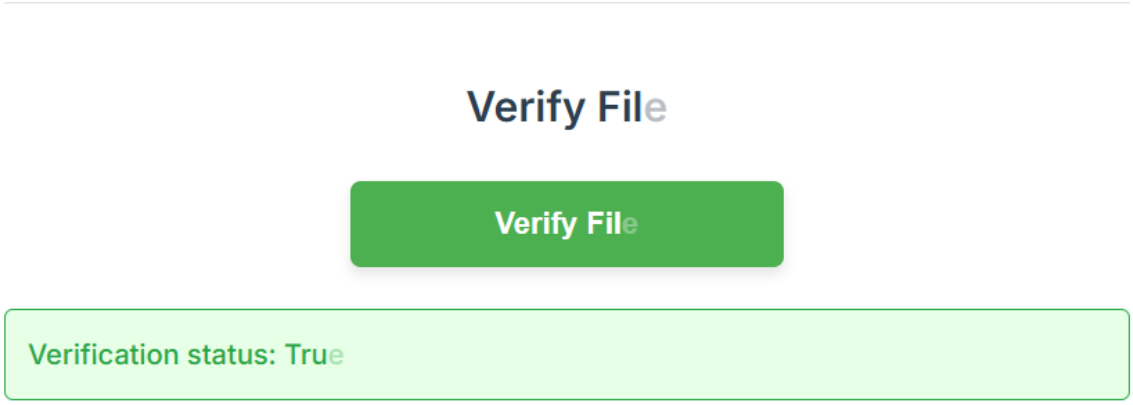


Рисунок 4.22 – Вигляд результатів верифікації підпису ECDSA

У випадку коли хоча б одного із параметрів не сходиться буде отримано такий результат, рисунок 4.23.

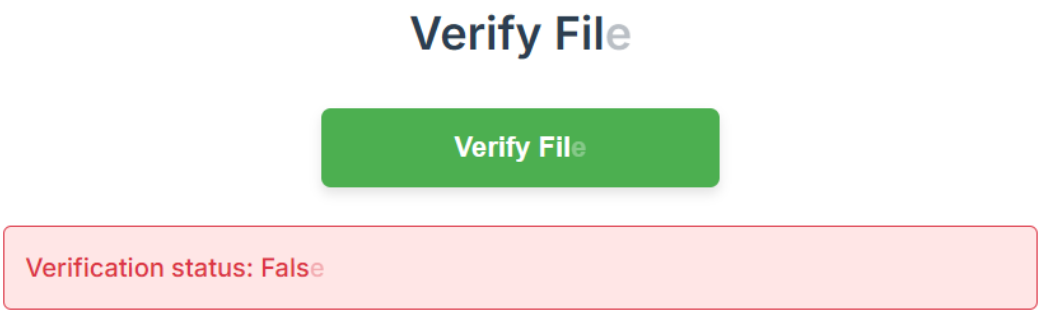


Рисунок 4.23 – Вигляд результатів хибної верифікації підпису ECDSA

Наступним кроком було проведено тестування підписання файлу із використанням ГП, для цього обрано пункт “Group Sign”. Сама сторінка складається із 3 форм для завантаження фалів необхідних для підписання та кнопки для виконання підписання. Так для підписання завантажено необхідні файли а саме приватний ключ користувача, публічний ключ та файл для підпису, всі необхідні ключі були згенеровані на попередніх кроках, зображено на рисунку 4.24 .

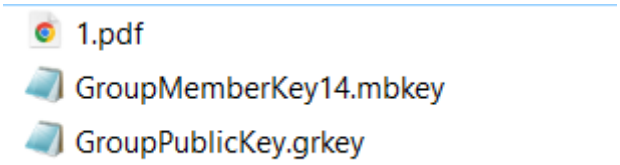


Рисунок 4.24 – Вигляд даних для створення ГП

Після завантаження всіх необхідних фалів у систему щоб підписати файл потрібно натиснути на кнопку “Sign File”, після чого файл буде підписано та завантажено на комп’ютер, результат зображено на рисунку 4.25.

Рисунок 4.25 – Вигляд файлу створеного ГП

Оскільки було отримано підпис файлу було б доречно перевірити роботу верифікації ГП, для цього необхідно перейти на сторінку “Group Verify”. Сторінка складається із однакових елементів як попередня, однак для верифікації фалу потрібно завантажити підпис, публічний ключ та файл для верифікації. Після заповнення всіх відповідних форм отримано такий результат, зображено на рисунку 4.26.

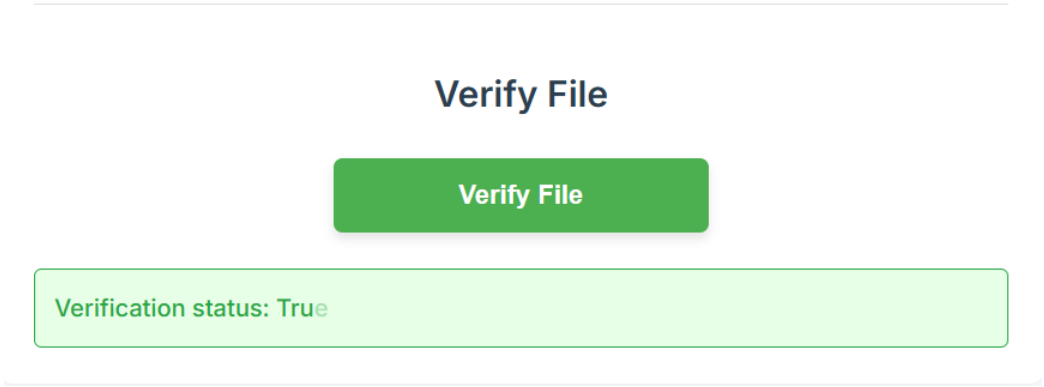


Рисунок 4.26 – Вигляд результатів верифікації ГП

У випадку не проходження верифікації буде таки самий результат як на рисунку 4.23.

Останньою перевіркою буде відкриття групового підпису, для цього потрібно зайти у систему від імені адміністратора, після чого обрати пункт “Open SGS”. Наступним кроком потрібно завантажити два файли: ключ адміністратора та підпис, після чого натиснути кнопку “Open Signature”. Як результат буде отримано частину приватного ключа підписанта, зображено на рисунку 4.27.

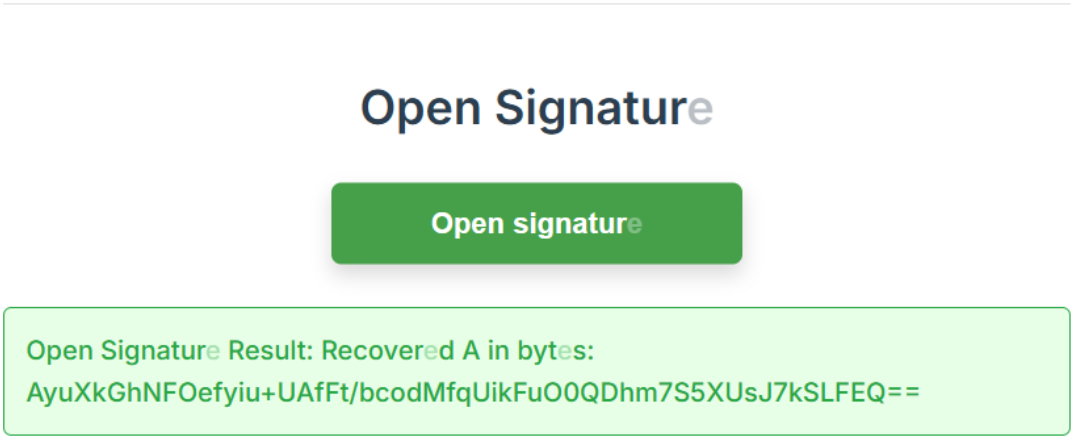


Рисунок 4.27 – Вигляд результатів відкриття підпису

Далі отримане значення потрібно порівняти із приватним ключем користувача, для досягнення більш зрозумілого процесу перевірки було взято два приватних ключа, зображено на рисунку 4.28 .

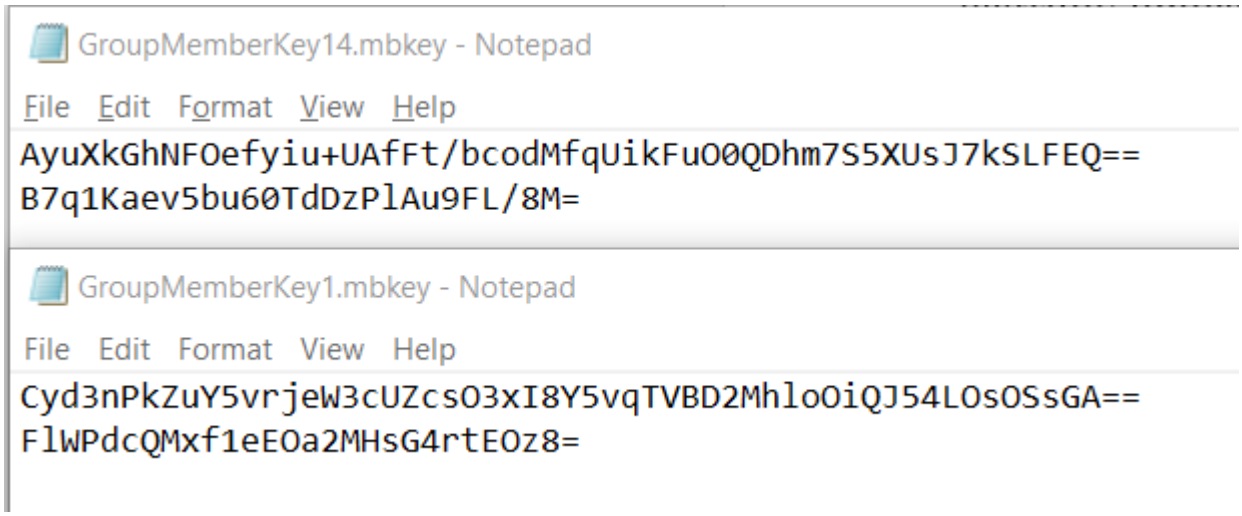


Рисунок 4.28 – Вигляд кодування приватних ключі для перевірки

Як видно із рисунку 4.27 та 4.28 значення отримане в наслідок відкриття підпису збігається із значенням, що знаходиться у файлі під назвою “GroupMemberKey14.mbkey”, таким чином можна зробити висновок, що саме цим ключем було підписано файл.

4.5 Висновки з розділу

Під час виконання розділу було протестована розроблену СЕД. Проведено такі види тестування як модульне, статичне та інтеграційне. Спочатку було визначено тестове середовище, для модульного тестування використано модуль JUnit 5 та для статично розширення PMD. Далі розроблено тестові модулі для застосунку та проведено необхідні перевірки. Наступним кроком проведено статистичне тестування за допомого розширення, що було обрано раніше. Останнім кроком проведення інтеграційне тестування всіх функцій веб-застосунку. Під час процесу тестування системи було виявлено певні помилки, що були успішно виправлені. Результат показав, що система повністю працююча та не має явних відхилень у своїй роботі.

ВИСНОВКИ

Під час виконання бакалаврської кваліфікаційної роботи було створено систему захищеного електронного документообігу у вигляді веб-застосунку із можливістю підпису документів із допомогою ГП.

Виконано аналіз СЕД, що включає аналіз головних задач таких систем, нормативно-правового регулювання, що належить до цієї тематики, методів формування ЕП, проведено порівняльний аналіз відомих рішень СЕД. У результаті досліджень згідно цієї тематики було визначено основну інформацію, що необхідна для розробки наступних розділів роботи. На основі отриманих результатів поставлено подальші завдання для виконання.

Наступним кроком розроблено архітектуру системи, для цього було створено її узагальнену архітектуру; архітектуру блоків ГП, авторизації та автентифікації, збереження та завантаження файлів. Також було обрано метод групового підпису, що буде використовуватись у роботі, та обґрунтовано чому саме потрібно використовувати саме ГП, а не інші альтернативи ЕП.

Після розробки архітектури було розпочато роботу над алгоритмами роботи системи. Спочатку було створено узагальнений алгоритм роботи системи, далі, згідно із відповідними матеріалами, було створено всі необхідні алгоритми для повного функціонування роботи ГП та ЕП у системі. Розробка алгоритмів дозволила обґрунтувати вибір засобів розробки веб-застосунку.

Далі після отримання всієї необхідної інформації було розпочато розробку безпосередньо веб-застосунку СЕД, одночасно із цим процесом було створено модульні тести для системи, що допомогли уникнути великої кількості помилок на етапі розробки.

Після виконання попередніх розділів роботи було виконано тестування вже створеної системи. Для цього було охарактеризовано тестове середовище системи, що включає в себе всі необхідні інструменти, які потрібно для тестування. Далі проведено модульне тестування системи, шляхом розроблення та використання додаткових та вже розроблених тестів. Наступним кроком проведено тестування засобами статичного тестування безпеки та виправлено виявлені помилки. На

завершальному етапі проєктування проведено інтеграційне тестування усіх функцій системи. За результатами виконання інтеграційного тестування визначено коректність реалізації архітектури, інтерфейсів між окремими її складовими, а також алгоритмів роботи цих складових.

Розроблена система відповідає всім поставленим вимогам та завданням. Відповідно до українського законодавства, для подальшого комерційного використання така система має бути допрацьована згідно нормативно-правових регулювань. Сама система має високий потенціал подальшого модернізування, одними із таких напрямків є впровадження нових методів підписання документів, запровадження кращої автоматизації процесів роботи системи, провадження додаткових способів захисту, зокрема гомоморфного шифрування.

Для інтегрування розробленої СЕД із інформаційними системами об’єктів критичної інфраструктури потрібно буде провести аудит відповідності нормативно правовій базі відповідного сегменту промисловості об’єкта критичної інфраструктури. Після такого аудити розроблену СЕД буде найбільш доцільно використовувати у структурі Міністерства цифрової трансформації України, банках, та державних підприємств пов’язаних із виготовлення програмного забезпечення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Д.В. Лобай, Ю.В. Баришев. Використання групових підписів у системах електронного документообігу. *Всеукраїнська науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії (2025)* Вінниця, 2025 3 с. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/24109/19950 (дата звернення: 12.05.2025).

2. InBase. Електронний документообіг: системи, види, особливості впровадження та як він працює в Україні. URL: https://inbase.com.ua/elektronnyj-dokumentoobig-systemy-vydy-osoblyvosti-vprovadzhennya-ta-yak-vin-praczyuye-v-ukrayini/ (дата звернення: 12.05.2025).

3. Вчасно.ЕДО. Внутрішні документи підприємства: правила і переваги переходу на електронний документообіг. URL: https://vchasno.ua/vnutrishni-dokumenty/ (дата звернення: 12.05.2025).

4. InBase. Види документообігу: детальний гайд для початківців з прикладами та поясненнями. URL: https://inbase.com.ua/vydy-dokumentoobigu-detalnyj-gajd/ (дата звернення: 12.05.2025).

5. Вчасно.ЕДО. Що таке електронний документообіг? URL: https://vchasno.ua/shcho-take-elektronnyi-dokumentoobig/ (дата звернення: 12.05.2025).

6. Deals. Електронний документообіг: види систем та їхні функції. URL: https://dealssign.com/blog/elektronnij-dokumentoobig-vidi-sistem-ta-yixni-funkci (дата звернення: 12.05.2025).

7. Про електронні документи та електронний документообіг. Закон України від 31.12.2023 № 851-IV. URL: https://zakon.rada.gov.ua/laws/show/851-15#Text (дата звернення: 12.05.2025).

8. Про електронну комерцію. Закон України від 01.01.2024 № 675-VIII. URL: https://zakon.rada.gov.ua/laws/show/675-19#Text (дата звернення: 12.05.2025).

9. Про міжнародне приватне право. Закон України від 23.12.2022 № 2709-IV URL: https://zakon.rada.gov.ua/laws/show/2709-15#Text (дата звернення: 11.06.2025).

10. Про електронну ідентифікацію та електронні довірчі послуги. Закон України від 18.12.2024 № 2155-VIII. URL: https://zakon.rada.gov.ua/laws/show/2155-19#Text (дата звернення: 12.05.2025).

11. Деякі питання документування управлінської діяльності, Постанова Кабінету Міністрів від 17.04.2025 № 55. URL: https://zakon.rada.gov.ua/laws/show/55-2018-%D0%BF#Text (дата звернення: 12.05.2025).

12. Про затвердження порядку роботи з електронними документами у діловодстві та їх підготовки до передавання на архівне зберігання. Наказ Міністерства юстиції України від 10.05.2025 № 1886/5. URL: https://zakon.rada.gov.ua/laws/show/z1421-14#Text (дата звернення: 12.05.2025).

13. General Data Protection Regulation: Regulation (EU) 2016/679 of the european parliament and of the council of 27 April 2016. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679 (accessed: 11.06.2025).

14. ДСТУ ISO 30300:2015. Інформація та документація. Системи керування документами. Основні положення і словник термінів. [Чинний від 2017-01-01]. Вид. офіц. Київ, 2015. URL: https://online.budstandart.com/ua/catalog/doc-page?id_doc=67046 (дата звернення: 12.05.2025).

15. ДСТУ ISO 30301:2015. Інформація та документація. Системи керування документами. Вимоги. [Чинний від 2016–01–01]. Вид. офіц. Київ, 2015. URL: https://online.budstandart.com/ua/catalog/doc-page?id_doc=77826 (дата звернення: 12.05.2025).

16. ДСТУ ISO 30302:2018. Інформація та документація. Системи керування записами. Настанови щодо впровадження. [Чинний від 2019–01–01]. Вид. офіц. Київ, 2018. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=80408 (дата звернення: 12.05.2025).

17. R.L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. URL: https://people.csail.mit.edu/rivest/Rsapaper.pdf (accessed: 12.05.2025).

18. Blockchain Behind The Scenes. What is Digital Signature Algorithm ?URL: https://medium.com/@21_000_000/digital-signature-algorithm-60c8318cf9b6 (accessed: 12.05.2025).

19. D. Johnson, A. Menezes. The Elliptic Curve Digital Signature Algorithm (ECDSA). URL: https://cacr.uwaterloo.ca/techreports/1999/corr99-34.pdf (accessed: 12.05.2025).

20. D. Boneh, B. Lynn, H. Shacham. Short signatures from the Weil pairing. URL: https://www.iacr.org/archive/asiacrypt2001/22480516.pdf (accessed: 12.05.2025).

21. ДСТУ 9212:2023. Інформаційні технології. Криптографічний захист інформації. Алгоритм електронного підпису на алгебраїчних решітках із відхилами. [Чинний від 2023–10–01]. Київ, 2023. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=102150 (дата звернення: 12.06.2025).

22. І. Самойлов, А. Матійко, А. Сторчак. Криптографія: навчальний посібник. Київ: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2023. 372 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/5faf3d32-f858-4eec-8f10-b059dc28d7e4/content (дата звернення: 11.06.2025).

23. Geeksforgeeks. Digital Signature Algorithm (DSA). URL: https://www.geeksforgeeks.org/digital-signature-algorithm-dsa/ (accessed: 12.05.2025).

24. Understanding How ECDSA Protects Your Data. URL: https://www.instructables.com/Understanding-how-ECDSA-protects-your-data/ (accessed: 12.05.2025).

25. Information Technology Laboratory National Institute of Standards and Technology Gaithersburg. Federal Information Processing Standards Publication 186-5. URL: https://doi.org/10.6028/NIST.FIPS.186-5 (accessed: 12.05.2025).

26. A. Muroch. BLS Signatures Part 1 – Overview. URL: https://alonmuroch-65570.medium.com/bls-signatures-part-1-overview-47d9eebf1c75 (accessed: 12.05.2025).

27. Леонорм. Нові надходження нормативних документів українською мовою: ДСТУ 9212:2023 "Інформаційні технології. Криптографічний захист

інформації. Алгоритм електронного підпису на алгебраїчних решітках із відхилами". URL: http://www.leonorm.lviv.ua/p/News/12_2023/NTNFD_28_12_2023.htm (accessed: 12.05.2025).

28. D. Chaum, E. van Heyst. Group Signatures. Advances in Cryptology - EUROCRYPT '91, LNCS 547, 1991, pp. 257-265. URL: https://link.springer.com/content/pdf/10.1007/3-540-46416-6_22.pdf (accessed: 12.05.2025).

29. A. Bender, J. Katz, R. Morselli. Ring Signatures: Stronger Definitions, and Constructions Without Random Oracles. URL: https://link.springer.com/chapter/10.1007/11681878_4 (accessed: 12.05.2025).

30. Coinbase. What is Multi-Signature (Multi-Sig)? URL: https://www.coinbase.com/ru/learn/wallet/what-is-a-multi-signature-multi-sig-wallet (accessed: 12.05.2025).

31. Fydoc. Функціонал. URL: https://flydoc.in.ua/functions/ (дата звернення: 12.05.2025).

32. Medoc. Інструкції по налаштуванню та роботі в M.E.Doc. URL: https://medoc.ua/faq/instructions (дата звернення: 12.05.2025).

33. Посібник користувача «СОТА». URL: https://sota-buh.com.ua/content/manuals/help.pdf (accessed: 12.05.2025).

34. Box. Product Guides. URL: https://support.box.com/hc/en-us/sections/21356697745811-For-Users (accessed: 12.05.2025).

35. DocuWare. Resources for New DocuWare Customers. URL: https://start.docuware.com/resources-for-new-docuware-users (accessed: 12.05.2025).

36. Revver. Features. URL: https://support.revverdocs.com/en/collections/3977776-features (accessed: 12.05.2025).

37. N. Fallon. DocuWare Review. URL: https://www.businessnewsdaily.com/docuware-review (accessed: 12.05.2025).

38. M. Freedman. Revver Review and Pricing. URL: https://www.businessnewsdaily.com/11259-best-low-cost-document-management-system.html (accessed: 12.05.2025).

39. L. Garms, A. Lehmann. Group Signatures with Selective Linkability. URL: https://eprint.iacr.org/2019/027.pdf (accessed: 12.05.2025).

40. D. Boneh, X. Boyen, H. Shacham. Short Group Signatures. URL: https://crypto.stanford.edu/~dabo/papers/groupsigs.pdf (accessed: 11.06.2025).

41. D. Pointcheval, O. Sanders. Short Randomizable Signatures. URL: https://crypto.stanford.edu/~dabo/papers/groupsigs.pdf (accessed: 11.06.2025).

42. Microsoft. ASP.NET documentation. URL: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-9.0 (accessed: 11.06.2025).

43. Spring. Spring Framework Overview. URL: https://docs.spring.io/spring-framework/reference/overview.html (accessed: 11.06.2025).

44. A. Menezes. An Introduction to Pairing-Based Cryptography. URL: https://www.math.uwaterloo.ca/~ajmeneze/publications/pairings.pdf (accessed: 11.06.2025).

45. JPBC Library. URL: http://gas.dia.unisa.it/projects/jpbc/ (accessed: 11.06.2025).

46. IntelliJIDEA. Features overview. URL: https://www.jetbrains.com/idea/features/# (accessed: 11.06.2025).

47. Eclipse. Developer Tools & IDE. URL: https://www.eclipse.org/topics/ide/ (accessed: 11.06.2025).

48. Gradle. Gradle User Manual. URL: https://docs.gradle.org/current/userguide/userguide.html (accessed: 11.06.2025).

49. Apache Maven. Introduction. URL: https://maven.apache.org/what-is-maven.html (accessed: 11.06.2025).

50. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 125 «Кібербезпека» освітньо-професійні програми «Безпека інформаційних і комунікаційних систем» та «Кібербезпека критичних систем» / Уклад. В. А. Каплун, О. П. Войтович. Вінниця: ВНТУ, 2023. 61 с.

ДОДАТКИ

Додаток А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Система захищеного електронного документообігу

Автор роботи: Лобай Дмитро Віталійович

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ , група 1 БКС-21 б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism **4,09** %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

√ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Завідувач кафедри ЗІ д. т. н., проф. _____ Володимир ЛУЖЕЦЬКИЙ

Гарант освітньої програми «Кібербезпека критичних систем» к. т. н., доц.
_____ Юрій БАРИШЕВ

Особа, відповідальна за перевірку _____ Валентина КАПЛУН

З висновком експертної комісії ознайомлений(-на)

Керівник _____ _____

Здобувач _____ _____

Додаток Б

ТЕКСТ ПРОГРАМИ ГРУПОВОГО ПІДПИСУ

Клас “BBSGroupSignature” :

```
package com.example.demo.groupSignatureLib;
import it.unisa.dia.gas.jpbc.*;
import it.unisa.dia.gas.plaf.jpbc.pairing.PairingFactory;
public class BBSGroupSignature {
private Pairing pairing;
private Field G1, G2, Zr;
private Element g1, g2, h, u, v, omega;
private Element xi_1, xi_2;
private Element A, x;
private Element gamma;
private Element T1,  T2,  T3,  c,
s_alpha,  s_beta,  s_x,
s_delta1,  s_delta2;
public BBSGroupSignature(String pairingParamsPath) {
this.pairing = PairingFactory.getPairing(pairingParamsPath);
this.G1 = pairing.getG1();
this.G2 = pairing.getG2();
this.Zr = pairing.getZr();
}
public void generateKeys() {

this.g1 = G1.newRandomElement().getImmutable();
this.g2 = G2.newRandomElement().getImmutable();
this.h = G1.newRandomElement().getImmutable();
this.xi_1 = Zr.newRandomElement().getImmutable();
this.xi_2 = Zr.newRandomElement().getImmutable();
this.u = h.powZn(xi_1.invert()).getImmutable();
this.v = h.powZn(xi_2.invert()).getImmutable();
this.gamma = Zr.newRandomElement().getImmutable();
this.omega = g2.powZn(gamma).getImmutable();}
public Element[] getGroupPublicKeys() {
Element[] publicKeys = new Element[6];
publicKeys[0] = g1;
publicKeys[1] = g2;
publicKeys[2] = h;
publicKeys[3] = u;
publicKeys[4] = v;
publicKeys[5] = omega;
```

```
return publicKeys;
}
public Element[] getGroupMemberKeys( ) {
Element newA,newx;
newx = Zr.newRandomElement().getImmutable();
newA = g1.powZn(gamma.add(newx).invert()).getImmutable();

Element[] privateKeys = new Element[2];
this.A = newA;
this.x = newx;
privateKeys[0] = newA;
privateKeys[1] = newx;
return privateKeys;
}
public Element[] getGroupManagerKeys() {
Element[] publicKeys = new Element[2];
publicKeys[0] = xi_1;
publicKeys[1] = xi_2;
return publicKeys;
}
public void sign
(String message,  Pairing pr, Field Zr, Element v,
Element u, Element h, Element A, Element x, Element g2,
Element omega) {
Element alpha = Zr.newRandomElement().getImmutable();
Element beta = Zr.newRandomElement().getImmutable();
T1 = u.powZn(alpha).getImmutable();
T2 = v.powZn(beta).getImmutable();
T3 = A.mul(h.powZn(alpha.add(beta))).getImmutable();
Element r_alpha = Zr.newRandomElement().getImmutable();
Element r_beta = Zr.newRandomElement().getImmutable();
Element r_x = Zr.newRandomElement().getImmutable();
Element r_delta1 = Zr.newRandomElement().getImmutable();
Element r_delta2 = Zr.newRandomElement().getImmutable();
Element R1 = u.powZn(r_alpha).getImmutable();
Element R2 = v.powZn(r_beta).getImmutable();
Element R3 = pr.pairing(T3, g2).powZn(r_x)
.mul(pr.pairing(h, omega).powZn(r_alpha.negate().add(r_beta.negate())))
.mul(pr.pairing(h, g2).powZn(r_delta1.negate().sub(r_delta2)))
.getImmutable();

Element R4 = T1.powZn(r_x).mul(u.powZn(r_delta1.negate())).getImmutable();
Element R5 = T2.powZn(r_x).mul(v.powZn(r_delta2.negate())).getImmutable();
```

```
String M_sign = message + T1 + T2 + T3 + R1 + R2 + R3 + R4 + R5;
c = Zr.newElementFromHash(Integer.toString(M_sign.hashCode()).getBytes(), 0,
4).getImmutable();
s_alpha = r_alpha.add(c.mul(alpha)).getImmutable();
s_beta = r_beta.add(c.mul(beta)).getImmutable();
s_x = r_x.add(c.mul(x)).getImmutable();
s_delta1 = r_delta1.add(c.mul(x.mul(alpha))).getImmutable();
s_delta2 = r_delta2.add(c.mul(x.mul(beta))).getImmutable();
}

public Element[] getSignature() {
Element[] publicKeys = new Element[9];
publicKeys[0] = T1;
publicKeys[1] = T2;
publicKeys[2] = T3;
publicKeys[3] =  c;
publicKeys[4] =  s_alpha;
publicKeys[5] = s_beta;
publicKeys[6] = s_x;
publicKeys[7] = s_delta1 ;
publicKeys[8] = s_delta2;
return publicKeys;
}
public boolean verify(String message,Pairing pr, Element T1, Element T2, Element
T3, Element c,
Element s_alpha, Element s_beta, Element s_x,
Element s_delta1, Element s_delta2 , Element u,
Element v, Element g1, Element g2,Element h, Element omega ,Field Zr) {
Element R1_bar =
u.powZn(s_alpha).mul(T1.powZn(c.duplicate().negate())).getImmutable();
Element R2_bar =
v.powZn(s_beta).mul(T2.powZn(c.duplicate().negate())).getImmutable();
Element R3_bar = pr.pairing(T3, g2).powZn(s_x)
.mul(pr.pairing(h, omega).powZn(s_alpha.duplicate().negate().sub(s_beta)))
.mul(pr.pairing(h,
g2).powZn(s_delta1.duplicate().negate().add(s_delta2.duplicate().negate())))
.mul(pr.pairing(T3, omega).div(pr.pairing(g1, g2)).powZn(c)).getImmutable();
Element R4_bar =
T1.powZn(s_x).mul(u.powZn(s_delta1.duplicate().negate())).getImmutable();
Element R5_bar =
T2.powZn(s_x).mul(v.powZn(s_delta2.duplicate().negate())).getImmutable();
```

```
String M_verify = message + T1 + T2 + T3 + R1_bar + R2_bar + R3_bar + R4_bar +
R5_bar;
Element c_ =
Zr.newElementFromHash(Integer.toString(M_verify.hashCode()).getBytes(), 0,
4).getImmutable();
return c_.isEqual(c);
}
public Element openSignature(Element T1, Element T2, Element T3, Element xi_1,
Element xi_2) {
Element A_ = T3.div(T1.powZn(xi_1).mul(T2.powZn(xi_2)));
return A_; }}
```

Клас “GroupSignatureController” :

```
package com.example.demo.groupSignatureLib;
import com.example.demo.NumberInputRequest;
import com.example.demo.files.FileManagerController;
import com.example.demo.files.FileStorageService;
import it.unisa.dia.gas.jpbc.Element;
import it.unisa.dia.gas.jpbc.Field;
import it.unisa.dia.gas.jpbc.Pairing;
import it.unisa.dia.gas.plaf.jpbc.pairing.PairingFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.ByteArrayResource;
import org.springframework.core.io.InputStreamResource;
import org.springframework.core.io.Resource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import java.io.*;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;
```

```
@RestController
public class GroupSignatureController {
Pairing bp = PairingFactory.getPairing("D:\\java bach
project\\params\\curves\\f.properties");
BBSGroupSignature bbs = new BBSGroupSignature("D:\\java bach
project\\params\\curves\\f.properties");
Field Zr = bp.getZr();
@Autowired
private FileStorageService fileStorageService;
private static final Logger log =
Logger.getLogger(FileManagerController.class.getName());
@GetMapping("/downloadKeys")
public ResponseEntity<Resource> downloadFile() {
try {
Path[] filePaths = new Path[] {
Paths.get("D:\\java bach project\\FileStorage\\GroupPublicKey.gbkey"),
Paths.get("D:\\java bach project\\FileStorage\\GroupManagerKey.mgkey"),
Paths.get("D:\\java bach project\\FileStorage\\GroupMemberKeys.zip")};
ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
ZipOutputStream zipOut = new ZipOutputStream(byteArrayOutputStream);
for (Path path : filePaths) {
if (!Files.exists(path) || !Files.isReadable(path)) {
return ResponseEntity.notFound().build();}
ZipEntry zipEntry = new ZipEntry(path.getFileName().toString());
zipOut.putNextEntry(zipEntry);
byte[] fileBytes = Files.readAllBytes(path);
zipOut.write(fileBytes);
zipOut.closeEntry();}
zipOut.close();
ByteArrayResource resource = new
ByteArrayResource(byteArrayOutputStream.toByteArray());

return ResponseEntity.ok()
.header(HttpHeaders.CONTENT_DISPOSITION, "attachment;
filename=\"keys_bundle.zip\"")
.contentType(MediaType.APPLICATION_OCTET_STREAM)
.contentLength(resource.contentLength())
.body(resource);
} catch (IOException ex) {
ex.printStackTrace();
return ResponseEntity.status(500).build();}}
@PostMapping("/generateKeys")
```

```
public ResponseEntity<String> generateKeys(@RequestBody NumberInputRequest
request) {
int number = request.getNumber();
if (number < 10 || number > 50) {
return ResponseEntity.badRequest().body("Number must be between 10 and 50.");
}
bbs.generateKeys();
Element[] groupPublicKeysKeys = bbs.getGroupPublicKeys();
Element g1 = groupPublicKeysKeys[0];
Element g2 = groupPublicKeysKeys[1] ;
Element h = groupPublicKeysKeys[2];
Element u = groupPublicKeysKeys[3];
Element v = groupPublicKeysKeys[4];
Element omega = groupPublicKeysKeys[5];
Element [] createGroupManagerKeys = bbs.getGroupManagerKeys();
Element xi_1  = createGroupManagerKeys [0];
Element xi_2 =createGroupManagerKeys [1] ;
String filename = "D:\\java bach project\\FileStorage\\GroupPublicKey.gbkey";
try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {
writer.write(Base64.getEncoder().encodeToString(g1.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(g2.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(h.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(u.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(v.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(omega.toBytes()));
writer.newLine();
} catch (IOException e) {
throw new RuntimeException(e);
}
try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
byte[] g1Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] g2Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] hBytes =Base64.getDecoder().decode(reader.readLine());
byte[] uBytes = Base64.getDecoder().decode(reader.readLine());
byte[] vBytes = Base64.getDecoder().decode(reader.readLine());
g1 = bp.getG1().newElementFromBytes(g1Bytes).getImmutable();
g2 = bp.getG2().newElementFromBytes(g2Bytes).getImmutable();
h = bp.getG1().newElementFromBytes(hBytes).getImmutable();
```

```
u = bp.getG1().newElementFromBytes(uBytes).getImmutable();
v= bp.getG1().newElementFromBytes(vBytes).getImmutable();
} catch (IOException e) {
throw new RuntimeException(e);
}
filename = "D:\\java bach project\\FileStorage\\GroupManagerKey.mgkey";
try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {
writer.write(Base64.getEncoder().encodeToString(xi_1.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(xi_2.toBytes()));
writer.newLine();
} catch (IOException e) {
throw new RuntimeException(e);}
try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
byte[] xi_1Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] xi_2Bytes = Base64.getDecoder().decode(reader.readLine());
xi_1 = bp.getZr().newElementFromBytes(xi_1Bytes).getImmutable();
xi_2 = bp.getZr().newElementFromBytes(xi_2Bytes).getImmutable();
} catch (IOException e) {
throw new RuntimeException(e);
}
List<Path> generatedFiles = new ArrayList<>();
Element[] keyPair;
Element A, x;
String folderPath = "D:\\java bach project\\FileStorage";
for (int i = 0; i < number; i++) {
String filename2 = String.format("GroupMemberKey%d.mbkey", i+1);
Path filePath = Paths.get(folderPath, filename2);
generatedFiles.add(filePath);
keyPair = bbs.getGroupMemberKeys();
A = keyPair[0];
x = keyPair[1];
try (BufferedWriter writer = new BufferedWriter(new
FileWriter(filePath.toFile()))) {
writer.write(Base64.getEncoder().encodeToString(A.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(x.toBytes()));
writer.newLine();
} catch (IOException e) {
throw new RuntimeException("Помилка запису ключа у файл: " + filePath, e);}}
Path zipPath = Paths.get(folderPath, "GroupMemberKeys.zip");
try (ZipOutputStream zipOut = new ZipOutputStream(new
FileOutputStream(zipPath.toFile()))) {
```

```
for (Path path : generatedFiles) {
if (Files.exists(path)) {
ZipEntry zipEntry = new ZipEntry(path.getFileName().toString());
zipOut.putNextEntry(zipEntry);
byte[] fileBytes = Files.readAllBytes(path);
zipOut.write(fileBytes);
zipOut.closeEntry();}}
} catch (IOException e) {
throw new RuntimeException("Помилка під час створення ZIP-архіву", e);}
System.out.println("ZIP-файл створено: " + zipPath.toString());
String resultMessage = "Keys successfully generated for number: " + number;
return ResponseEntity.ok(resultMessage);}
@PostMapping("/uploadPrKey")
public ResponseEntity<String> uploadFile(@RequestParam("file") MultipartFile
file){
try {
fileStorageService.saveFilePath(file,"D:\\java bach
project\\Files\\SignFolder","prkey.mrkey");
return ResponseEntity.ok("File uploaded successfully.");
} catch (IOException e) {
log.log(Level.SEVERE, "Exception during upload", e);
return ResponseEntity.status(500).body("Upload failed.");}}
@PostMapping("/uploadPubKey")
public ResponseEntity<String> uploadPubKey(@RequestParam("file") MultipartFile
file) {
try {
fileStorageService.saveFilePath(file,"D:\\java bach
project\\Files\\SignFolder","GroupPublicKey.gbkey");
return ResponseEntity.ok("File uploaded successfully.");
} catch (IOException e) {
log.log(Level.SEVERE, "Exception during upload", e);
return ResponseEntity.status(500).body("Upload failed.");}}
@PostMapping("/uploadFileToSign")
public ResponseEntity<String> uploadFileToSign(@RequestParam("file") MultipartFile
file) {
try {
fileStorageService.saveFilePath(file,"D:\\java bach
project\\Files\\SignFolder","fileToSign.txt");
return ResponseEntity.ok("File uploaded successfully.");
} catch (IOException e) {
log.log(Level.SEVERE, "Exception during upload", e);
return ResponseEntity.status(500).body("Upload failed.");}}
@GetMapping("/signGroupFile")
```

```
public ResponseEntity<Resource> signFile() {
String stringFile = "";
Element g1, g2, h, u, v, omega;
Element A,x ;
String fileName = "D:\\java bach project\\Files\\SignFolder\\fileToSign.txt";
try {
stringFile = new String(Files.readAllBytes(Paths.get(fileName)),
java.nio.charset.StandardCharsets.UTF_8);
} catch (IOException e) {
System.err.println("Error reading file '" + fileName + "': " + e.getMessage());
e.printStackTrace();}
String filename = "D:\\java bach project\\Files\\SignFolder\\prkey.mrkey";
try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
byte[] ABytes = Base64.getDecoder().decode(reader.readLine());
byte[] xBytes = Base64.getDecoder().decode(reader.readLine());
A = bp.getG1().newElementFromBytes(ABytes).getImmutable();
x = bp.getZr().newElementFromBytes(xBytes).getImmutable();
} catch (IOException e) {
throw new RuntimeException(e);}
filename = "D:\\java bach project\\Files\\SignFolder\\GroupPublicKey.gbkey";
try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
byte[] g1Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] g2Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] hBytes =Base64.getDecoder().decode(reader.readLine());
byte[] uBytes = Base64.getDecoder().decode(reader.readLine());
byte[] vBytes = Base64.getDecoder().decode(reader.readLine());
byte[] omegaBytes = Base64.getDecoder().decode(reader.readLine());
g1 = bp.getG1().newElementFromBytes(g1Bytes).getImmutable();
g2 = bp.getG2().newElementFromBytes(g2Bytes).getImmutable();
h = bp.getG1().newElementFromBytes(hBytes).getImmutable();
u = bp.getG1().newElementFromBytes(uBytes).getImmutable();
v= bp.getG1().newElementFromBytes(vBytes).getImmutable();
omega = bp.getG2().newElementFromBytes(omegaBytes).getImmutable();
} catch (IOException e) {
throw new RuntimeException(e);}
bbs.sign(stringFile, bp,  Zr,  v,  u,  h,  A,  x,  g2,  omega);
Element [] signatureData = bbs.getSignature();
Element T1 = signatureData[0],  T2= signatureData[1],  T3= signatureData[2],  c=
signatureData[3],
s_alpha= signatureData[4],  s_beta= signatureData[5],  s_x= signatureData[6],
s_delta1= signatureData[7],  s_delta2= signatureData[8];
filename = "sign.sign";
try (BufferedWriter writer = new BufferedWriter(new FileWriter(filename))) {
```

```
writer.write(Base64.getEncoder().encodeToString(T1.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(T2.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(T3.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(c.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(s_alpha.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(s_beta.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(s_x.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(s_delta1.toBytes()));
writer.newLine();
writer.write(Base64.getEncoder().encodeToString(s_delta2.toBytes()));
writer.newLine();
} catch (IOException e) {
throw new RuntimeException(e);}
try {
Path path = Paths.get(filename);
if (!Files.exists(path)) {
return ResponseEntity.notFound().build();}
InputStreamResource resource = new
InputStreamResource(Files.newInputStream(path));
return ResponseEntity.ok()
.header(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=\"" + filename +
"\"")
.contentLength(Files.size(path))
.contentType(MediaType.APPLICATION_OCTET_STREAM)
.body(resource);
} catch (IOException e) {
log.log(Level.SEVERE, "Error reading file", e);
return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();}}
@PostMapping("/uploadFileToVerify")
public ResponseEntity<String> uploadFileToVerify(@RequestParam("file")
MultipartFile file) {
try {
fileStorageService.saveFilePath(file,"D:\\java bach
project\\Files\\VerifyFolder","fileToVerify.txt");
return ResponseEntity.ok("File uploaded successfully.");
} catch (IOException e) {
```

```
log.log(Level.SEVERE, "Exception during upload", e);
return ResponseEntity.status(500).body("Upload failed.");}}
@PostMapping("/uploadGroupSignature")
public ResponseEntity<String> uploadSignature(@RequestParam("file") MultipartFile
file) {
try {
fileStorageService.saveFilePath(file,"D:\\java bach
project\\Files\\VerifyFolder","signatureToVerify.sign");
return ResponseEntity.ok("File uploaded successfully.");
} catch (IOException e) {
log.log(Level.SEVERE, "Exception during upload", e);
return ResponseEntity.status(500).body("Upload failed.");}}
@GetMapping("/getVerification")
public ResponseEntity<String> getVerification() {
String fileName = "D:\\java bach project\\Files\\VerifyFolder";
Element T1,  T2,  T3,  c,
s_alpha,  s_beta,  s_x,
s_delta1,  s_delta2;
Element g1 ,g2, h, u, v, omega ;
try (BufferedReader reader = new BufferedReader(new FileReader(
"D:\\java bach project\\Files\\VerifyFolder\\signatureToVerify.sign"))) {
byte[] T1Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] T2Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] T3Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] cBytes = Base64.getDecoder().decode(reader.readLine());
byte[] s_alphaBytes = Base64.getDecoder().decode(reader.readLine());
byte[] s_betaBytes = Base64.getDecoder().decode(reader.readLine());
byte[] s_xBytes = Base64.getDecoder().decode(reader.readLine());
byte[] s_delta1Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] s_delta2Bytes = Base64.getDecoder().decode(reader.readLine());
T1 = bp.getG1().newElementFromBytes(T1Bytes).getImmutable();
T2= bp.getG1().newElementFromBytes(T2Bytes).getImmutable();
T3= bp.getG1().newElementFromBytes(T3Bytes).getImmutable();
c= bp.getZr().newElementFromBytes(cBytes).getImmutable();
s_alpha= bp.getZr().newElementFromBytes(s_alphaBytes).getImmutable();
s_beta= bp.getZr().newElementFromBytes(s_betaBytes).getImmutable();
s_x= bp.getZr().newElementFromBytes(s_xBytes).getImmutable();
s_delta1= bp.getZr().newElementFromBytes(s_delta1Bytes).getImmutable();
s_delta2= bp.getZr().newElementFromBytes(s_delta2Bytes).getImmutable();
} catch (IOException e) {
throw new RuntimeException(e);}
String stringFile = "";
fileName = "D:\\java bach project\\Files\\VerifyFolder\\fileToVerify.txt";
```

```
try {
stringFile = new String(Files.readAllBytes(Paths.get(fileName)),
java.nio.charset.StandardCharsets.UTF_8);
} catch (IOException e) {
System.err.println("Error reading file '" + fileName + "': " + e.getMessage());
e.printStackTrace(); }
try (BufferedReader reader = new BufferedReader
(new FileReader("D:\\java bach
project\\Files\\SignFolder\\GroupPublicKey.gbkey"))) {
byte[] g1Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] g2Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] hBytes =Base64.getDecoder().decode(reader.readLine());
byte[] uBytes = Base64.getDecoder().decode(reader.readLine());
byte[] vBytes = Base64.getDecoder().decode(reader.readLine());
byte[] omegaBytes = Base64.getDecoder().decode(reader.readLine());
g1 = bp.getG1().newElementFromBytes(g1Bytes).getImmutable();
g2 = bp.getG2().newElementFromBytes(g2Bytes).getImmutable();
h = bp.getG1().newElementFromBytes(hBytes).getImmutable();
u = bp.getG1().newElementFromBytes(uBytes).getImmutable();
v= bp.getG1().newElementFromBytes(vBytes).getImmutable();
omega = bp.getG2().newElementFromBytes(omegaBytes).getImmutable();
} catch (IOException e) {
throw new RuntimeException(e);}
boolean isVerified =  bbs.verify( stringFile , bp,  T1,  T2,  T3,  c,
s_alpha,  s_beta,  s_x,
s_delta1,  s_delta2 ,  u,
v,  g1,  g2, h,  omega , Zr);
if (isVerified) {
return ResponseEntity.ok("Verification status: True");
} else {
return ResponseEntity.ok("Verification status: False");}}
@PostMapping("/uploadManagerKey")
public ResponseEntity<String> uploadManagerKey(@RequestParam("file") MultipartFile
file) {
try {
fileStorageService.saveFilePath(file,"D:\\java bach
project\\Files\\OpenSignatureFolder",
"GroupManagerKey.mkey");
return ResponseEntity.ok("File uploaded successfully.");
} catch (IOException e) {
log.log(Level.SEVERE, "Exception during upload", e);
return ResponseEntity.status(500).body("Upload failed.");}}
@GetMapping("/openSignature")
```

```
public ResponseEntity<String> openSignature() {
Element T1,  T2,  T3,  c,
s_alpha,  s_beta,  s_x,
s_delta1,  s_delta2;
Element A;
Element xi_1;
Element xi_2;
try (BufferedReader reader = new BufferedReader(new FileReader(
"D:\\java bach project\\Files\\VerifyFolder\\signatureToVerify.sign"))) {
byte[] T1Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] T2Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] T3Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] cBytes = Base64.getDecoder().decode(reader.readLine());
byte[] s_alphaBytes = Base64.getDecoder().decode(reader.readLine());
byte[] s_betaBytes = Base64.getDecoder().decode(reader.readLine());
byte[] s_xBytes = Base64.getDecoder().decode(reader.readLine());
byte[] s_delta1Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] s_delta2Bytes = Base64.getDecoder().decode(reader.readLine());
T1 = bp.getG1().newElementFromBytes(T1Bytes).getImmutable();
T2= bp.getG1().newElementFromBytes(T2Bytes).getImmutable();
T3= bp.getG1().newElementFromBytes(T3Bytes).getImmutable();
c= bp.getZr().newElementFromBytes(cBytes).getImmutable();
s_alpha= bp.getZr().newElementFromBytes(s_alphaBytes).getImmutable();
s_beta= bp.getZr().newElementFromBytes(s_betaBytes).getImmutable();
s_x= bp.getZr().newElementFromBytes(s_xBytes).getImmutable();
s_delta1= bp.getZr().newElementFromBytes(s_delta1Bytes).getImmutable();
s_delta2= bp.getZr().newElementFromBytes(s_delta2Bytes).getImmutable();
} catch (IOException e) {
throw new RuntimeException(e);}
try (BufferedReader reader = new BufferedReader(new FileReader("D:\\java bach
project\\Files\\OpenSignatureFolder\\GroupManagerKey.mkey"))) {
byte[] xi_1Bytes = Base64.getDecoder().decode(reader.readLine());
byte[] xi_2Bytes = Base64.getDecoder().decode(reader.readLine());
xi_1 = bp.getZr().newElementFromBytes(xi_1Bytes).getImmutable();
xi_2 = bp.getZr().newElementFromBytes(xi_2Bytes).getImmutable();
} catch (IOException e) {
throw new RuntimeException(e);}
Element A_ = bbs.openSignature(T1, T2, T3,xi_1,xi_2);
System.out.println("Recovered A: " + A_);
return ResponseEntity.ok("Recovered A in bytes: "
+Base64.getEncoder().encodeToString(A_.toBytes()) );
}}
```

Додаток В

ІЛЮСТРАТИВНА ЧАСТИНА
СИСТЕМА ЗАХИЩЕНОГО ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

Виконав: студент 4 курсу групи 1БКС-21б
спеціальності 125 Кібербезпека

_____ Дмитро ЛОБАЙ
16 червня 2025 р

Керівник: к. т. н., доцент каф. ЗІ

_____ Юрій БАРИШЕВ
16 червня 2025 р

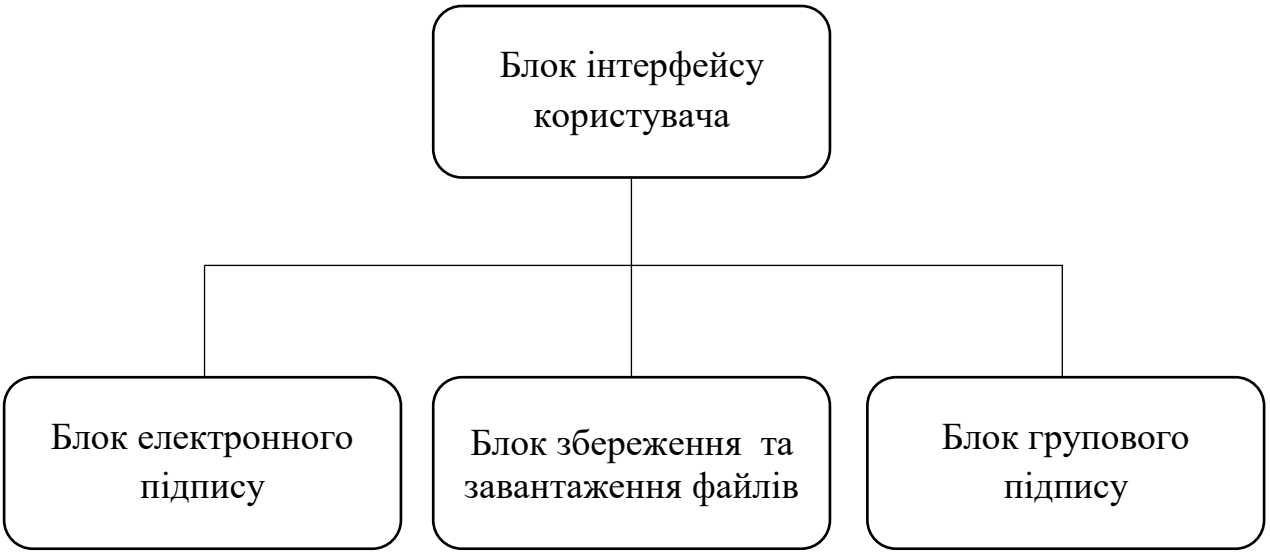
Порівняльний аналіз систем електронного документообігу

№	Назва СЕД	Вартість на рік за користувача	Інтеграція	Функціональні особливості
1	FlyDoc	4650 до 9300 грн	BAS та інш.	Має найкращу інтеграцію із BAS, має весь необхідний функціонал який потрібний для виконання задач на підприємстві.
2	M.E.Doc	Від 1900 до 2150 грн, додаткові модулі придбаються окремо.	BAS та інш.	Для кожного виду користувача має окремі рішення, які будуть необхідні для їх роботи, є можливість придбання додаткових модулів, для забезпечення специфічних потреб.
3	СОТА	700 до 3300 грн; додаткові модулі придбаються окремо.	BAS та інш.	Своїми можливостями досить схожа до M.E.Doc, однак має менше функцій.
4	Box	Від 18 до 45 євро	понад 1500 застосунків	Має кращі інструменти одночасної роботи та високим рівнем інтеграції із різними бізнес системами
5	DocuWare	Від 25 до 100 доларів	понад 500 застосунків	Вирізняється потужними інструменти автоматизації робочих процесів
6	Revverdocs	Від 15 до 50 доларів	Microsoft Office, Salesforce, Dropbox, Google Drive	Спрямована на зручність у роботі для користувачів, має потужні інструменти автоматизації без використання коду

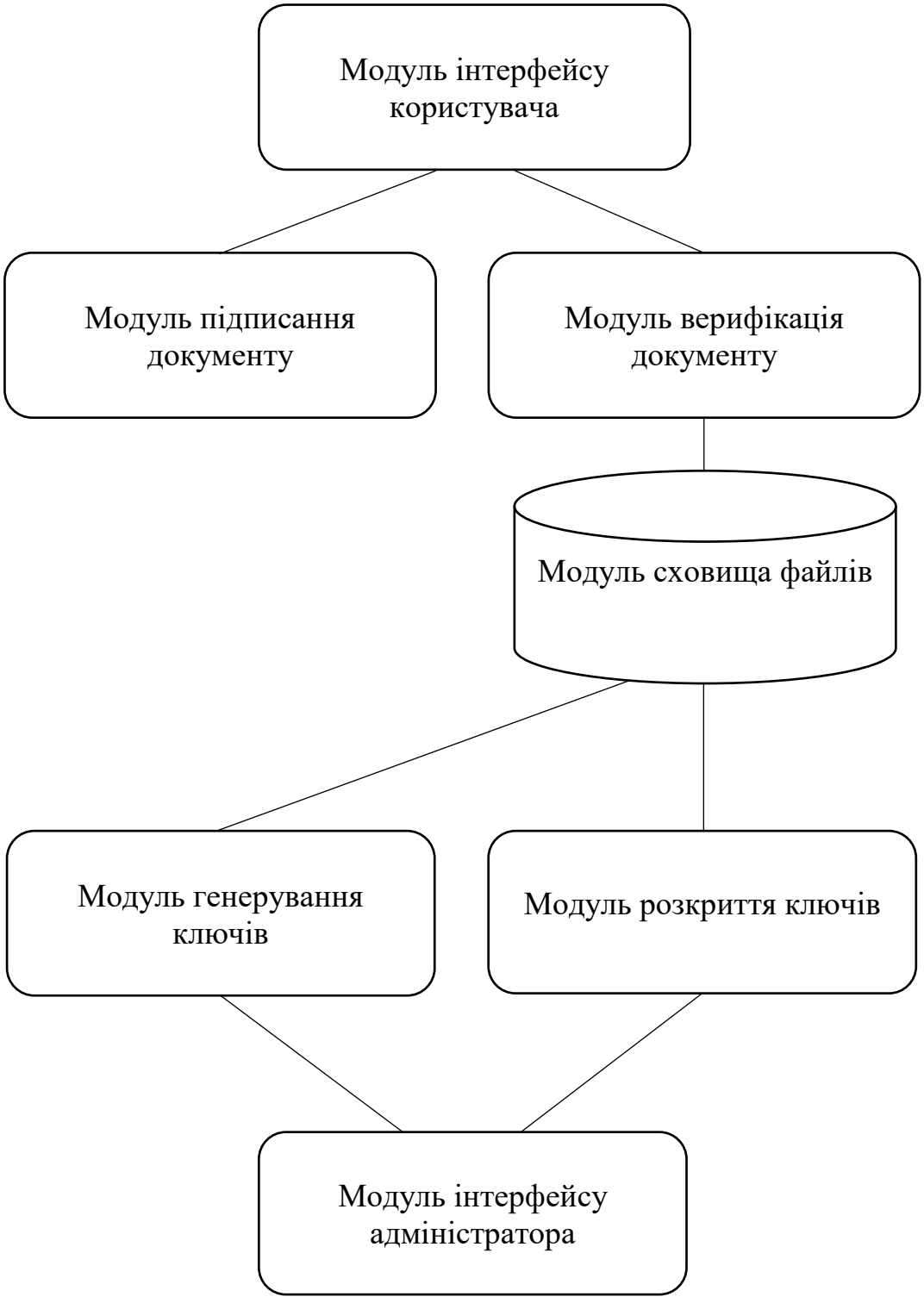
Методи групового підпису

№	Назва схеми	Анонімність	Ідентифікації підписанта	Особливість	Призначення
1	Group Signatures with Selective Linkability	Присутня можливість визначення, що підписи належать одному учаснику	Відсутня	Контрольоване порушення анонімності	Для сценаріїв, де важлива анонімність, але потрібно виявляти повторні дії одного учасника
2	Short Group Signatures	Повна анонімність з можливістю відкриття підпису	Присутня	Короткий розмір і простота обчислень	Для систем, де потрібна ефективність із збереження властивостей ГП
3	Short Randomizable Signatures	підпис можна змінити без зміни змісту	Присутня	Рандомізація підпису при збереженні його дійсності	Для підвищення конфіденційності в середовищах, де підпис потрібно рандомізувати

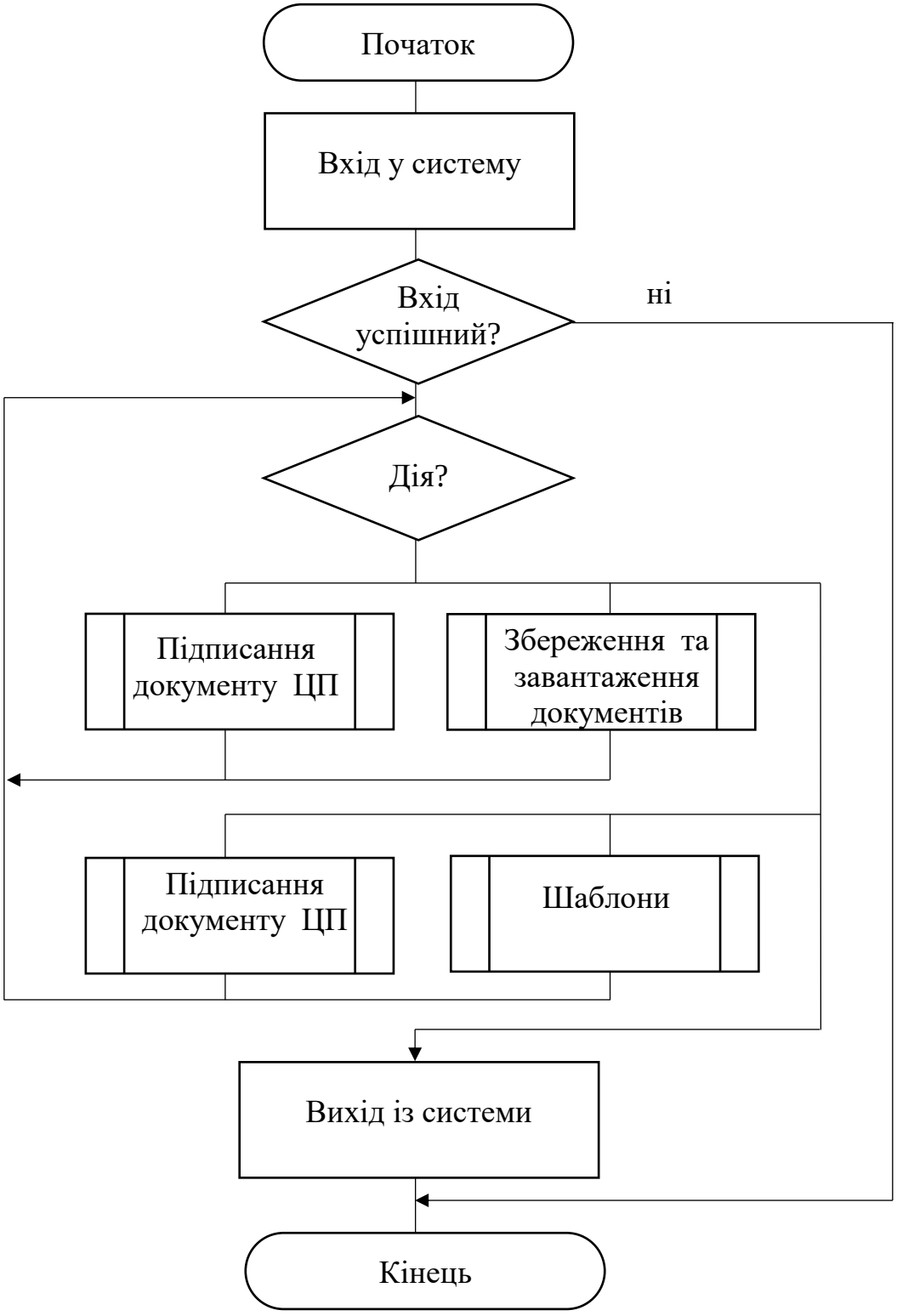
Узагальнена архітектура



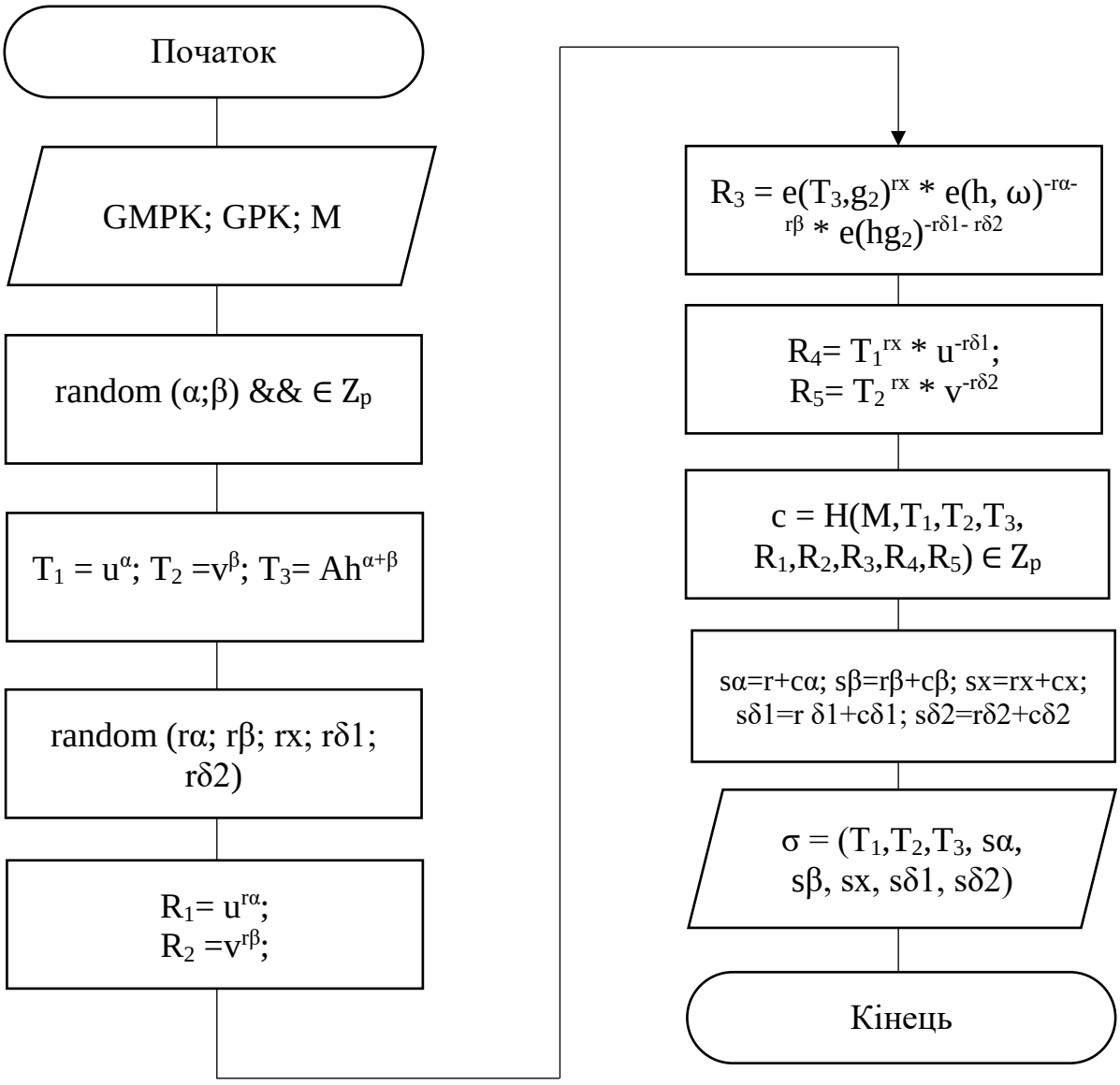
Блок групового підпису



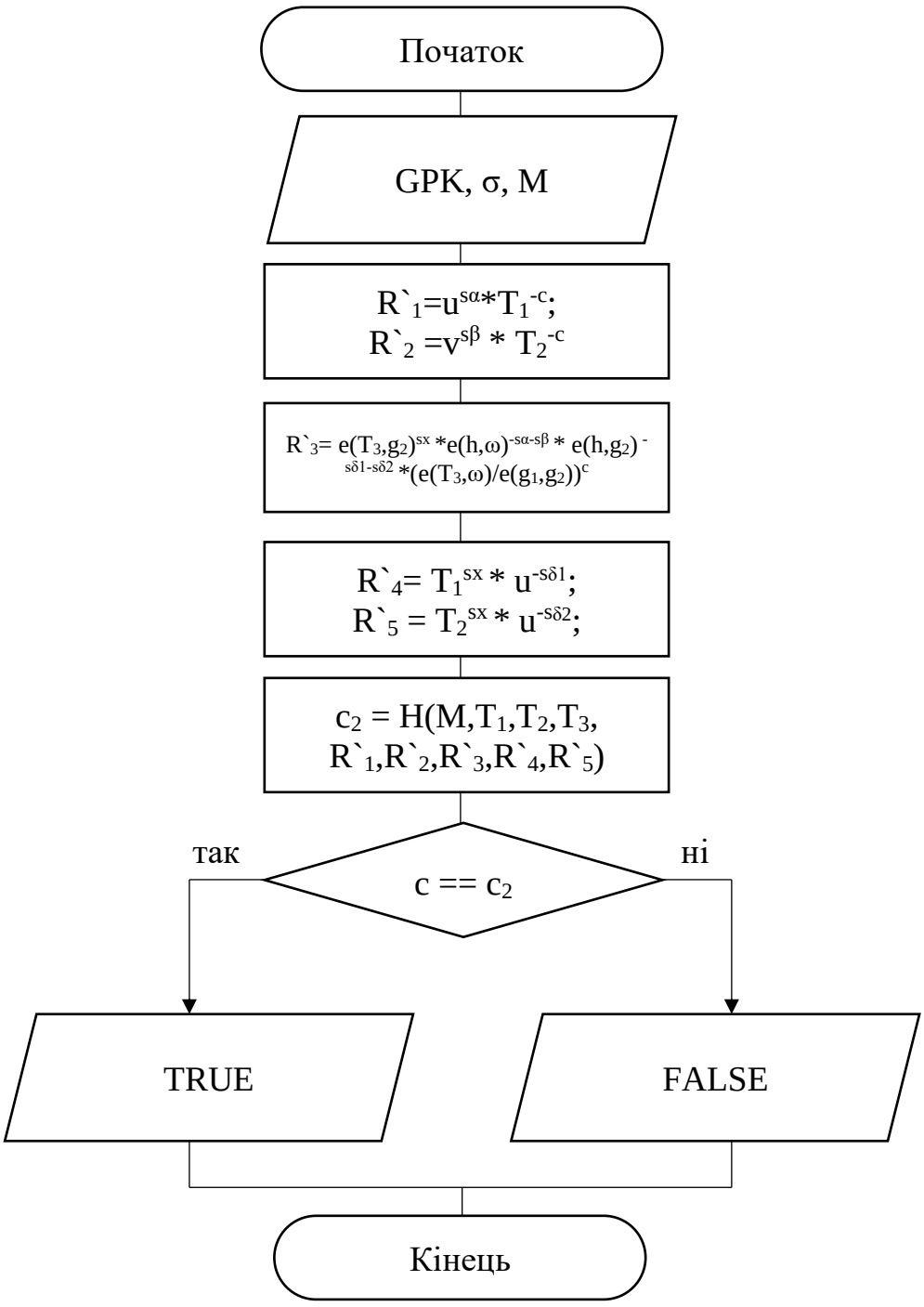
Узагальнений алгоритм



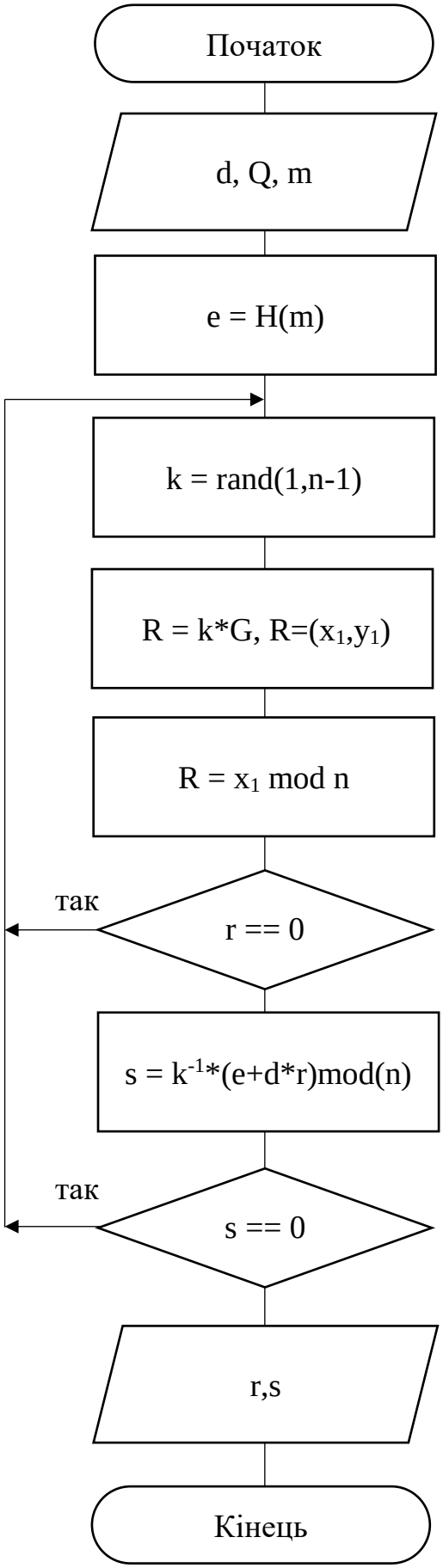
Алгоритм формування групового підпису



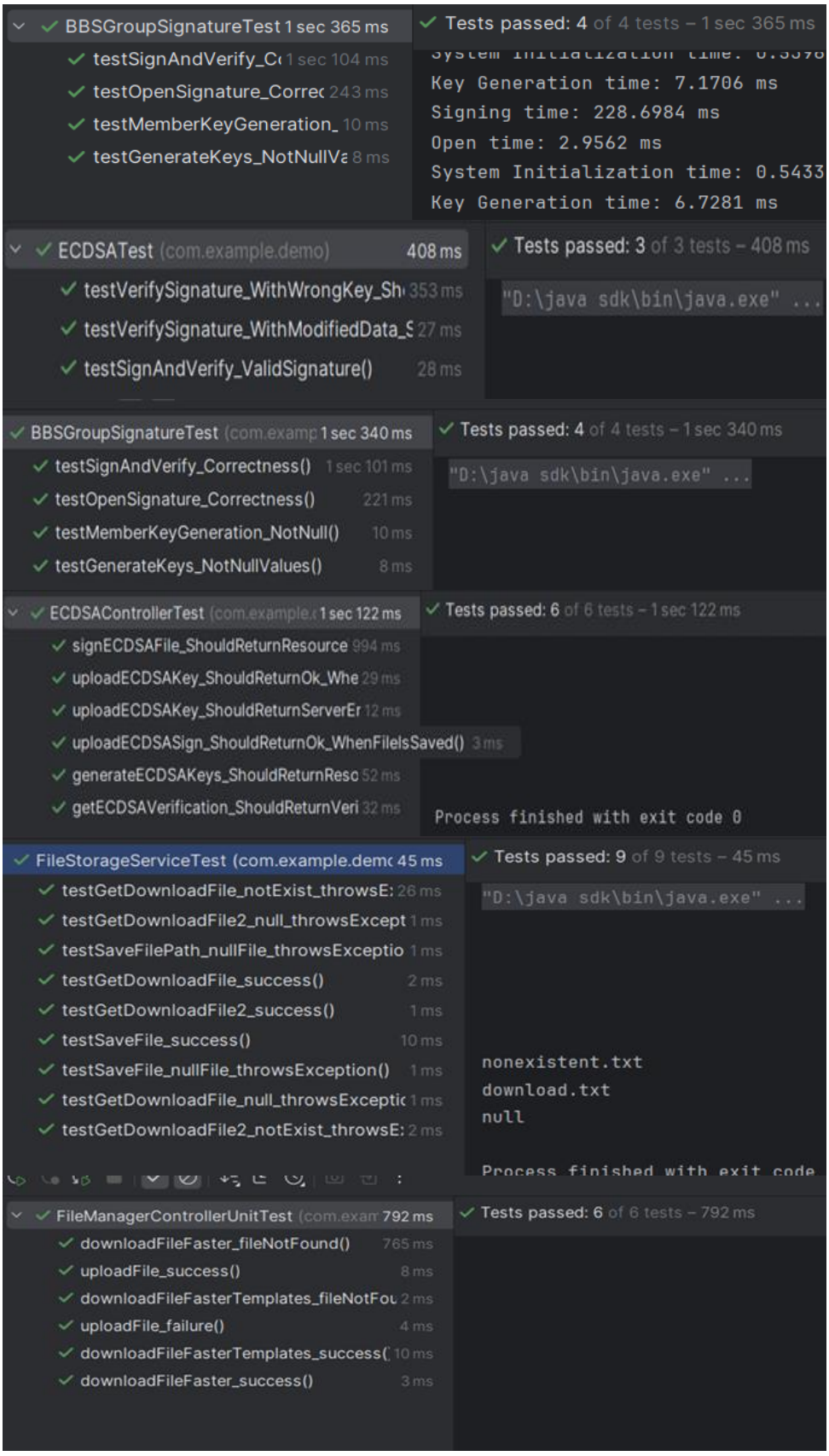
Алгоритм перевірки групового підпису



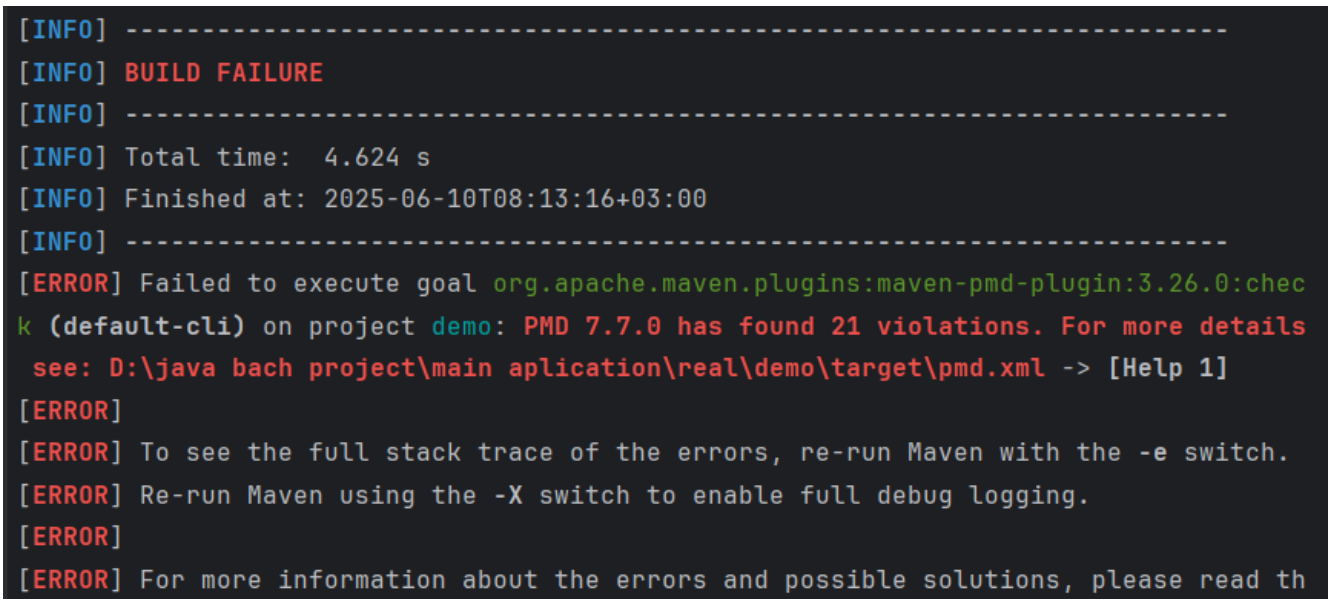
Алгоритм ЕЦП



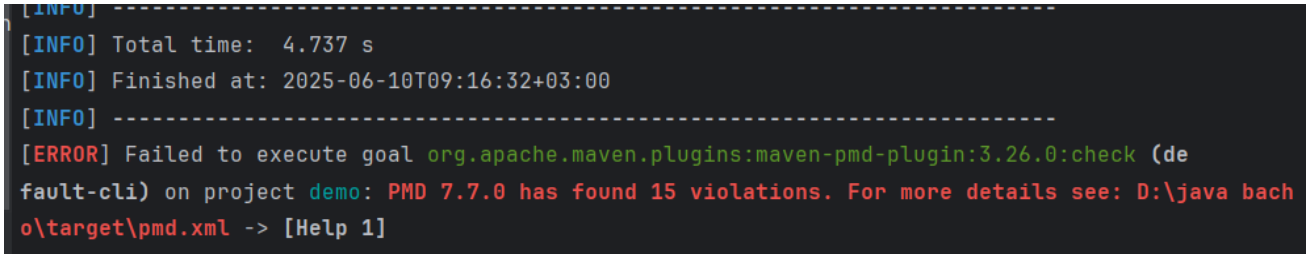
Результати модульного тестування



Результати статичного тестування безпеки



Результати первинного тестування



Результати тестування після виправлення недоліків