Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри кібербезпеки та захисту інформації
_____ Іван ПАРХОМЕНКО
«____» червня 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітній ступень бакалавр

освітня програма Кібербезпека
(назва освітньо-професійної програми)

на тему: Засіб захисту даних на базі алгоритму з відкритим ключем

Виконавець: студент IV курсу, групи КБ-41

_____ **Анатолій ЛІНІЙЧУК**
(підпис) (ім'я, прізвище)

	Ім'я, прізвище	**Підпис**
Керівник	Микола БРАІЛОВСЬКИЙ	

Нормоконтроль	Юрій ЩЕБЛАНІН	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)
освітньої програми Кібербезпека
(назва освітньо-професійної програми)

Студенту **КБ-41** (група) **Анатолію Олександровичу Лінійчуку** (прізвище ім'я по батькові)

Тема кваліфікаційної роботи Засіб захисту даних на базі алгоритму з відкритим ключем

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №3 від 20.10.2022 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Концепція захисту даних, алгоритми шифрування з відкритим ключем

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Проаналізувати існуючі методи захисту даних, дослідити принципи роботи кваліфікаційного електронного підпису його роль у забезпечені інформаційної безпеки, реалізувати алгоритм DSA,розробити програмний модуль для який буде виконувати функціонал захисту даних

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розроблений програмний модуль для захисту даних з використанням алгоритму з відкритим ключем

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 24 жовтня 2022 року

Завдання видав _____ (підпис) Микола БРАІЛОВСЬКИЙ (ім'я, прізвище)

Завдання прийняла до виконання _____ (підпис) Анатолій ЛІНІЙЧУК (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	24.10.2022 – 22.01.2023	*виконано*
2	Аналіз літератури	23.01.2023 – 11.02.2023	*виконано*
3	Обґрунтування вибору рішення	12.02.2023 – 15.02.2023	*виконано*
4	Аналіз існуючих методів захисту даних	16.02.2023 – 04.03.2023	*виконано*
5	Аналіз видів шифрування для КЕП	05.03.2023 – 21.03.2023	*виконано*
6	Програмна реалізація обраного алгоритму шифрування	22.03.2023 – 08.04.2023	*виконано*
7	Аналіз створеного додатку для захисту даних	09.04.2023 – 10.05.2023	*виконано*
8	Оформлення пояснювальної записки	11.05.2023 – 27.05.2023	*виконано*
9	Підготовка до захисту кваліфікаційної роботи	28.05.2023 – 12.06.2023	*виконано*

Завдання видав _____ (підпис) Микола БРАІЛОВСЬКИЙ (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) Анатолій ЛІНІЙЧУК (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 12 червня 2023 року

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, додатків, має 59 сторінок основного тексту, 27 рисунків, 2 таблиці та 13 формул. Список використаних джерел містить 19 найменування і займає 2 сторінку.

Методи дослідження кваліфікаційної роботи:

- аналіз літератури;
- аналіз документів;
- порівняння;

Об'єктом дослідження є процес захисту даних з використанням кваліфікаційного електронного підпису .

Предметом дослідження в даній роботі є методи захисту даних, які базуються на створеному кваліфікаційному електронному підпису.

вивчення та узагальнення вітчизняної і зарубіжної практики. У роботі проаналізована існуюча література з методів захисту інформації, виконаний аналіз документів, порівняння, вивчення та узагальнення вітчизняної і зарубіжної практики з теми захисту інформації.

Розроблений лістинг додатку, що виконує процес створення кваліфікаційного електронного підписку що, може використовуватися користувачами для створення свого підпису, підписання документів та перевірки вже підписаних документів.

Ключові слова: Захист персональних даних, кваліфікаційний електронний підпис, шифр DSA.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

КЕП	–	Кваліфікаційний електронний підпис
ЕП	–	Електронний підпис
RSA	–	Криптографічний алгоритм з відкритим ключем
DSA	–	Digital Signature Algorithm
ECDSA	–	Elliptic Curve Digital Signature Algorithm
ПФЧ	–	Проблема факторизації чисел
ПДЛ	–	Проблема дискретного алгоритму
ПДЛЕК	–	проблема дискретного логарифму в групі точок
SHA	–	Secure Hash Algorithm
SPKI	–	Simple public-key infrastructure
VS	–	Visual Studio
JS	–	JavaScript

ЗМІСТ

ВСТУП

Актуальність теми: У зв'язку з неперервним розвитком інформаційних технологій, вимоги до захисту персональних даних зростають. Це викликає серйозну загрозу для цілісності та конфіденційності інформації, тому необхідно уважно переглянути питання щодо захисту важливих даних. 20 років тому проблема захисту інформації вирішувалась за допомогою криптографічного шифрування, встановлення брандмауерів та обмеження доступу. Однак у сучасний час ці технології стають недостатніми, оскільки будь-яка інформація, яка має фінансову, конкурентну, військову чи політичну цінність, є під загрозою. Більш того, існує ризик перехоплення управління критично важливою інформаційною інфраструктурою..

Належний рівень захисту персональних даних є одним з найважливіших аспектів дотримання політики інформаційної безпеки на підприємствах. Державна політика спрямована на посилення контролю в цій сфері. Згідно з Законом "Про захист персональних даних", законодавство про захист персональних даних формують Конституція України, сам Закон, інші закони та нормативно-правові акти, а також міжнародні договори України, на які Верховна Рада України надала згоду щодо їх обов'язковості.

Метою роботи є створення додатку який дозволить виконувати процес захисту даних, який використовує кваліфікаційних електронний підпис. Для досягнення цієї мети необхідно виконати такі завдання:

- Проаналізувати методи захисту даних які існують на даний момент.
- Дослідити всі існуючі методи захисту даних з використанням кваліфікаційного електронного підпису.
- Розробити програмний модуль для кваліфікаційного електронного підпису.

Об'єктом дослідження - процес захисту даних з використанням кваліфікаційного електронного підпису.

Предмет дослідження - методи захисту даних, які базуються на створеному кваліфікаційному електронному підпису.

Галузь застосування -полягає у використанні та впровадженні суб'єктами інформаційної безпеки з метою захисту цілісності даних.

Практична значимість даної роботи полягає в розробці програмного криптографічного алгоритму DSA з використанням інтегрованого середовища розробки для мов програмування C/C++ та редактора вихідних кодів Visual Studio Code. Це дозволить забезпечити цілісність інформаційних ресурсів.

Розроблений програмний криптографічний алгоритм DSA буде мати практичне застосування в сфері інформаційної безпеки. Використання інтегрованого середовища розробки для мов програмування C/C++ та редактора вихідних кодів Visual Studio Code дозволить зручно розробляти та вдосконалювати цей алгоритм.

Результати роботи мають велике значення для забезпечення цілісності інформаційних ресурсів та зміцнення систем захисту даних. Використання розробленого програмного криптографічного алгоритму DSA та інтегрованого середовища розробки сприятиме покращенню безпеки і забезпеченню конфіденційності даних у різних сферах, де важлива цілісність інформації.

РОЗДІЛ 1
МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ

Забезпечення безпеки інформації відіграє надзвичайно важливу роль у суспільстві. Це можуть бути паперові або магнітні носії з графічними або буквено-цифровими даними. Голосова інформація виникає під час комунікації та звукового відтворення. Телекомунікаційна інформація пов'язана зі зберіганням та передачею даних по каналах зв'язку, де носієм є електричний струм або електромагнітні хвилі при використанні оптичних або радіоканалів. Однак, незалежно від форми та носія, інформацію необхідно забезпечувати стислою формою та вживати заходів для її захисту від витоку.

1.1 Основні методи захисту інформації

Основними об'єктами інформації можуть бути:

Інформаційні ресурси, які можуть включати державну таємницю та конфіденційні дані.

Засоби та системи інформатизації, такі як системи, мережі, інформаційні та комп'ютерні системи, а також програмне забезпечення, таке як операційні системи, системи управління базами даних та інші типи програмного забезпечення. Ці засоби використовуються для обробки, передачі та зберігання даних, включаючи "закриту" інформацію, схема яких зображена на Рисунку 1.1.

Технічні методи і системи, які застосовуються на місці обробки конфіденційних даних і виконують допоміжні функції.

Найпоширенішими методами захисту даних є:

- Фізичне обмеження, що передбачає фізичну блокаду шляху до захищеного носія інформації.
- Контроль доступу, що забезпечує захист інформаційних ресурсів

шляхом контролю використання кожного ресурсу, включаючи апаратне та програмне забезпечення та елементи баз даних.

– Шифрування, що полягає в криптографічному захисті даних, зокрема при передачі довгих каналів, і вважається надійним методом.
– Регламентація, що передбачає захист інформації та даних шляхом встановлення правил і процедур, що мінімізують можливість несанкціонованого доступу.
– Примус, що полягає в забезпеченні дотримання правил поводження зі захищеною інформацією, включаючи передачу, обробку та використання, і може мати адміністративну або матеріальну відповідальність за порушення цих правил

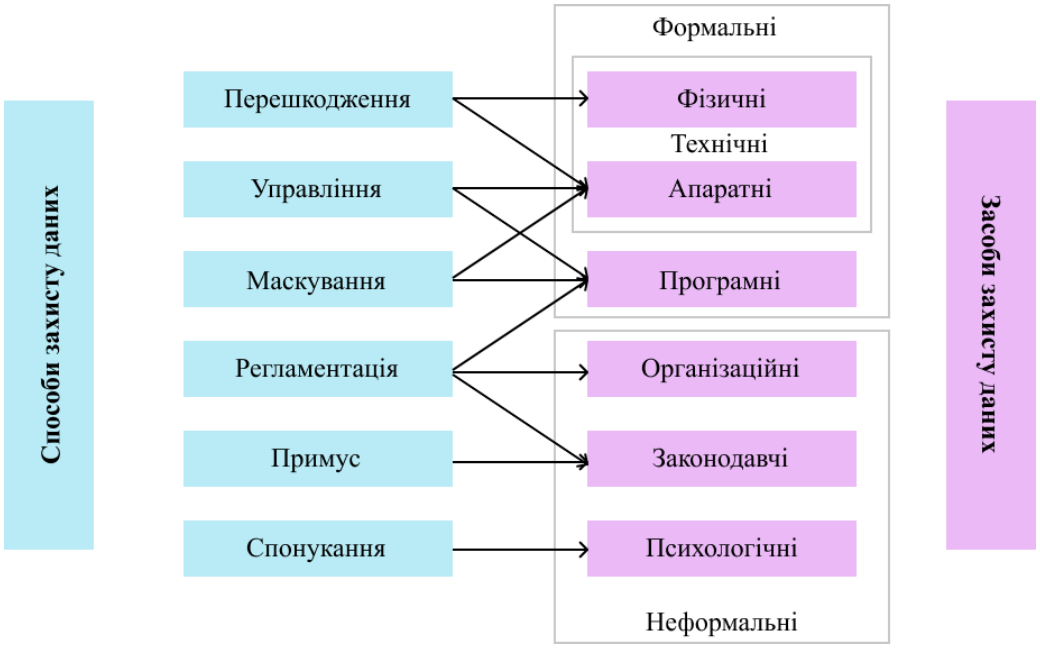


Рисунок 1.1 - Основні методи захисту інформації

1.2 Методи захисту даних, які обмежують доступ

Основними аспектами захисту інформації в комп'ютерних мережах є:

Ідентифікація ресурсів комп'ютерної мережі, користувачів і персоналу шляхом присвоєння персональних ідентифікаторів. Це дозволяє встановити, хто

має доступ до яких ресурсів.

Перевірка ідентифікатора при вході в систему для підтвердження справжності об'єкта. Цей метод дозволяє перевірити, чи є особа або ресурс дійсним, використовуючи надані при реєстрації дані.

Контроль повноважень, який враховує відповідність часу, ресурсів і запитуваних процедур встановленим регламентом. Це забезпечує дотримання правил доступу та використання ресурсів у встановлені години роботи.

Встановлення регламенту, що визначає робочий час і дозволяє визначений діапазон роботи.

Реєстрація кожного доступу до захищених ресурсів, що дозволяє зберігати записи про всі взаємодії з цими ресурсами.

Реакція на спроби несанкціонованого доступу, такі як відмова в запиті або активація сигналізації. Це допомагає виявити та обмежити незаконні дії.

Захист інформації має велике значення в різних сферах життя, існує багато різноманітних методів, які потребують уважного застосування для забезпечення безпеки.

1.3 Організаційні методи захисту інформації

Служба безпеки повинна мати навички розробки комплексу організаційних заходів для захисту інформації. Зазвичай вони використовують такі методи:

- Створення внутрішніх нормативних документів, які встановлюють правила роботи з комп’ютерною технікою та конфіденційною інформацією
- У рамках збереження конфіденційних даних проводяться періодичні перевірки персоналу та організуються навчання з питань збереження конфіденційності. Крім того, можуть укладатися додаткові угоди до трудового договору, які чітко визначають відповідальність

працівника за неправомірне використання або розголошення інформації..

- Розмежування зон відповідальності для уникнення ситуацій, коли вся цінна інформація знаходиться під контролем однієї особи. Крім цього, служба безпеки повинна сприяти використанню загальних програм документообігу та переконатися, що файли особливого значення зберігаються на мережевих дисках.
- Використання програмних комплексів, які захищають інформацію від знищення або копіювання будь-яким користувачем системи, включаючи топ-менеджерів компанії.

Розробка планів відновлення системи в разі її відмови.

1.4 Програмні та апаратні методи захисту даних

Основні методи захисту інформації включають використання програмних і апаратних засобів. Серед них можна виділити наступні:

- Застосування віддаленого зберігання та регулярного резервного копіювання найважливіших інформаційних даних.
- Резервне копіювання та дублювання всіх підсистем, що містять значну інформацію.
- Розподіл ресурсів мережі для відновлення її функціонування в разі виникнення проблем з окремими її елементами.
- Застосування резервних систем живлення, що забезпечують безперебійне живлення.
- Забезпечення безпеки інформаційних даних та належного захисту у разі пожежі або збитків, спричинених водою, пошкодженню комп'ютерного обладнання.
- Використання програмного забезпечення, яке забезпечує належний захист інформаційних баз даних від несанкціонованого доступу.

1.5 Криптографічний метод захисту даних

Метою використання криптографічних методів є захист інформаційної системи від цілеспрямованих руйнівних впливів (атак) з боку зловмисників. Основні завдання криптографії включають:

Забезпечення конфіденційності даних: це означає запобігання несанкціонованому доступу до даних. Для досягнення цієї мети використовується шифрування даних, що перетворює їх таким чином, що тільки користувачі з відповідним ключем можуть прочитати їх.

Забезпечення цілісності даних: це гарантує, що дані не були змінені незаконними користувачами під час передачі або зберігання. Це охоплює виявлення будь-яких змін, видалень, замін або повторного пересилання даних, які були піддані несанкціонованим змінам.

Забезпечення аутентифікації: це перевірка суб'єктів або даних на їхню відповідність реальності або правдивості. При аутентифікації суб'єкта А має бути встановлено не тільки його права, але й запобігнуто можливості перевіряючому суб'єкту В самостійно скористатися отриманою інформацією з метою підробки себе під суб'єкта А.

Забезпечення неможливості відмови від авторства: це запобігання можливості суб'єктів відмовитись від своїх дій, зокрема від підпису інформації. Саме це забезпечує той факт зо авторство документу не можна приписати іншим особам і що ніхто не може якимось чином створити, замінити, твердити, що він був підписаний. Для досягнення цієї мети використовується цифровий підпис.

Отже, криптографічні методи допомагають забезпечити захист інформації, забезпечуючи конфіденційність, цілісність, аутентифікацію та неможливість відмови від авторства.

1.6 Кваліфікаційних електронний підпис

Кваліфікаційних електронний підпис (КЕП) є методом аутентифікації та підтвердження цілісності електронних даних. Цей підпис отримується шляхом кодування або шифрування набору електронних даних за допомогою криптографічних методів та його поєднання з особистим ключем. Для перевірки підпису використовується відкритий ключ. Надійний засіб кваліфікаційного електронного підпису має сертифікат відповідності або позитивний експертний висновок, отриманий в результаті державної експертизи в галузі криптографічного захисту інформації. КЕП використовується фізичними та юридичними особами для аутентифікації підписувача та підтвердження цілісності електронних даних. Він дозволяє визначити походження інформації та розмежувати відповідальність за неї. Кваліфікаційних електронний підпис надає юридичну силу електронному документу, згідно з українським законодавством. Цей підпис забезпечує контроль за походженням та цілісністю інформації і є важливим інструментом для забезпечення безпеки інформації на різних рівнях.

КЕП встановлюється за допомогою закритого ключа та перевіряється за допомогою відкритого ключа. Зберігання приватного ключа в безпеці надає непідробність. Додатково, будь-які несанкціоновані зміни в електронному документі будуть виявлені. Генерація приватного ключа КЕП з використанням випадкових чисел, а обчислення відкритого ключа з приватного ключа використовуються для цього. Закритий ключ є унікальним та потребує конфіденційного зберігання, оскільки він дозволяє створювати кваліфікаційних електронний підпис.

Документ підписується за допомогою приватного ключа КЕП, який належить виключно його власнику і є унікальним. Цей приватний ключ відповідає публічному ключу, за допомогою якого можна перевірити, чи належить підписаний документ його власнику.

Застосування кваліфікованого електронного підпису (КЕП) має кілька

переваг:

- Зменшує час, що витрачається на транзакцію та обмін документами, завдяки швидкому та ефективному електронному підпису.

- Покращує та знижує витрати на процес підготовки, доставки, обліку та зберігання документів, оскільки фізична обробка та утримання паперових документів стають незначними.

- Забезпечує достовірність документації, оскільки підписані електронні документи мають перевірену автентичність та непереборну зв'язаність з власником підпису.

- Зменшує ризик фінансових втрат, збільшуючи конфіденційність обміну інформацією та унеможливлюючи несанкціонований доступ до документів.

- Допомагає побудувати ефективну систему обміну документами в межах компанії, сприяючи швидкому та безпечному передаванню електронних документів між співробітниками та сторонами.

Застосування КЕП має значний потенціал для оптимізації бізнес-процесів та поліпшення безпеки та ефективності обміну документами.

Узагальнюючи, КЕП є ефективним засобом для аутентифікації, підтвердження цілісності та забезпечення безпеки електронних даних. Він має юридичну силу і дозволяє контролювати походження інформації в електронному вигляді.

1.7 Види кваліфікаційного електронного підпису

Існують три види електронних цифрових підписів:

1. простий,
2. вдосконалений некваліфікований
3. вдосконалений кваліфікований.

Простий кваліфікований електронний підпис підтверджує авторство електронного підпису певної особи. Однак, його рівень захисту є низьким і він не забезпечує надійну захисту від підробки, фокусуючись лише на ідентифікації автора документа.

Вдосконалений некваліфікований кваліфікаційних електронний підпис отримується за допомогою криптографічних перетворень та надає можливість ідентифікувати особу, яка підписала документ, а також виявити зміни після підписання.

Вдосконалений кваліфікований кваліфікаційних електронний підпис є найбільш універсальним і стандартизованим. Він має високий рівень захисту і дозволяє вказати ключ для перевірки підпису у кваліфікованому сертифікаті. Документ, підписаний таким підписом, аналогічний паперовому документу з власноручним підписом. Використання вдосконаленого кваліфікованого підпису не потребує додаткових домовленостей між учасниками і дозволяє точно вказати автора документа та виявити зміни після підписання.

Отже, простий, вдосконалений некваліфікований і вдосконалений кваліфікований електронні цифрові підписи відрізняються за рівнем захисту та можливостями ідентифікації та виявлення змін

1.8 Застосування різних видів КЕП

Простий КЕП дозволяє юридичним особам подавати заявки на державні та комунальні послуги, підписуючи їх простим ЕП уповноважених осіб. Використання простого ЕП для таких послуг дозволено, якщо закони не забороняють подання заявок в електронній формі та не передбачають іншого виду ЕП.

Вдосконалений некваліфікований КЕП може бути визнаний електронним документом, еквівалентним паперовому документу з власноручним підписом. Однак, для цілей податкового обліку, такий КЕП не є еквівалентним паперовому

документу. Використання вдосконаленого некваліфікованого КЕП може створювати суперечки з контролюючими органами.

Вдосконалений кваліфікований КЕП вимагає підписувати електронну накладну керівником або іншими уповноваженими особами за допомогою посиленого кваліфікованого ЕП. Заяви про реєстрацію (зняття з реєстрації) в податковому органі та претензії щодо повернення або заліку суми податку також повинні бути підтверджені вдосконаленим кваліфікованим ЕП.

Отже, використання різних видів КЕП має свої обмеження та впливає на юридичну силу та еквівалентність електронних документів.

1.9 Кваліфікаційний електронний підпис, як один із методів захисту даних

Електронні документи стають все поширенішими в бізнес-та урядових середовищах, замінюючи традиційні паперові документи. З ростом обсягу інформації, обміну даними та загроз безпеці, захист електронних документів стає все важливішим. Традиційні методи захисту, такі як електронні печатки або власноручні підписи, виявляються недостатніми. В цьому контексті використання кваліфікаційного електронного підпису стає ключовим. Відповідно до українського законодавства, кваліфікаційних електронний підпис має такий же юридичний статус, як рукописний підпис або печатка, і його визнання визначається умовами, зазначеними в законодавстві. Проте використання кваліфікаційного електронного підпису не змінює процедуру підписання договорів та інших документів, передбачених законодавством для письмових операцій.

Кваліфікаційних електронний підпис складається з секретного ключа та сертифіката відкритого ключа. Відкритий ключ перевірки використовується для підтвердження належності відкритого ключа кваліфікаційного електронного підпису конкретній особі. При перевірці КЕП порівнюються дані первинного та

отриманого документів, і результат тесту дає відповідь "true" або "false".

Це зображено на Рисунку 1.2 нижче кваліфікаційного електронного підпису, де видно, що ключі та сертифікати використовуються для підписання та перевірки КЕП.

Кваліфікаційних електронний підпис забезпечує перевірку автентичності

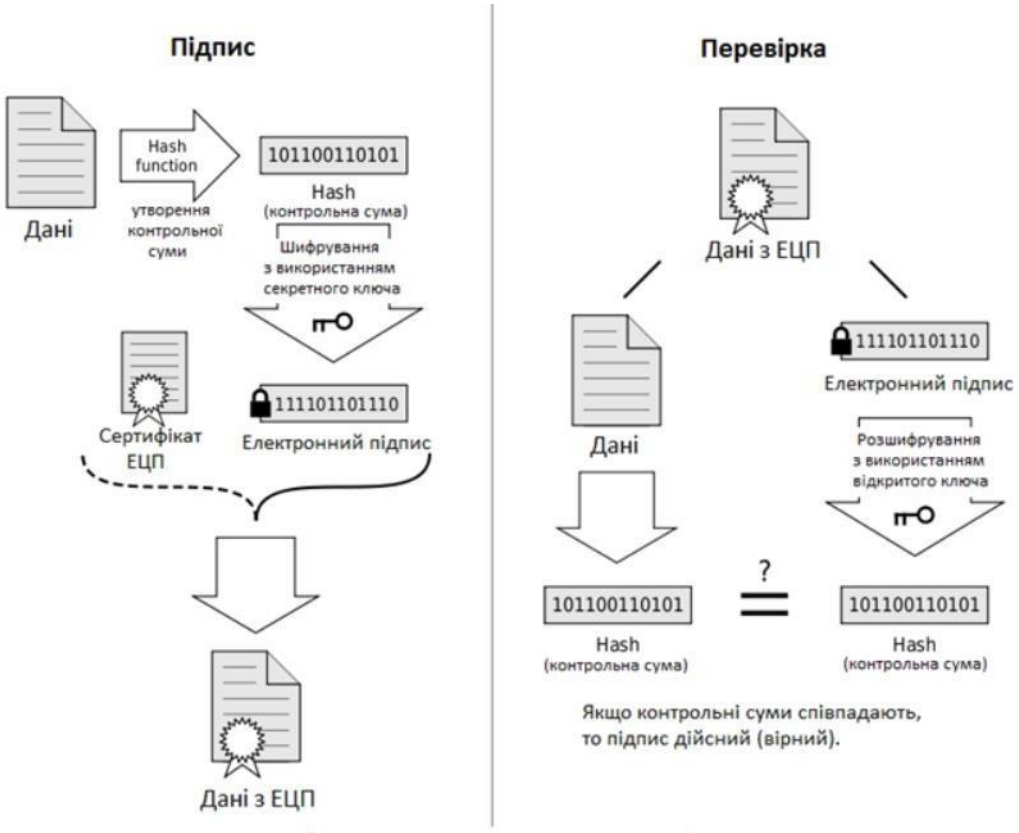


Рисунок 1.2 Схема роботи КЕП

та непорушеності документа. Якщо в процесі передачі даних будь-які зміни були внесені до документа (навіть незначні), це буде помітно, і підпис буде недійсним. Сертифікат відкритого ключа містить особисті дані власника, що дозволяє однозначно ідентифікувати автора документа.

Отже, можна зробити висновок, що кваліфікаційних електронний підпис є невід'ємною частиною сучасного документообігу, оскільки його використання забезпечує високий рівень захисту даних і спрощує різні процедури (наприклад, укладання угод, звітування перед державними органами). Він також дозволяє економити час, забезпечує надійність створених документів і збереження конфіденційної інформації про особу, яка їх створила.

Висновки до першого розділу

У першому розділі дипломної роботи було розглянуто вразливості інформації та різні методи захисту даних. Основні способи включають:

– Методи обмеження доступу для захисту даних.
– Організаційні методи захисту даних.
– Технічні методи захисту даних.
– Криптографічні методи захисту даних.

РОЗДІЛ 2
АНАЛІЗ МЕТОДІВ ШИФРУВАННЯ КЕП

Практичні схеми реалізації КЕП:

- Схема RSA
- Схема Ель-Гамаля
- DSA
- ECDSA

2.1 Алгоритм RSA

RSA є криптографічним алгоритмом з відкритим ключем, який ґрунтується на складності факторизації великих цілих чисел. Цей алгоритм є першим, що дозволяє застосовувати його як для шифрування, так і для кваліфікаційного електронного підпису (КЕП). RSA широко використовується в багатьох криптографічних застосунках.

Генерація ключової пари RSA включає такі кроки:

1) Беруться два простих числа p та q.
2) Обраховується значення $n = p * q$.
3) Береться будь-яке натуральне число

$$e\big(1 < 3 < \varphi(n)\big), \text{НОД}\left(e, \varphi(n)\right) = 1 \quad (2.1)$$

4) Дотримуючись алгоритму Евкліда вирішується рівняння

$$e * d + (p - 1) * (q - 1) * y = 1 \quad (2.2)$$

5) В результаті вирішення рівняння, отримуємо значення закритого ключа d.

Підпис формується згідно рівняння

$$s = M^d mod\ n \quad (2.3)$$

Перевірка підпису

$$M` = s^e mod\ n. \tag{2.4}$$

Якщо $h\,(M) = hash\,(M')$ - КЕП вірний, $h\,(M) \neq hash\,(M')$ - КЕП не вірний. Схема данного алгоритму можна побачити на Рисунку 2.1

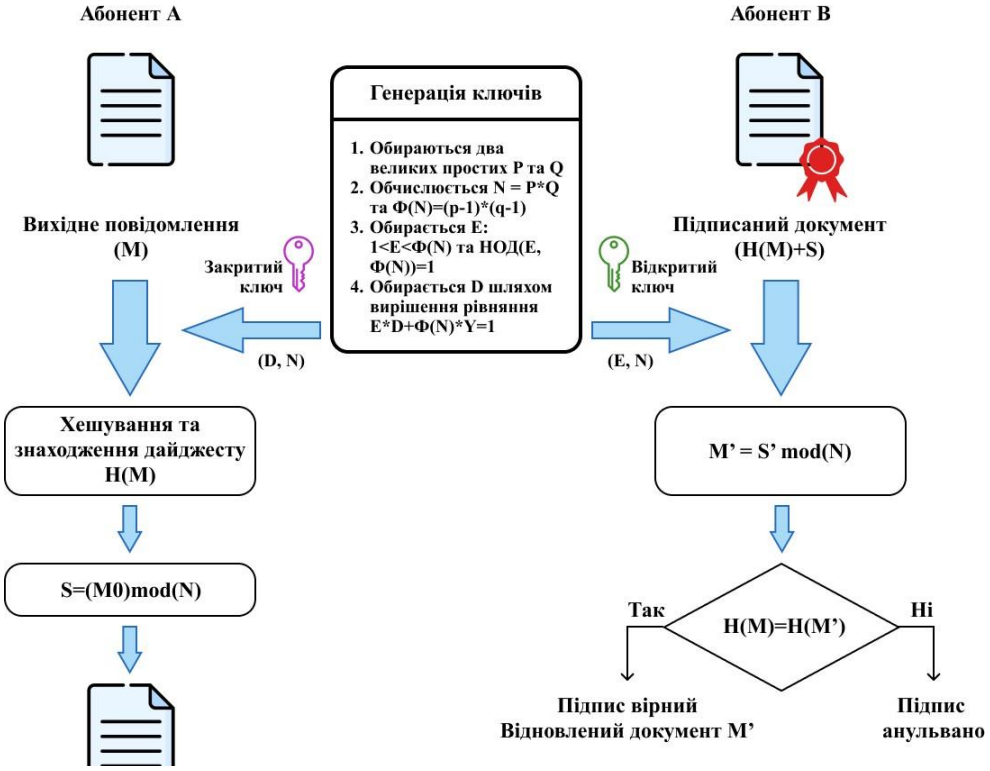


Рисунок 2.1 Схема роботи RSA

2.2 Алгоритм Ельм-Гамеля

Схема Ель-Гамаля — криптосистема з відкритим ключем, яку засновано на складності обчислення дискретних логарифмів у скінченному полі. Криптосистема включає в себе алгоритм шифрування і алгоритм цифрового підпису.

Генерація ключової пари:

1) Обирається будь-яке (правда, досить велике) просте число p.
2) Для нього визначається довільний примітивний (утворюючий) елемент число g.
3) Генерується будь-яке випадкове число x – закритий ключ.

4) Обраховується значення y за формулою 2.5

$$y = gx \, mod \text{ р}. \quad (2.5)$$

5) Комбінація $(g, \text{р}, y)$ є відкритим ключем одержувача.

Формування підпису:

1) Генерується випадкове число k, унікальне для кожного підписання документа, взаємно просте з числом $(\text{р} - 1), \text{НСД}$
2) Визначається значення r за формулою 2.6

$$r = (gk \, mod \text{ р}). \quad (2.6)$$

3) Визначається значення s за формулою 2.7

$$s = ((h - \text{х} * r) / k) \, mod \, (\text{р} - 1), \quad (2.7)$$

де h - контрольна сума документу, що підписується, пара чисел (r, s) є КЕП для документа, що має контрольну суму h.

Перевірка (верифікація) підпису.

1) Обчислюється значення u за формулою 2.8

$$u = ((yr) * (rs) \, mod \text{ р}). \quad (2.8)$$

2) Обчислюється значення h за формулою 2.9

$$h = hash \, (m) \text{ і } v = (gh \, mod \text{ р}) \quad (2.9)$$

3) Перевіряється рівність значень $u = v$ (якщо рівність виконується, то підпис вірний, в іншому випадку документ підроблений).

2.3 Алгоритм DSA

DSA - криптографічний алгоритм з використанням відкритого ключа для створення електронного підпису, але не для шифрування

Параметри:

Відкритий ключ (p, q, g, y) p - просте число довжиною від 512 до 1024 бітів;

q - 160-бітовий простий множник $p - 1$;

$$g = h(p-1)/q \, mod \, p, \quad (2.10)$$

де h - довільне число, $h < (p - 1)$,

для якого $h^{(p-1)/q}\ mod\ p\ >\ 1;$

$$y\ =\ gx\ mod\ p\ (p-\text{бітове число})$$

Закритий ключ (x)

$$1\ <\ x\ <\ q\ (160-\text{бітове число})$$

Формування підпису.

Відправник створює довільне число k, менше q

Відправник створює r та s за формулами 2.11 та 2.12:

$$r\ =\ (g^k\ mod\ p)\ mod\ q \tag{2.11}$$

$$s\ =\ (k^{-1}\ (H(m)\ +\ x\ r\ ))\ mod\ q, \tag{2.12}$$

де $H(m)$ – хеш-функція

Відправник надсилає підпис - (r, s)

Перевірка (верифікація) підпису.

Одержувач перевіряє підпис, роблячи обчислення по формулам 2.13 – 2.16

$$w\ =\ s^{-1}\ mod\ q \tag{2.13}$$

$$u1\ =\ (H(m)\ *\ w)\ mod\ q \tag{2.14}$$

$$u2\ =\ (r\ w)\ mod\ q \tag{2.15}$$

$$v\ =\ ((g^{u1}\ *\ y^{u2}\ )\ mod\ p\ )\ mod\ q \tag{2.16}$$

Якщо $v\ =\ r$, то підпис вірний

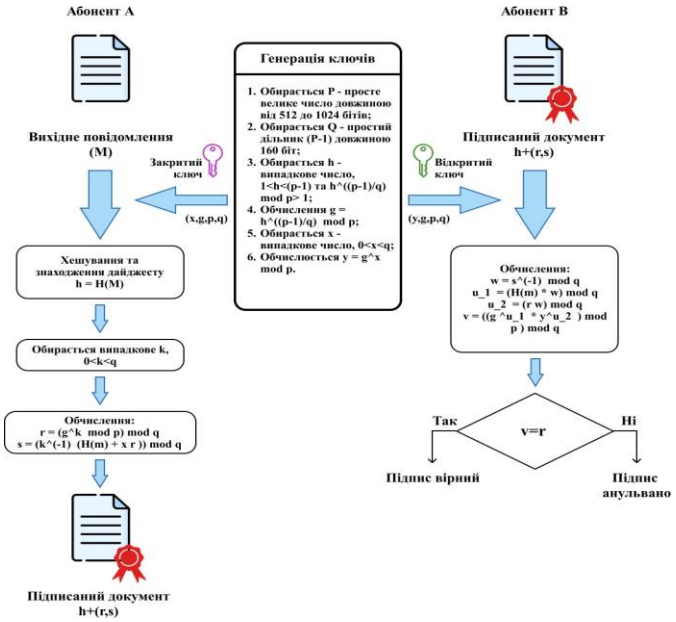


Рисунок 2.2 Схема роботи алгоритму DSA

2.4 Алгоритм ECDSA

ECDSA - алгоритм з відкритим ключем для створення цифрового підпису, аналогічний за своєю будовою DSA, але визначений, на відміну від нього, не над кільцем цілих чисел, а в групі точок еліптичної кривої.

Генерація ключової пари:

1) Вибираємо еліптичну криву E, визначену на Z_p. Кількість точок в $E(Z_p)$ має ділитися на велике ціле n;
2) Обираємо точку $P \in E(Z_3)$ порядку n;
3) Обираємо будь-яке число $d \in [1, n-1]$;
4) Обчислюємо $Q = dP$;
5) Секретним ключем робимо (d, E, P, n), відкритим – $(\mathrm{E}, \mathrm{P}, n, Q)$.

Формування підпису:

1) Обираємо довільне число $k \in [1, n-1]$;
2) Обчислюємо $k\mathrm{P} = (\mathrm{x1}, \mathrm{y1})$ та $r = \mathrm{x1}\ (mod\ n)$. Якщо $r \neq 0$, переходимо до кроку 3, в іншому випадку – повертаємося до кроку 1;
3) Обчислюємо $k^{-l}(mod\ n)$;
4) Обчислюємо $s = [k^{-l}(h(M) + d \times r)] mod\ n$. Якщо $s \neq 0$, переходимо до кроку 5, в іншому випадку – повертаємося до кроку 1;
5) Підписом під повідомленням M є пара цілих чисел (r, s).

Перевірка (верифікація) підпису.

1) Якщо r і s – цілі числа, належать до інтервалу $[1, n-1]$, переходимо до кроку 2, в іншому випадку – результат перевірки є негативний (підпис анулюється);
2) Обчислюємо $w = s^{-l}(mod\ n)$ та $h(M)$;

3) Обчислюємо $u_1 = [h(\mathrm{M}) \times w] \, mod \, n$ та $u_2 = (r \times w) \, mod \, n$;

4) Обчислюємо $u_1 = \mathrm{P} + u_2 Q = (\mathrm{x}_0, \mathrm{y}_0)$ та $v = x_0 \, (mod \, n)$;

5) Підпис є справжній за $v = r$.

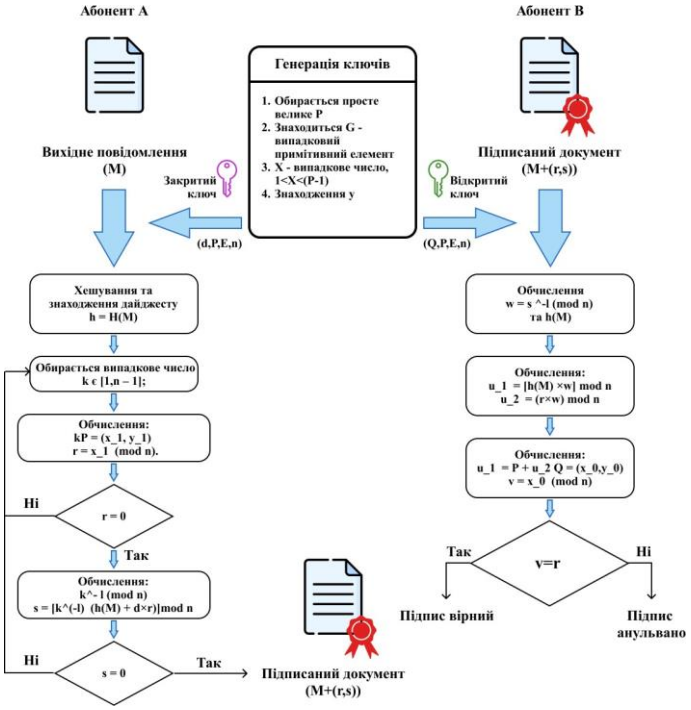


Рисунок 2.3 Схема роботи ECDSA

2.5 Порівняння різних методів шифрування, та їх використання в ЕП

Було проведено аналіз та порівняння всіх згаданих алгоритмів шифрування, результати представлені в таблицях нище.

Таблиця 2.1

Порівняння криптографічних методів

Алгоритм	Проблема	Довжина підпису	Розмір публічного ключа, біт	Хеш-функція	Можливість відновлення повідомлення
ECDSA	ПДЛЕК	Розмір скінченного поля × 4	112-570	SHA	Ні
Ель-Гамаля	ПДЛ	Довжина ключа × 2	1024-4096	MD та SHA	Ні

RSA	ПФЧ	Довжина ключа × 2	1024-4096	MD	Так
DSA	ПДЛ	Розмір скінченного поля × 4	1024-3072	SHA	Ні

Таблиця 2.2.

Порівняння швидкості роботи алгоритмів КЕП

Алгоритм	Розмір ключа, біт	Швидкість підпису, мс	Швидкість перевірки, мс
DSA	1024	0.42	0.52
ECDSA	256	2.88	8.53
RSA	1024 2048	1.48 6.05	0.07 0.16
Ель-Гамаля	1024 2048	0.45 0.83	1.18 3.84

2.6 Вразливості алгоритму DSA

Алгоритм цифрового підпису DSA (Digital Signature Algorithm) є одним з найпоширеніших алгоритмів використовуваних для забезпечення аутентифікації та недережності даних у криптографії. Вперше введений в 1994 році, DSA став стандартом для багатьох криптографічних протоколів та систем. Незважаючи на свою популярність, DSA має кілька вразливостей, які можуть бути використані для злому підписів і порушення безпеки систем, що його використовують.

1. Вразливість побічних каналів

Однією з основних вразливостей алгоритму DSA є його вразливість до атак, пов'язаних з використанням побічних каналів. Побічні канали включають такі елементи як час виконання операцій, споживання енергії, електромагнітні випромінювання та інші. Зловмисники можуть використовувати ці побічні канали для отримання інформації про приватний ключ, використовуваний для генерації цифрового підпису. Знання приватного ключа дозволяє атакуючому створювати фальшиві підписи та підробляти ідентичність автора.

2. Вразливість до атак з використанням криптоаналізу

DSA також має вразливості, пов'язані зі стійкістю своїх математичних параметрів. Атаки з використанням криптоаналізу можуть дозволити зловмиснику відновити приватний ключ або визначити інші конфіденційні дані. Наприклад, атака з використанням методу факторизації може зламати приватний ключ, якщо параметри алгоритму DSA обрані неправильно або є недостатньо складними.

3. Вразливість до атаки з використанням підробки підпису

Ще однією вразливістю алгоритму DSA є можливість атаки з використанням підробки підпису. Зловмисники можуть спробувати знайти такі параметри алгоритму, як підпис та повідомлення, які будуть перевірені як правильні. Це може призвести до створення фальшивого підпису, що приймається як легітимний, що підриває довіру до систем, які використовують DSA.

4. Вразливість до атаки на випадковість

DSA використовує генератор випадкових чисел для створення криптографічних ключів. Якщо генератор випадкових чисел не є повністю випадковим або його вихід може бути передбачений, це може призвести до зламу системи. Наприклад, якщо зловмисник може вгадати випадкове число, використане для створення приватного ключа, він може використовувати цю інформацію для створення фальшивих підписів або злому системи.

2.7 Криптоаналіз алгоритму DSA

Криптоаналіз алгоритму цифрового підпису DSA (Digital Signature Algorithm) відіграє важливу роль у виявленні слабкостей та вразливостей цього алгоритму. Криптоаналітики та дослідники вдосконалюють методи і техніки атак для розкриття приватних ключів, порушення цілісності підпису та інших видів атак, що стосуються DSA. У цьому розділі ми розглянемо деякі з основних

методів криптоаналізу, які використовуються для виявлення вразливостей алгоритму DSA.

1. Факторизація

Одним з найбільш поширених методів криптоаналізу, що застосовується до алгоритму DSA, є факторизація. Заснований на математичній задачі факторизації, цей підхід спрямований на розкриття приватного ключа шляхом розкладання великого простого числа на множники. Якщо вдалося розкласти просте число, яке використовується у параметрах DSA, це може призвести до злому алгоритму та підробки підпису.

2. Атаки з використанням розширення підпису

Атаки, що використовують розширення підпису, націлені на злам алгоритму DSA, змінюючи підпис та повідомлення таким чином, щоб отримати додаткову інформацію про приватний ключ або забезпечити фальшиві підписи. Наприклад, атака з використанням розширення підпису може використовувати відомі підписи і повідомлення для побудови нового підпису, що є легітимним, але містить додаткову інформацію для атакуючого.

3. Атаки на побічні канали

Атаки на побічні канали використовують інформацію, яка витікає через побічні канали, такі як час виконання операцій, споживання енергії або електромагнітне випромінювання. Ці атаки можуть допомогти відновити приватний ключ, використовуваний у процесі підпису. Наприклад, зловмисник може аналізувати час виконання операцій DSA, щоб встановити значення приватного ключа.

4. Статистичний криптоаналіз

Статистичний криптоаналіз використовує методи статистики та ймовірності для аналізу даних, отриманих від алгоритму DSA. Ці методи можуть допомогти знайти патерни, слабкості або аномалії в процесі генерації підписів або використання приватного ключа. Наприклад, статистичний аналіз може виявити нерівномірності у розподілі випадкових чисел, що використовуються в

DSA, що може свідчити про вразливість.

2.8 Визначення актуальності DSA

Визначення актуальності шифру DSA (Digital Signature Algorithm) є важливим аспектом дослідження криптографічних алгоритмів. З огляду на швидкий технологічний розвиток і появу нових викликів у сфері кібербезпеки, необхідно періодично оцінювати стійкість та ефективність алгоритму DSA. У цьому розділі ми розглянемо деякі чинники, які впливають на актуальність шифру DSA та його потенційне місце в сучасній криптографії.

1. Комп'ютерна потужність

З появою більш потужних обчислювальних систем і зростанням обчислювальної потужності, виникає необхідність у шифрах, які витримують значно більші обчислювальні виклики. Шифр DSA, як і більшість криптографічних алгоритмів, має враховувати цей фактор. Важливо періодично переглядати параметри DSA та виконувати оновлення, щоб забезпечити відповідну стійкість до сучасних обчислювальних зусиль.

2. Криптографічні атаки

З появою нових атак та криптоаналітичних методів, необхідно періодично оцінювати стійкість DSA. Наприклад, факторизація, атаки з використанням розширення підпису та атаки на побічні канали - це лише кілька з численних атак, які можуть бути спрямовані на DSA. Якщо з'являються нові методи атаки, які можуть зламати DSA або порушити його безпеку, це може позначитися на актуальності алгоритму.

3. Стандарти та рекомендації

Актуальність шифру DSA також пов'язана з його використанням у стандартах і рекомендаціях. Наявність DSA у рекомендованих криптографічних стандартах, таких як стандарти NIST (National Institute of Standards and Technology), може вплинути на його актуальність та прийняття в індустрії. Якщо

існують нові стандарти або рекомендації, які замінюють DSA на більш безпечні алгоритми, це може вплинути на його актуальність.

4. Альтернативи

Актуальність DSA також залежить від наявності та ефективності альтернативних алгоритмів. Якщо на ринку з'являються нові алгоритми цифрового підпису, які мають більшу стійкість або кращу ефектиfвність, це може зменшити актуальність DSA. В такому випадку, розгляд альтернативних алгоритмів та їх порівняння можуть вплинути на прийняття рішення щодо використання DSA.

2.9 Вразливості алгоритму Ель-Гамеля

Алгоритм Ель-Гамелля є одним із популярних асиметричних криптографічних алгоритмів, використовуваних для шифрування та цифрового підпису. Однак, як і в усіх криптографічних алгоритмах, він має певні вразливості. У цьому розділі ми розглянемо деякі з основних вразливостей алгоритму Ель-Гамелля та їх можливі наслідки.

1. Вразливість до атаки на основі підслуховування

Алгоритм Ель-Гамелля піддається атакам на основі підслуховування, де злоумисники можуть перехопити шифровані повідомлення та використовувати їх для відновлення приватного ключа або самого повідомлення. Ця вразливість виникає внаслідок використання одного й того ж ключа для кількох повідомлень, що може призвести до викриття приватного ключа або зламу шифру.

2. Вразливість до атаки на основі повторного використання ключа

Якщо в алгоритмі Ель-Гамелля використовується один і той самий ключ для кількох повідомлень, то це може створити вразливість до атаки на основі повторного використання ключа. Злоумисники можуть складати рівняння на основі шифрованих повідомлень та використовувати їх для отримання додаткової інформації або зламу шифру.

3. Вразливість до атаки зі зміною повідомлення

Алгоритм Ель-Гамелля також може бути вразливим до атаки зі зміною повідомлення, де злоумисники можуть змінювати шифровані повідомлення без виявлення. Це може призвести до недостовірності та порушення цілісності даних.

4. Вразливість до атаки на основі обраних шифрованих повідомлень

Атака на основі обраних шифрованих повідомлень може бути використана проти алгоритму Ель-Гамелля, коли злоумисники мають можливість обирати шифровані повідомлення та отримувати відповідні розшифровані значення. Це може допомогти їм відновити приватний ключ або зламати шифр.

2.10 Криптоаналіз алгоритму Ель-Гамеля

Криптоаналіз алгоритму Ель-Гамеля є важливим аспектом дослідження криптографічних систем. Алгоритм Ель-Гамеля використовується для шифрування та цифрового підпису, і важливо мати розуміння його стійкості та потенційних вразливостей. У цьому розділі ми розглянемо деякі методи криптоаналізу алгоритму Ель-Гамеля та їх вплив на безпеку системи.

1. Атака на основі підслуховування

Атака на основі підслуховування є одним з найбільш поширених видів атак на криптографічні системи. В алгоритмі Ель-Гамеля, якщо злоумисник може перехопити шифровані повідомлення та відповідні підписи, він може спробувати використати їх для отримання секретних ключів або порушення цілісності повідомлень. Для запобігання таким атакам, важливо використовувати надійний канал комунікації та застосовувати додаткові заходи безпеки, такі як цифрові підписи та шифрування даних.

2. Атака на основі перебору

Атака на основі перебору спробує відновити приватний ключ, перебираючи можливі значення. В алгоритмі Ель-Гамеля, приватний ключ є

секретним числом, і його відновлення може призвести до злому системи. Щоб запобігти такій атаки, важливо використовувати достатньо довгі ключі та відповідні параметри, які забезпечують достатню стійкість до перебору.

3. Атака на основі знання частини приватного ключа

В алгоритмі Ель-Гамеля, якщо злоумисник знає деяку частину приватного ключа, він може використати цю інформацію для відновлення решти ключа. Це може статися, наприклад, якщо приватний ключ генерується з використанням недостатньо випадкових чи слабких параметрів. Для запобігання такій атаки, важливо використовувати сильні параметри та генерувати приватні ключі відповідно до рекомендацій безпеки.

4. Атака на основі колізій

Атака на основі колізій спробує знайти два різні повідомлення, які мають однаковий хеш або підпис. Якщо в алгоритмі Ель-Гамеля використовується хеш-функція зі слабкою стійкістю до колізій, це може привести до компрометації системи. Для запобігання такій атаки, важливо використовувати сильні хеш-функції та застосовувати додаткові методи безпеки, такі як використання солей або дерев хешування.

2.11 Актуальність алгоритму Ель-Гамеля

Алгоритм Ель-Гамеля є одним з важливих асиметричних криптографічних алгоритмів, який знаходить широке застосування в галузі шифрування та цифрових підписів. Однак, з огляду на швидкий розвиток криптографічних атак і нових викликів безпеці, актуальність алгоритму Ель-Гамеля потребує постійного оцінювання.

1. Обчислювальна складність

Алгоритм Ель-Гамеля відомий своєю високою обчислювальною складністю, особливо у порівнянні з іншими асиметричними криптографічними алгоритмами, такими як RSA. Це робить його привабливим для застосування в

різних системах, особливо там, де обчислювальні ресурси обмежені. Тим не менше, з появою потужних квантових комп'ютерів, які можуть вплинути на обчислювальну складність алгоритму Ель-Гамеля, виникає необхідність розгляду альтернативних алгоритмів, які будуть стійкими до квантових атак.

2. Стійкість до криптоаналізу

Стійкість алгоритму Ель-Гамеля до різних видів криптоаналізу є одним з головних критеріїв його актуальності. Незважаючи на те, що алгоритм Ель-Гамеля є відомим і добре вивченим, нові атаки та криптоаналітичні методи можуть вплинути на його безпеку. Тому необхідно постійно проводити оцінку стійкості алгоритму до нових видів атак та виробляти відповідні заходи для запобігання і зменшення ризиків.

3. Розмір ключів та продуктивність

Актуальність алгоритму Ель-Гамеля також залежить від його продуктивності та розміру ключів. Якщо алгоритм вимагає дуже великі ключі або має обчислювальні витрати, які несумісні з вимогами сучасних систем, то це може обмежити його застосування та зробити його менш актуальним. Важливо розглядати оптимізації алгоритму Ель-Гамеля, такі як використання еліптичних кривих, які дозволяють зменшити розмір ключів і поліпшити продуктивність.

4. Застосування алгоритму

Актуальність алгоритму Ель-Гамеля також залежить від його реальних застосувань та потреб користувачів. Якщо алгоритм Ель-Гамеля забезпечує потрібний рівень безпеки для конкретних сценаріїв, включаючи шифрування даних та цифровий підпис, то він залишається актуальним і потрібним.

2.12 Вразливості алгоритму RSA

RSA (Rivest-Shamir-Adleman) є одним з найбільш поширених асиметричних криптографічних алгоритмів, використовуваних для шифрування, цифрових підписів та обміну ключами. Протягом свого існування RSA був

підданий численним криптоаналітичним дослідженням, які призвели до виявлення деяких вразливостей. У цьому розділі ми розглянемо деякі з найвідоміших вразливостей алгоритму RSA і їх вплив на безпеку системи.

1. Факторизація модуля N

Одна з основних вразливостей алгоритму RSA полягає у складності проблеми факторизації великого цілого числа N, яке є частиною публічного ключа. Якщо злоумисник може успішно розкласти N на прості множники, то він зможе відновити приватний ключ і зламати систему. З іншого боку, якщо використовується достатньо велике число N з безпечної криптографічної довжини, наприклад, 2048 або 4096 біт, факторизація стає обчислювально непрактичною навіть для потужних комп'ютерів.

2. Повторне використання псевдовипадкових чисел

Ефективність алгоритму RSA в значній мірі залежить від якості використовуваних псевдовипадкових чисел. Якщо використовується той самий псевдовипадковий числовий генератор для кількох шифрувань або підписів, може статися повторне використання ключів. Це може призвести до компрометації системи, оскільки злоумисник може використати цю вразливість для відновлення приватного ключа. Для запобігання цьому необхідно використовувати криптографічно безпечний генератор псевдовипадкових чисел та забезпечувати унікальність ключів для кожного шифрування або підпису.

3. Side-Channel атаки

Side-Channel атаки є типом атак, які використовують додаткову інформацію, отриману шляхом спостереження фізичних властивостей пристрою, що виконує криптографічні операції. Це можуть бути атаки, засновані на витоку інформації про електричне споживання, електромагнітні випромінювання або час виконання операцій. RSA може бути вразливим до таких атак, якщо не застосовувати відповідні контрміри для зменшення витоку інформації. Наприклад, застосування захисних методів, таких як маскування, може допомогти ускладнити виявлення витоку інформації та збільшити стійкість до

таких атак.

4. Вплив квантових комп'ютерів

З огляду на швидкий розвиток квантових комп'ютерів, алгоритм RSA стає вразливим до квантових атак. Зокрема, Шоровий алгоритм, розроблений Пітером Шором, може ефективно розв'язувати проблему факторизації і дискретного логарифмування, які є основними складовими RSA. Таким чином, з'являється необхідність розгляду квантово-стійких алгоритмів шифрування та цифрових підписів для забезпечення безпеки в епоху квантових комп'ютерів.

2.13 Криптоаналіз алгоритму RSA

RSA (Rivest-Shamir-Adleman) є одним з найпоширеніших асиметричних криптографічних алгоритмів, який забезпечує безпеку шифрування, цифрового підпису та обміну ключами. Однак, протягом часу було розроблено різні криптоаналітичні методи, спрямовані на злам алгоритму RSA. У цьому розділі ми розглянемо деякі з найвідоміших методів криптоаналізу RSA і їх вплив на безпеку системи.

1. Факторизація числа N

Одним з найважливіших криптоаналітичних методів щодо RSA є факторизація числа N, яке є продуктом двох великих простих чисел p і q. Якщо злоумисник успішно розкладає N на прості множники, він може відновити приватний ключ і отримати доступ до зашифрованих даних або підробити цифровий підпис. Існує кілька алгоритмів факторизації, таких як метод факторизації Ферма, метод решета чисел і алгоритм квадратичного решета, які спрямовані на ефективне розкладання чисел на прості множники.

2. Атаки методом побічних каналів

Атаки методом побічних каналів використовують інформацію, отриману з фізичних властивостей пристрою, що виконує операції RSA, наприклад, електричне споживання, електромагнітні випромінювання або час виконання.

Наприклад, атака заснована на аналізі електричного споживання може дозволити злоумиснику відновити приватний ключ RSA. Щоб захиститись від таких атак, можуть використовуватися заходи, такі як використання маскування, випадковість та шумові примішання.

3. Атаки методом вибраного шифротексту

Атаки методом вибраного шифротексту (Chosen Ciphertext Attacks, CCA) використовуються для отримання інформації, яка не доступна в рамках звичайної криптосистеми RSA. В цих атаках злоумисник має можливість шифрувати довільні повідомлення і отримувати розшифровані версії цих повідомлень. Це може призвести до витоку інформації про приватний ключ або повідомлення, яке зашифровано. Для запобігання атакам CCA використовуються схеми підписування, такі як схеми з підписом під ключем або використання попереднього шифрування.

4. Атаки на основі дискретного логарифмування

RSA заснований на проблемі обчислення дискретного логарифмування. Тому атаки, спрямовані на ефективне розв'язування цієї проблеми, можуть впливати на безпеку RSA. Алгоритми, такі як алгоритм Шора, який використовує квантові комп'ютери, можуть ефективно розв'язувати дискретне логарифмування, що зроблює RSA вразливим до квантових атак.

2.14 Акутальність алгоритму RSA

Актуальність алгоритму RSA залишається значною в сучасній криптографії, хоча наявні також альтернативні алгоритми. Ось декілька причин, чому RSA продовжує бути актуальним:

Використання у веб-протоколах: RSA є одним з основних алгоритмів, який використовується для захищеного обміну ключами та шифрування веб-протоколів, таких як SSL/TLS. Багато серверів та веб-сайтів все ще використовують RSA для забезпечення конфіденційності та цілісності даних.

Ефективність для підписів: RSA є ефективним алгоритмом для створення цифрових підписів, які використовуються для підтвердження автентичності даних та ідентифікації відправника. Він дозволяє швидке створення та перевірку підписів, що робить його популярним для багатьох застосувань.

Застосування в криптографічних пристроях: RSA є широко застосовуваним алгоритмом в різних криптографічних пристроях, таких як смарт-карти, USB-токени та апаратні модулі безпеки. Його використання в цих пристроях дозволяє забезпечити безпечне зберігання приватних ключів та виконання криптографічних операцій.

Впровадження квантово-стійких алгоритмів: Хоча квантові комп'ютери загрожують безпеці RSA через ефективні алгоритми факторизації та дискретного логарифмування, активно проводяться дослідження для розробки квантово-стійких алгоритмів, які можуть замінити RSA у майбутньому. Проте, поки що RSA залишається використовуваним алгоритмом із великим рівнем безпеки до появи широкомасштабних квантових комп'ютерів.

Необхідно враховувати, що безпека RSA залежить від відповідного використання та налаштування параметрів, таких як довжина ключа та використання безпечного генератора псевдовипадкових чисел. При забезпеченні належного застосування, RSA може продовжувати залишатися актуальним і надійним алгоритмом для багатьох криптографічних застосувань.

2.15 Вразливості алгоритму ECDSA

ECDSA (Elliptic Curve Digital Signature Algorithm) є асиметричним криптографічним алгоритмом підпису, який базується на еліптичних кривих. Хоча ECDSA вважається безпечним алгоритмом, існують певні вразливості та атаки, які можуть впливати на його безпеку. У цьому розділі ми розглянемо деякі з найвідоміших вразливостей алгоритму ECDSA та їх можливі наслідки.

Вразливості на основі погано підібраних параметрів Вибір відповідних

параметрів еліптичної кривої є важливою частиною ECDSA. Погано підібрані параметри, такі як недостатня довжина ключа, слабкі еліптичні криві або використання вразливих параметрів, можуть призвести до зниження безпеки алгоритму. Злоумисники можуть використовувати ці вразливості для підробки підписів або відновлення приватного ключа.

Атаки на основі криптографічних обчислень. Існують різні атаки, які використовують математичні властивості еліптичних кривих для скорочення часу розрахунків або підвищення ефективності побудови підпису. Деякі з цих атак включають атаку Полларда на дискретне логарифмування (Pollard's rho algorithm) та атаку Полларда на пристрілок (Pollard's kangaroo algorithm). Ці атаки можуть зменшити безпеку алгоритму та зробити його більш піддаємим атакам.

Атаки методом фальсифікації підпису. Атаки на основі фальсифікації підпису спрямовані на створення недійсних підписів, які будуть визнані як справжні. Ці атаки можуть виникнути, якщо використовується неправильний режим підписування або недостатній контроль цілісності підписування. Як результат, злоумисник може здійснити фальсифікацію підписів та підробку даних.

Атаки на основі побічних каналів. Атаки на основі побічних каналів використовують інформацію, яка витікає під час виконання криптографічних операцій, таких як затримки часу, споживання енергії або електромагнітні випромінювання. Ці атаки можуть допомогти зловмиснику витягти додаткову інформацію про приватний ключ або підписи, що може призвести до компрометації безпеки алгоритму.

2.16 Криптоаналіз алгоритму ECDSA

ECDSA (Elliptic Curve Digital Signature Algorithm) є асиметричним криптографічним алгоритмом підпису, який базується на еліптичних кривих. Хоча ECDSA вважається сильним криптографічним алгоритмом,

використовувати його належним чином потребує розуміння можливих атак та вразливостей. У цьому розділі ми розглянемо деякі основні методи криптоаналізу алгоритму ECDSA та їх можливі наслідки.

1. Атаки на основі підрахунку приватного ключа

Одна з основних цілей криптоаналізу ECDSA полягає в підрахунку приватного ключа на основі доступної інформації. Існують різні атаки, такі як атака на основі підрахунку дискретного логарифму, атака на основі зламу еліптичних кривих та атака на основі підрахунку порядку точок. Ці атаки можуть використовувати математичні властивості еліптичних кривих та витрачати значний обчислювальний ресурс для відновлення приватного ключа.

2. Атаки на основі фальсифікації підпису

Атаки на фальсифікацію підпису спрямовані на створення недійсних підписів, які будуть визнані як справжні. Ці атаки можуть використовувати вразливості в режимах підпису або використовувати слабкість в генерації випадкових чисел. Атаки, такі як атака на основі відомого підпису, можуть дозволити зловмиснику відтворити підписи без доступу до приватного ключа.

3. Атаки на основі сторонніх каналів

Атаки на сторонні канали використовують інформацію, яка витікає під час виконання криптографічних операцій, таких як затримки часу або споживання енергії. Ці атаки можуть допомогти зловмиснику витягнути додаткову інформацію про приватний ключ або підписи, що може призвести до компрометації безпеки алгоритму.

4. Атаки на основі кількості повторних підписів

Деякі атаки на ECDSA використовують специфічні властивості алгоритму, що дозволяють скоротити час або ресурси, необхідні для підпису. Наприклад, атака на основі підпису на основі відомої частини підпису може знизити кількість повторних підписів, необхідних для відновлення приватного ключа.

2.17 Актальність алгоритму ECDSA

ECDSA (Elliptic Curve Digital Signature Algorithm) є одним з найбільш поширених алгоритмів цифрового підпису на основі еліптичних кривих. Цей алгоритм має багато переваг, таких як короткі ключі, висока швидкодія та ефективність, що робить його популярним для багатьох криптографічних застосувань. У цьому розділі ми розглянемо актуальність алгоритму ECDSA в сучасному криптографічному середовищі.

1. Стійкість до квантових обчислювальних атак

Однією з ключових переваг ECDSA є його стійкість до квантових обчислювальних атак. Зараз існує загроза квантових комп'ютерів, які можуть зламати багато сучасних криптографічних алгоритмів, зокрема алгоритм RSA та Диффі-Геллмана. Однак, алгоритм ECDSA використовує еліптичні криві, які є відмінними від інших алгоритмів і мають властивості, які забезпечують стійкість до квантових обчислювальних атак. Це робить ECDSA актуальним і надійним алгоритмом для майбутнього, коли квантові комп'ютери стануть доступними.

2. Ефективність та швидкодія

ECDSA відомий своєю ефективністю та швидкодією порівняно з іншими алгоритмами цифрового підпису. Еліптичні криві вимагають коротших ключів порівняно з традиційними алгоритмами, такими як RSA, що полегшує обробку даних і зменшує обчислювальні витрати. Швидкодія ECDSA робить його ідеальним вибором для протоколів мережевої безпеки, криптовалют, електронних голосувань та багатьох інших застосувань, де швидка обробка підписів є важливою.

3. Підтримка стандартами

ECDSA є широко використовуваним алгоритмом і підтримується багатьма стандартними організаціями та протоколами. Наприклад, алгоритм ECDSA включений до стандартів Інституту інженерів з електротехніки та електроніки (IEEE), Національного інституту стандартів і технологій (NIST) та Консорціуму

відкритого коду (OSS). Ця широка підтримка забезпечує сумісність та інтероперабельність алгоритму ECDSA з різними системами та протоколами, що підвищує його актуальність і значущість.

Висновки до другого розділу

У другому розділі було розглянуто питання про кваліфікаційний електронний підпис, його різновиди, механізм роботи та використання. Були виділені три основні різновиди:

- Простий кваліфікаційний електронний підпис.
- Вдосконалений некваліфікований кваліфікаційний електронний підпис.
- Вдосконалений кваліфікований кваліфікаційний електронний підпис.

На сьогоднішній день метод захисту даних з використанням кваліфікаційного електронного підпису є актуальним та ефективним засобом захисту інформації.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ОБРАНОГО АЛГОРИТМУ ШИФРУВАННЯ

У сфері інформаційних технологій однією з проблем щодо правових питань є захист персональних даних. В умовах глобалізації, інформація про користувачів може бути використана третіми особами з усього світу, часто незаконно. Тому захист інформації стає невід'ємною частиною сучасного суспільства. На основі висновків, зроблених у першому розділі дипломної роботи, можна стверджувати, що одним з найефективніших методів захисту даних є використання кваліфікаційного електронного підпису (КЕП).

Після проведеного аналізу різних алгоритмів було встановлено, що алгоритм DSA є найбільш підходящим для захисту даних. У подальшому в роботі буде здійснено програмну реалізацію саме цього алгоритму.

3.1 Опис засобів для створення програмного модуля

Для розробки програмного забезпечення, що реалізовує обраний алгоритм підпису, рекомендується використовувати інтегроване середовище розробки (IDE) для мов програмування C/C++, таке як Visual Studio Code, яке включає в себе потрібні інструменти та можливості для ефективної роботи з вихідними кодами.

3.2 Інтегроване середовище розробки DEV C++

Dev-C++ є повноцінним інтегрованим середовищем розробки (IDE) для мов програмування C і C++, спеціально розробленим для платформи Windows. З моменту свого випуску у 1998 році, воно отримало широку популярність серед

розробників, студентів і дослідників. Dev-C++ було часто згадано в літературі з програмування на C++ та наукових публікаціях, і його залишають улюбленим інструментом навчання в університетах та школах по всьому світу.

Основні особливості Dev-C++ включають:

– Dev-C++ - це легке та переносне середовище розробки (IDE) для мов програмування C і C++ на платформі Windows.

– Воно підтримує різні компілятори, зокрема засновані на GCC, такі як Mingw і Cygwin. Це дозволяє швидко створювати графічні і консольні програми для Windows, а також статичні бібліотеки та DLL-бібліотеки.

– Dev-C++ має вбудований засіб для налагодження програм.

– Інші функції включають браузер класів, автодоповнення коду, перегляд функцій, підтримку профілювання та підтримку понад 30 мов програмування.

– В IDE є настроюваний редактор коду, керівник проектів, шаблони для створення власних типів проектів, генерація файлу Makefile, редагування та компіляція файлів ресурсів, менеджер інструментів та функції пошуку та заміни об'єктів.

– Крім того, Dev-C++ підтримує CVS (Concurrent Versions System) для управління версіями.

3.3 Visual Studio Code

Visual Studio Code - це один з найкращих редакторів вихідних кодів, який, незважаючи на свою легкість, володіє широким спектром можливостей. Він підтримує операційні системи Windows, macOS та Linux і надає зручну роботу з JavaScript, TypeScript та Node.js, а також з багатьма іншими мовами програмування, такими як C++, C#, Java, Python, PHP та Go, завдяки великій кількості розширень.

Visual Studio Code пропонує наступні переваги:

- Інтегровані функції, які полегшують роботу, наприклад автоматичне підсвічування повторюваних змінних.

- Простота використання та швидка модифікація сценарію.

Основою Visual Studio Code є відмінний редактор вихідного коду, який ідеально підходить для повсякденного використання. Завдяки підтримці багатьох мов програмування, VS Code покращує продуктивність завдяки функціям, таким як підсвічування синтаксису, відступи, виділення фрагментів та інші. Зрозумілі комбінації клавіш, легке налаштування та показ комбінацій клавіш, запропонованих спільнотою, допомагають зосередитися на коді.

Для серйозного програмування важливо мати інструменти, які розуміють код краще, ніж прості текстові блоки. Visual Studio Code надає вбудовану підтримку для автозаповнення коду IntelliSense, розширеного розуміння семантики коду, навігації та рефакторингу.

Крім того, VS Code інтегрується з інструментами побудови та сценарію, що дозволяє прискорити робочі процеси. Він також має вбудовану підтримку Git, що дозволяє працювати з керуванням версіями безпосередньо в редакторі, включаючи перегляд змін.

3.4 Мова програмування C++

C++ є переносною мовою програмування, яка використовується для створення високопродуктивних додатків. Вона була розроблена Б'ярне Страуструпом як розширення мови C. C++ надає програмістам високий рівень контролю над системними ресурсами та пам'яттю. Мова була оновлена тричі, у 2011, 2014 та 2017 роках, до версій C++11, C++14 та C++17 [15].

Нижче наведені бібліотеки, які можуть бути використані в програмі:

<iostream> - це бібліотека введення/виведення, яка містить стандартні об'єкти потоків.

<cstdlib> - ця бібліотека діє так, ніби кожне ім'я з <cstdlib> знаходиться у

глобальному просторі імен.

<sstream> - це бібліотека введення/виведення для класів std::basic_stringstream, std::basic_istringstream та std::basic_ostringstream.

<fstream> - ця бібліотека введення/виведення для класів std::basic_fstream, std::basic_ifstream та std::basic_ofstream.

<string> - це бібліотека для роботи з рядками, яка містить шаблонний клас std::basic_string.

<math.h> - ця бібліотека діє так, ніби кожне ім'я з <cmath> знаходиться у глобальному просторі імен, за винятком назв математичних спеціальних функцій.

<ctime> - це бібліотека, яка містить утиліти для роботи з часом та датою.

<locale> - ця бібліотека використовується для локалізації

```
1   #include <iostream>
2   #include <stdlib.h>
3   #include <sstream>
4   #include <fstream>
5   #include <conio.h>
6   #include <string>
7   #include <math.h>
8   #include <ctime>
9   #include <locale>
10  #include <Windows.h>
```

Рисунок 3.1 Використані бібліотеки для реалізування DSA алгоритму

3.5 Мова програмування JavaScript

JavaScript (звичайно позначається як JS) - це легка, інтерпретована об'єктно-орієнтована мова з першокласними можливостями. Вона відома переважно як мова сценаріїв для веб-сторінок, але також використовується в різних небраузерних середовищах. JavaScript є прототипною мовою сценаріїв, яка підтримує різні парадигми програмування, включаючи динамічну, об'єктно-орієнтовану, функціональну та необхідну.

JavaScript використовується на клієнтському веб-сайті, де вона

використовується для програмування поведінки веб-сторінки відповідно до подій. JavaScript - це мова сценаріїв, яка є легкою для вивчення, але потужною в своїх можливостях. Вона часто використовується для керування поведінкою веб-сайтів. JavaScript може виступати як процедурна та об'єктно-орієнтована мова. У JavaScript об'єкти створюються програмно, а методи та властивості можуть бути додані до порожніх об'єктів під час виконання програми. Це відрізняється від мов, які використовують визначення класів, таких як C++ та Java. Після побудови об'єкт може бути використаний як шаблон (або прототип) для створення подібних об'єктів. Динамічні можливості JavaScript включають створення виконуваних об'єктів, змінні списки, функціональні змінні, створення динамічних сценаріїв (за допомогою eval), інтроспекцію об'єктів (за допомогою циклу for ... in) та відновлення початкового коду (програми JavaScript можуть розібрати тіла функцій у вихідний текст).

3.6 Програмна реалізація алгоритму шифрування

Відповідно до алгоритму були задані початкові значення p, q, x, k, H, де

p – просте число, $2^{L-1} < p < 2^L$, де $512 < L < 1024$ і L кратне 64

q – простий дільник $p - 1$, довжиною 160 бітів;

x – випадкове або псевдовипадкове число, $0 < x < q$.

k – випадкове або псевдовипадкове число, $0 < k < q$.

```
#include <iostream>
#include <stdlib.h>
#include <sstream>
#include <fstream>
#include <conio.h>
#include <string>
#include <math.h>
#include <ctime>
#include <locale>
#include <Windows.h>
using namespace std;
int p = 569, q = 71, x = 6, k = 2, H = 668;
```

Рисунок 3.2 Оголошення змінних

Оголошення функцій перевірки h на відповідність: h – ціле, $1 < h < p - 1$ і $h^{(p-1)/q} \bmod p > 1$, та для вводу h користувачем

```
bool check_h(int h)
{
    return h > 1 && h < p - 1 && fmod(pow(h, (p - 1) / q), p) > 1;
}
int get_h()
{
    int h;
    do
    {
        cout << endl
             << "Enter number h: ";
        cin >> h;
    } while (!check_h(h));
    return h;
}
```

Рисунок 3.3 Створення функції

Основна функція програми полягає у виконанні обчислень для отримання глобальної компоненти відкритого ключа (g) та відкритого ключа користувача (y) відповідно до встановленого алгоритму. Далі, з використанням змінних, оголошених на початку програми, а також обчислених значень g та y, генерується підпис. Змінні r та s обчислюються відповідно до формул, що використовуються в алгоритмі DSA, і їх значення виводяться на екран.

```
int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    setlocale(LC_CTYPE, "eng");
    clock_t start = clock();
    int h = get_h();
    // while (true) {
    cout << "h = " << h << endl;

    int g = fmod(pow(h, (p - 1) / q), p);
    cout << "g = " << g << endl;
    cout << "x = " << x << endl;

    int y = fmod(pow(g, x), p);
    cout << "y = " << y << endl;
    cout << "k = " << k << endl;

    cout << endl
         << "Create signature: " << endl;

    int r = fmod(y, q);
    cout << "r = " << r << endl;

    cout << "Hash H(M) = " << H << endl;
    int s = fmod(pow(k, -1) * (H + r * x), q);
    cout << "s = " << s << endl;

    cout << "Signature: (" << r << ", " << s << ")" << endl;
```

Рисунок 3.4 Реалізація створеного підпису

Здійснюється перевірка (верифікація) підпису шляхом обчислення змінних $w, u1, u2$ та v згідно з формулами, що використовуються в алгоритмі DSA. Перевіряється умова $v = r$, і на основі цього виробляється висновок щодо підтвердження або непідтвердження підпису. Також вимірюється час, необхідний для виконання програми.

```
    cout << endl
         << "Verefication of signature: " << endl;

    int w = fmod(s, q);
    int u1 = fmod(H * w, q);
    int u2 = fmod(r * w, q);

    int v = fmod(fmod(pow(g, u1) * pow(y, u2), p), q);
    cout << "v = " << v << ", r = " << r << endl
         << endl;
    string msg = v == r ? "Verificated!" : "Not verificated!";
    cout << msg;
    clock_t end = clock();
    double seconds = (double)(end - start) / CLOCKS_PER_SEC;
    cout << endl
         << endl
         << "Time for create or verific: "
         << seconds << " s." << endl
         << endl;
    return 0;
}
```

Рисунок 3.5 Перевірка підпису

Робота програмно реалізованого алгоритму DSA зі створення та верифікації КЕП. У даному випадку $v \neq r$ (Рис.12), отже підпис анульовано.

```
Enter number h: 23
h = 23
g = 69
x = 6
y = 293
k = 2

Create signature:
r = 9
Hash H(M) = 668
s = 6
Signature: (9, 6)

Verefication of signature:
v = 12, r = 9

Not verificated!

Time for create or verific: 7.185 s.
```

Рисунок 3.6 Робота програми, анульований підпис

Аналогічна процедура, але у даному випадку $v = r$ (Рис 13), отже підпис веріфіковано

```
Enter number h: 20
h = 20
g = 372
x = 6
y = 447
k = 2

Create signature:
r = 21
Hash H(M) = 668
s = 42
Signature: (21, 42)

Verefication of signature:
v = 21, r = 21

Verificated!

Time for create or verific: 3.484 s.
```

Рисунок 3.8 Робота програми, верифікований підпис

3.7 Програмний модуль захисту інформації за допомогою шифрування з відкритим ключем

За допомогою редактора вихідних кодів VS Code програмно реалізовано генерацію ключів за допомогою алгоритму DSA. У коді використовується тип "spki" для створення відкритого ключа. Цей тип використовується як спеціальна інфраструктура відкритих ключів, призначена для усунення недоліків стандарту X.509.

Для створення особистого ключа використовується тип "pkcs8", який є стандартом синтаксису для зберігання приватних ключів у сучасній криптографії

```
const crypto = require('crypto');
const fs = require('fs');

const KEYS_FILE_PATH = 'keys/';
const KEYS_FILE_EXTENSION = '.pem';

module.exports.generateKeys = () => {
    crypto.generateKeyPair(
        'rsa',
        {
            modulusLength: 4096,
            publicKeyEncoding: {
                type: 'spki',
                format: 'pem',
            },
            privateKeyEncoding: {
                type: 'pkcs8',
                format: 'pem',
            },
        },
        (err, publicKey, privateKey) => {
            fs.writeFileSync(`${KEYS_FILE_PATH}publicKey${KEYS_FILE_EXTENSION}`, publicKey, {
                encoding: 'utf8',
                flag: 'w',
            });
            fs.writeFileSync(`${KEYS_FILE_PATH}privateKey${KEYS_FILE_EXTENSION}`, privateKey, {
                encoding: 'utf8',
                flag: 'w',
            });
        }
    );
};
```

Рисунок 3.9 Реалізація генераціх ключів

Далі створюємо функцію підпису документу

```
const crypto = require('crypto');
const fs = require('fs');

module.exports.sign = (filename, callback) => {
  const privateKey = fs.readFileSync('keys/privateKey.pem', 'utf-8');
  const doc = fs.readFileSync('uploads/' + filename);
  const signer = crypto.createSign('RSA-SHA256');
  signer.write(doc);
  signer.end();
  const signature = signer.sign(privateKey, 'hex');
  fs.writeFileSync('uploads/signature.hex', signature);
  callback(signature);
};
```

Рисунок 3.10 Програмний код підпису документу за допогою ключів

Останнім кроком слугує реалізація функції перевірки підписаного документу

```
const crypto = require('crypto');
const fs = require('fs');

module.exports.verify = (filename, isKeyUploaded, isSignatureUploaded, callback) => {
  const publicKeyFolder = isKeyUploaded ? 'uploads/' : 'keys/';
  const signatureFolder = isSignatureUploaded ? 'uploads/' : '';

  let public_key = fs.readFileSync(`${publicKeyFolder}publicKey.pem`, 'utf-8');
  const signature = fs.readFileSync(`${signatureFolder}signature.hex`, 'utf-8');
  const doc = fs.readFileSync(`uploads/${filename}`);

  const verifier = crypto.createVerify('RSA-SHA256');
  verifier.write(doc);
  verifier.end();

  const result = verifier.verify(public_key, signature, 'hex');
  console.log('Digital Signature Verification : ' + result);

  callback(result);
};
```

Рисунок 3.11 Програмний код перевірки підписаного документу

Для оцінки ефективності розробленого методу захисту даних необхідно виконати тестування додатка згідно наступного алгоритму:

– Завантажити документ для підпису.
– Згенерувати ключову пару.
– Здійснити підпис завантаженого документа.
– Провести верифікацію підпису.
– Внести зміни в документ (атака).
– Повторно провести верифікацію.

Для початку перевірки можна відкрити початкове меню розробленого додатка

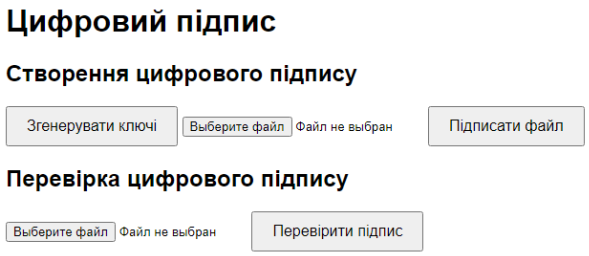


Рисунок 3.12 Почтакове меню додатку

Продовжуємо здійснювати перший крок алгоритму, який передбачає завантаження документу для захисту шляхом накладання кваліфікаційного електронного підпису (КЕП). За допомогою опції "Файл" у меню програми здійснюється імпорт текстового файлу.

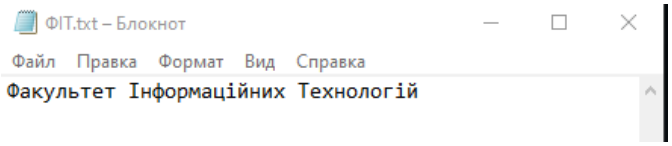


Рисунок 3.13 Файл для накладання підпису

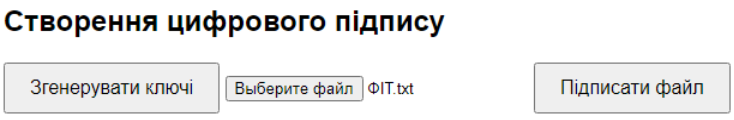


Рисунок 3.14 Завантаження файлу

Після завантаження файлу, потрібно згенерувати особистий та публічний ключі для їх подальшого використання, натискаємо на кнопку «Згенерувати ключі»

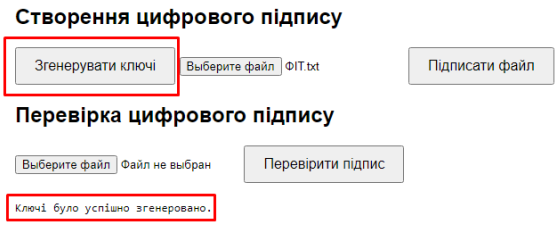


Рисунок 3.15 Успішна генерація ключів

Після генерації ключів – вони зберігаються в окремій папці, для зручного перегляду та їх передачі

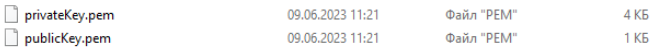


Рисунок 3.16 Згенерована пара ключів

Рисунок 3.17 Публічний ключ

Рисунок 3.18 Частина приватного ключа

Після успішної генерації ключів – виконується підпис документу – та

генерація самого підпису в форматі hex

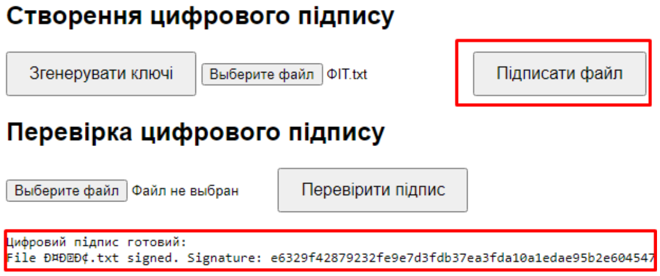


Рисунок 3.19 Момент підписання документу

Рисунок 3.20 Файл підпису

Проводимо верифікацію підписаного файлу (без змін) та самого підпису, задля можливості атаки в наступному кроці

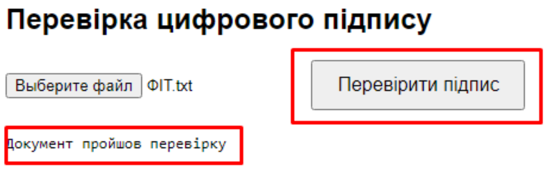


Рисунок 3.21 Перевірка документу

Редагуємо підписаний документ для перевірки надійності створеного додатку.

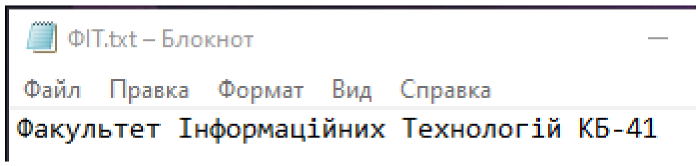


Рисунок 3.22 Відредагований документ

Робимо повторну верифікацію документу (так звану атаку), задля перевірки додатку.

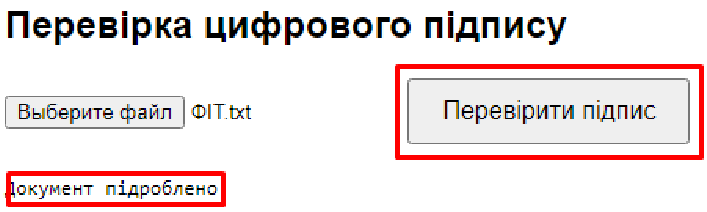


Рисунок 3.23 Атака та її результат

Висновки до третього розділу

У третьому розділі дипломної роботи було реалізовано обраний у першому розділі алгоритм підпису за допомогою програмного модуля. Виконано генерацію ключів, підписання документа обраним алгоритмом, а також проведено верифікацію кваліфікаційного електронного підпису до і після внесення змін в документ. Для реалізації DSA алгоритму було використане інтегроване середовище програмування для мов розробки C/C++ - Dev C++. Основною перевагою цього середовища є його легкість використання та приємний для користувача інтерфейс.

Для створення кваліфікаційного електронного підпису з використанням реалізованого алгоритму, інтерфейсу та функціональності веб-додатку, було використано редактор вихідного коду - VS Code, який підтримує багато мов програмування, зокрема JavaScript. Код реалізації алгоритму був написаний на мові C++, яка відома своєю продуктивністю та здатністю до динамічної обробки інформації. Ця мова часто використовується для проектів, де продуктивність коду має велике значення.

Для правильної роботи кваліфікаційного електронного підпису, інтерфейсу та функціональності веб-додатку було використано мову JavaScript. JavaScript має багато переваг, таких як швидкість виконання, простота використання, широкий набір функціональності та універсальність.

Завдяки розробці програмного модуля для обраного алгоритму підпису у другому розділі дипломної роботи було продемонстровано, що кваліфікаційний електронний підпис є надійним методом захисту інформації. Він дозволяє запобігти змінам в офіційних документах, зберігаючи їх цілісність.

ВИСНОВКИ

Інформаційна сфера постійно зазнає змін, що призводить до появи нових загроз. Цей розвиток небезпеки вимагає створення нових методів захисту даних. У дипломній роботі було розглянуто одне з найбільш актуальних питань сучасної інформаційної безпеки - забезпечення цілісності даних. Був розроблений метод захисту даних, який ґрунтується на використанні кваліфікаційного електронного підпису. Під час виконання роботи було здійснено наступні кроки:

1) Проведено аналіз існуючих методів захисту даних, що підтвердив ефективність криптографічних методів для забезпечення інформаційної безпеки.

2) Проведено аналіз основних алгоритмів для реалізації кваліфікаційного електронного підпису, таких як RSA, Ель-Гамаль, DSA та ECDSA, для вибору найбільш підходящого алгоритму.

3) Розроблено програмну реалізацію криптографічного алгоритму DSA з використанням інтегрованого середовища розробки для мов програмування C/C++ - Dev C++ та редактора вихідного коду Visual Studio Code. Це дозволило забезпечити цілісність інформаційних ресурсів.

На основі результатів перевірки додатку можна зробити висновок, що розроблений метод захисту даних є надійним, оскільки при внесенні змін у підписаний документ програма повідомляє про спробу фальсифікації підпису.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Про захист персональних даних" [Текст]: Закон України № 2297-VI від 01 червня 2010 р. / Верховна Рада України // Відомості Верховної Ради України. – 2010. – №34. – Ст. 481.

2. "Методи захисту інформації" [Електронний ресурс] – Режим доступу до ресурсу: https://bezpeka.com.ua/informacionnaya_bezopasnost/zaschita_informacii/methody_zaschity_informacii/

3. Ісмайлов К.Ю. "Криптографічні методи захисту інформації: види та вимоги до них" // Одеський державний університет внутрішніх справ. – 2018. – С.181

4. "Назначение криптографических методов защиты информации" [Електронний ресурс] – Режим доступу до ресурсу: http://aguryanov.blogspot.com/2012/09/blog-post_3412.html

5. "Електронний цифровий підпис" [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Електронний_цифровий_підпис

6. "Про електронний цифровий підпис" [Текст]: Закон України № 852-IV від 05 жовтня 2003 р. / Верховна Рада України // Відомості Верховної Ради України. – 2003. – №36. – Ст. 276

7. "КЕП" [Електронний ресурс] – Режим доступу до ресурсу: https://www.audit-it.ru/terms/agreements/elektronno_tsifrovaya_podpis_etsp.html

8. "Про затвердження форм заяв у сфері державної реєстрації юридичних осіб, фізичних осіб - підприємців та громадських формувань" [Текст]: Наказ № 3268/5 від 18 листопада 2016 р.

9. RSA [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/RSA

10. Схема Ель-Гамаля [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Схема_Ель-Гамаля

11. DSA [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/DSA

12. Elliptic Curve Digital Signature Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm

13. Home – Dev C++ [Електронний ресурс] – Режим доступу до ресурсу: https://www.bloodshed.net/

14. Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs/editor/whyvscode

15. Веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/cpp/cpp_intro.asp

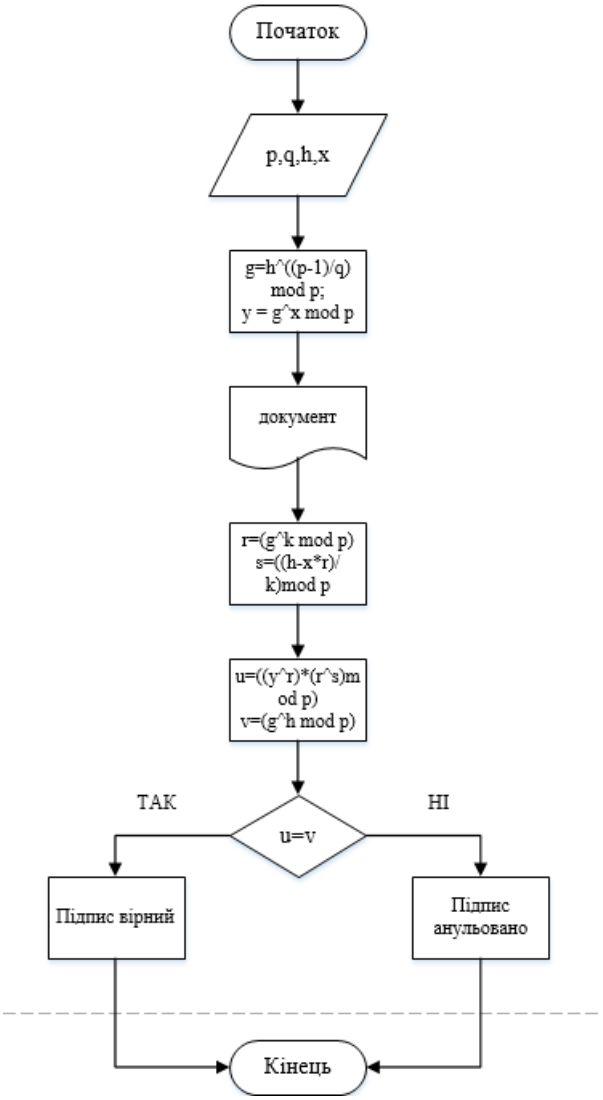
16. Веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://en.cppreference.com/w/cpp/header

17. Веб-сайт [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

18. SPKI [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/SPKI

19. PKCS8 [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/PKCS_8

ДОДАТОК А
СХЕМА DSA - АЛГОРИТМУ



ДОДАТОК Б
ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ

```
#include <iostream>
#include <stdlib.h>
#include <sstream>
#include <fstream>
#include <conio.h>
#include <string>
#include <math.h>
#include <ctime>
#include <locale>
#include <Windows.h>
using namespace std;
int p = 569, q = 71, x = 6, k = 2, H = 668;
bool check_h(int h)
{    return h > 1 && h < p - 1 && fmod(pow(h, (p - 1) / q), p) > 1;
}
int get_h()
{
   int h;
   do
   {
      cout << endl
           << "Enter number h: ";
      cin >> h;
   } while (!check_h(h));
   return h;
}
int main()
{
   SetConsoleCP(1251);
   SetConsoleOutputCP(1251);
   setlocale(LC_CTYPE, "eng");
   clock_t start = clock();
```

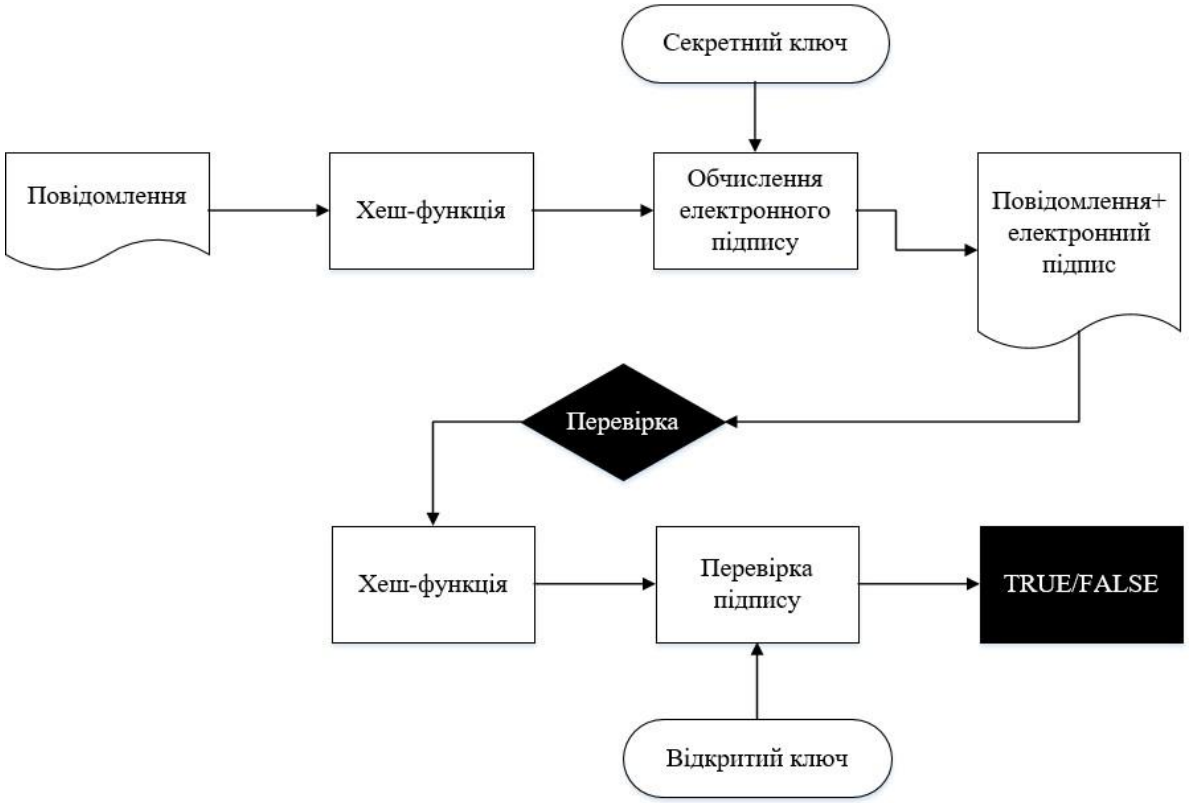
ПРОДОВЖЕННЯ ДОДАТКУ Б

```
int h = get_h();
    // while (true) {
    cout << "h = " << h << endl;

    int g = fmod(pow(h, (p - 1) / q), p);
    cout << "g = " << g << endl;
    cout << "x = " << x << endl;

    int y = fmod(pow(g, x), p);
    cout << "y = " << y << endl;
    cout << "k = " << k << endl;

    cout << endl
         << "Create signature: " << endl;

    int r = fmod(y, q);
    cout << "r = " << r << endl;

    cout << "Hash H(M) = " << H << endl;
    int s = fmod(pow(k, -1) * (H + r * x), q);
    cout << "s = " << s << endl;

    cout << "Signature: (" << r << ", " << s << ")" << endl;

    cout << endl
         << "Verefication of signature: " << endl;

    int w = fmod(s, q);
    int u1 = fmod(H * w, q);
    int u2 = fmod(r * w, q);

    int v = fmod(fmod(pow(g, u1) * pow(y, u2), p), q);
    cout << "v = " << v << ", r = " << r << endl
         << endl;
    string msg = v == r ? "Verificated!" : "Not verificated!";
    cout << msg;
    clock_t end = clock();
    double seconds = (double)(end - start) / CLOCKS_PER_SEC;
    cout << endl
         << endl
         << "Time for create or verific: "
         << seconds << " s." << endl
         << endl;
    return 0;
}

                int w = fmod(s, q);
```

ПРОДОВЖЕННЯ ДОДАТКУ Б

```
int u1 = fmod(H * w, q);
    int u2 = fmod(r * w, q);

    int v = fmod(fmod(pow(g, u1) * pow(y, u2), p), q);
    cout << "v = " << v << ", r = " << r << endl
        << endl;
    string msg = v == r ? "Verificated!" : "Not verificated!";
    cout << msg;
    clock_t end = clock();
    double seconds = (double)(end - start) / CLOCKS_PER_SEC;
    cout << endl
        << endl
        << "Time for create or verific: "
        << seconds << " s." << endl
        << endl;
    return 0;
}
```

ДОДАТОК В
БЛОК СХЕМА АЛГОРИТМУ РОБОТИ ПРОГРАМНОГО МОДУЛЮ КЕП



ДОДАТОК Г
КОД МОДУЛЮ ГЕНЕРАЦІЇ КЛЮЧІВ

```
const crypto = require('crypto');
const fs = require('fs');

const KEYS_FILE_PATH = 'keys/';
const KEYS_FILE_EXTENSION = '.pem';

module.exports.generateKeys = () => {
  crypto.generateKeyPair(
    'rsa',
    {
      modulusLength: 4096,
      publicKeyEncoding: {
        type: 'spki',
        format: 'pem',
      },
      privateKeyEncoding: {
        type: 'pkcs8',
        format: 'pem',
      },
    },
    (err, publicKey, privateKey) => {
      fs.writeFileSync(`${KEYS_FILE_PATH}publicKey${KEYS_FILE_EXTENSION}`,
publicKey, {
        encoding: 'utf8',
        flag: 'w',
      });
      fs.writeFileSync(`${KEYS_FILE_PATH}privateKey${KEYS_FILE_EXTENSION}`,
privateKey, {
        encoding: 'utf8',
        flag: 'w',
      });
    }
  );
}
```

ДОДАТОК Д
КОД СТВОРЕННЯ ЕЛЕКТРОННОГО ЦИФРОВОГО ПІДПИСУ

```
const crypto = require('crypto');
const fs = require('fs');

module.exports.sign = (filename, callback) => {
  const privateKey = fs.readFileSync('keys/privateKey.pem', 'utf-8');
  const doc = fs.readFileSync('uploads/' + filename);
  const signer = crypto.createSign('RSA-SHA256');
  signer.write(doc);
  signer.end();
  const signature = signer.sign(privateKey, 'hex');
  fs.writeFileSync('uploads/signature.hex', signature);
  callback(signature);
```

ДОДАТОК Е
КОД СТВОРЕННЯ ПРОЦЕДУРИ ВЕРИФІКАЦІЇ ПІДПИСУ

```
const crypto = require('crypto');
const fs = require('fs');

module.exports.verify = (filename, isKeyUploaded, isSignatureUploaded, callback) => {
  const publicKeyFolder = isKeyUploaded ? 'uploads/' : 'keys/';
  const signatureFolder = isSignatureUploaded ? 'uploads/' : '';

  let public_key = fs.readFileSync(`${publicKeyFolder}publicKey.pem`, 'utf-8');
  const signature = fs.readFileSync(`${signatureFolder}signature.hex`, 'utf-8');
  const doc = fs.readFileSync(`uploads/${filename}`);

  const verifier = crypto.createVerify('RSA-SHA256');
  verifier.write(doc);
  verifier.end();

  const result = verifier.verify(public_key, signature, 'hex');
  console.log('Digital Signature Verification : ' + result);

  callback(result);
};
```