Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)
Кафедра Інфокомунікаційної інженерії імені В.В. Поповського
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Аналіз методів комбінованого шифрування бази даних із даними в стані спокою
(тема)

Виконав:
студент 2 курсу, групи АМСЗІм-21-2
Полєжаєв А. В.
(прізвище, ініціали)

Спеціальність: 125 Кібербезпека
(код і повна назва спеціальності)
Тип програми: освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма: Адміністративний менеджмент у сфері захисту інформації
(повна назва освітньої програми)

Керівник: професор кафедри ІКІ ім. В.В. Поповського
Радівілова Т. А.
(посада, прізвище, ініціали)

2023р.

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)
Кафедра Інфокомунікаційної інженерії імені В.В. Поповського
(повна назва)
Рівень вищої освіти другий (магістерський)
Спеціальність 125 Кібербезпека
(код і повна назва)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Адміністративний менеджмент у сфері захисту інформації
(повна назва)

ЗАТВЕРДЖУЮ
Зав. кафедри_____
(підпис)
«____» _____2023р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Полєжаєву Андрію Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Аналіз методів комбінованого шифрування бази даних із даними в стані спокою
затверджена наказом по університету від «1» травня 2023р. № 422Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 31.05.2023р.
3. Вихідні дані до роботи: бази даних, шифрування, комбіноване шифрування даних в стані спокою.
4. Перелік питань, що потрібно опрацювати в роботі:
1) Для яких баз даних можливе комбіноване шифрування
2) Науковий аналіз терміну шифрування
3) Які основні методи шифрування наявні в наш час
4) Аналіз та вибір наявних методів шифрування для створення програми
5) Створення програмного продукту для шифрування інформаційної системи

5. Перелік графічного матеріалу із зазначенням креслень, плакатів, комп'ютерних ілюстрацій: Демонстраційний матеріал у вигляді ppt-презентації;

6. Консультанти розділів роботи

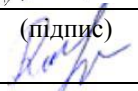
Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		(підпис)	(дата)
Основна частина	професор кафедри ІКІ ім. В.В. Поповського Радівілова Т. А.		29.05.2023

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	27.02.2023	Виконано
2	Збір матеріалів для дослідження	10.03.2023	Виконано
3	Розробка 1 розділу	15.03.2023	Виконано
4	Розробка 2 розділу	25.03.2023	Виконано
5	Розробка 3 розділу	25.04.2023	Виконано
6	Оформлення кваліфікаційної роботи	20.05.2023	Виконано

Дата видачі завдання 27 лютого 2023 року

Студент  Полєжаєв А. В.
(підпис) (прізвище, ініціали)

Керівник роботи  Радівілова Т. А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 81 с., 61 рис., 2 табл., 20 джерел.

АНАЛІЗ МЕТОДІВ КОМБІНОВАНОГО ШИФРУВАННЯ БАЗИ ДАНИХ ІЗ ДАНИМИ В СТАНІ СПОКОЮ.

Об’єкт дослідження – процес аналізу методів комбінованого шифрування бази даних із даними в стані спокою.

Предмет дослідження – методи комбінованого шифрування бази даних.

Мета роботи – аналіз методів комбінованого шифрування бази даних та створення програмного продукту.

Метод дослідження – розрахунковий аналіз та створення програмного продукту.

В кваліфікаційній роботі проаналізовані класичні методи шифрування, а також існуючі модифікації класичних методів шифрування. Наведено порівняльний аналіз класичних симетричних та асиметричних шифрів. Розроблено комбінований метод шифрування, який складається з асиметричного методу шифрування Вільямса та симетричного методу шифрування DES. Розроблений комбінований метод зберігає свої переваги перед класичним методом. Комбінований метод шифрування забезпечує підвищену швидкість роботи та підвищену стійкість до атак на основі підбору шифротексту. Він також є достатньо стійким до атак за рахунок декомпозиції відкритого ключа через його великий розмір довжини та використання симетричного і асиметричного шифрування. Також проведено аналіз сучасних концепцій створення баз даних, детально проаналізовано методи проектування баз даних, алгоритми створення баз даних та їх шифрування. Також створено програмний продукт із розробленим методом комбінованого шифрування згідно завданню.

ABSTRACT

The report contains 81 p., 61 fig., 2 tables, 20 sources.

ANALYSIS OF COMBINED DATA-AT-REST DATABASE ENCRYPTION METHODS.

The object of research is the process of analysis of methods of combined encryption of a database with idle data.

The subject of research is methods of combined database encryption.

The purpose of the work is to analyze methods of combined database encryption and create a software product.

The research method is computational analysis and creation of a software product.

The qualification work analyzes classical encryption methods, as well as existing modifications of classical encryption methods. A comparative analysis of classical symmetric and asymmetric ciphers is presented. A combined encryption method consisting of an asymmetric Williams encryption method and an symmetric DES encryption method is developed. The developed combined method retains its advantages over the classical method. The combined encryption method provides increased speed and increased resistance to attacks based on ciphertext matching. It is also sufficiently resistant to attacks due to the decomposition of the public key due to its large length and the use of symmetric and asymmetric encryption. The author also analyzes modern concepts of database creation, analyzes in detail the methods of database design, algorithms for creating databases and their encryption. Also, a software product with the developed method of combined encryption according to the task was created.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД – база даних
ІПС – інформаційно-пошукові системи
ІС – інформаційні системи
ІТ – інформаційні технології
НЛ – несанкціонована людина
НСД – найменший спільний дільник
ООБД – об’єктно-орієнтована база даних
ООСУБД – об’єктно-орієнтована система управління базами даних
РБД – реляційна база даних
СУБД – система управління базами даних
AES – Advanced Encryption Standard
DES – Data Encryption Standard
GKP – Generator Key Pare
IDE – Integrated Development Environment
MS – Microsoft Software
RMD – Relation Model Data
RSA – Rivest-Shamir-Adleman
SCM – Source Code Management
SQL – Structured Query Language
TDE – Transparent Data Encryption
WPF – Windows Presentation Foundation
XML – Extensible Markup Language

ВСТУП

В даний час практично вся організаційна діяльність пов'язана зі зберіганням і обробкою персональних даних різного типу. Найчастіше зловмисників цікавить інформація, що зберігається в сховищах підконтрольних їм державних установ і організацій. Серйозною проблемою для цих організацій є забезпечення захисту інформації, в тому числі захисту персональних даних громадян та відомостей. Отримавши доступ до бази даних, зловмисник може змінити дані, що зберігаються в базі даних, або використовувати їх у власних цілях. Для захисту бази даних від несанкціонованого доступу та зчитування використовуються засоби електронного цифрового підпису, засоби шифрування, засоби захисту від підробок, засоби виготовлення ключових файлів, а також самі ключові файли та засоби кодування.

Метою даної дипломної роботи є аналіз методів комбінованого шифрування бази даних та створення програмного продукту інформаційної системи, що включає: шифрування персональних даних, захист користувача паролем і контроль доступу до інформаційної бази. Створене рішення має забезпечити захист бази даних від читання та несанкціонованого доступу в межах однієї організації. Шифрування може проводитись різними методами. Обрані методи шифрування повинні відповідати наступним вимогам: мати високу швидкість роботи та мати високий рівень захисту.

Для досягнення поставленої мети у роботі мають бути вирішені нижчевказані завдання.

1) Аналіз існуючих баз даних.
2) Огляд мов web-програмування.
3) Огляд функцій хешування.
4) Аналіз існуючих алгоритмів шифрування.
5) Вибір алгоритму шифрування, що підходить для реалізації поставленого завдання.
6) Проектування структури БД.
7) Проектування системи шифрування БД.
8) Реалізація та тестування шифрування БД.

1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

Будь-який додаток є відображенням деякої частини реального світу і, таким чином, містить його формальний опис у вигляді даних. Великі масиви даних зазвичай розташовуються окремо від виконуваних програм і організовуються у вигляді баз даних. Починаючи з 60-х років для обробки даних почали використовувати спеціальні програмні системи, які називаються системами управління базами даних (СУБД) [1].

1.1 Поняття бази даних

Система управління базами даних відповідає за здійснення наступних дій: фізичне розташування даних та їх опис; пошук даних; постійно оновлювати базу даних; захист даних від помилкових оновлень і несанкціонованого доступу; обслуговувати одночасні запити даних від кількох користувачів.

Сховище в базі даних має певну логічну структуру, яка представлена певною моделлю, підтримуваною СУБД. Наступні моделі даних є найбільш важливими.

1) Нашарування.
2) Мережна.
3) Відношення.
4) Об'єктно-орієнтована.

В ієрархічній моделі дані представлені у вигляді деревовидної (ієрархічної) структури. Дуже зручно мати справу з ієрархічною та впорядкованою інформацією, але дуже клопітно працювати з інформацією зі складними логічними зв’язками.

Модель мережі призначена для представлення даних у вигляді довільного графа. Сильна сторона мережевих та ієрархічних моделей даних полягає в тому, що вони можуть ефективно контролювати рівень пам’яті та показники ефективності. Недолік мережевої моделі даних полягає в тому, що побудована на ній схема бази даних є дуже складною та жорсткою [1].

Назва реляційної моделі даних (RMD) походить від англійського терміну Relation-відношення. Модель даних описує певний набір загальних понять і властивостей, які повинні мати всі конкретні СУБД і керовані ними бази даних, якщо вони засновані на цій моделі.

Реляційна база даних – це набір взаємопов’язаних реляційних схем [1].

Реляційна модель даних – дозволяє представляти інформацію про предметні області за допомогою взаємопов’язаних таблиць [1].

У реляційній базі даних уся інформація підсумовується в таблицях, рядки та стовпці яких називаються записами та полями відповідно. Ці таблиці називаються відношеннями. Записи в таблиці не повторюються. Їх унікальність забезпечується первинним ключем, який складається з набору полів, які однозначно ідентифікують запис.

Приклад позначень реляційної бази даних наведено на рисунку 1.1.

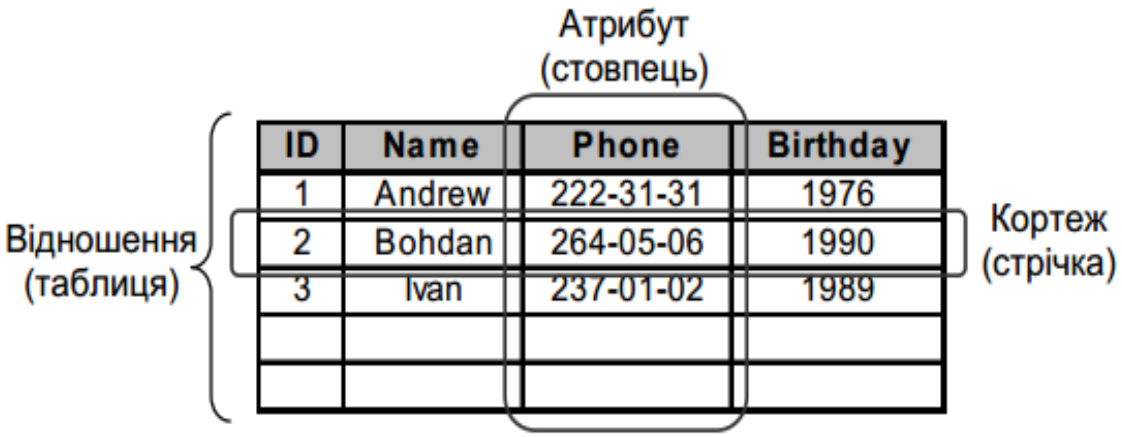


Рисунок 1.1 – Приклад реляційної бази даних

Переваги наведено нижче.

1) Ця модель даних відображає інформацію найбільш зручним для користувача способом.

2) Базується на розвиненому математичному аналізі, що дозволяє досить стисло описати основні операції над даними.

3) Дозволяє створювати мови для маніпулювання даними непроцедурного типу.

4) Маніпулювання даними на рівні вихідної бази даних і можливість зміни.

Недоліками є найповільніший доступ до даних та складність розробки.

Найбільш зручною є реляційна модель. Основні принципи реляційних баз даних можна сформулювати по пунктах далі.

1) Усі дані на концептуальному рівні представлені як упорядкована організація, визначена у вигляді рядків і стовпців і звана відношенням. Більш поширеним синонімом «зв'язку» є таблиця (або «набір записів» або набір результатів). Саме звідси походить термін «реляційні бази даних», а не зв’язки між таблицями.

2) Усі значення є скалярами. Це означає, що для будь-якого рядка і стовпця будь-якого відношення існує одне і тільки одне значення.

3) Усі операції виконуються над цілим відношенням, і результат цих операцій також є цілим відношенням. Цей принцип називається замкнутістю. Таким чином, результати однієї операції (наприклад, запиту) можна використовувати як вхідні дані для виконання іншої операції (підзапиту).

СУБД (система управління базами даних) – набір мовних та програмних інструментів, призначених для створення, обслуговування та спільного використання баз даних багатьма користувачами [2].

СУБД називають програмну систему, призначену до створення загальної бази даних, використовуваної для вирішення безлічі завдань [2].

Основними компонентами СУБД є середовище користувача, алгоритмічна мова для програмування систем обробки даних, реляційна, ієрархічна чи мережева модель даних.

Склад СУБД [3] наведено нижче.

1) Ядро – відповідає за управління даними у зовнішній та оперативній пам'яті та журналізацію.

2) Процесор мови бази даних – забезпечує оптимізацію запитів на вилучення та зміну даних та створення машинно-незалежного виконуваного внутрішнього коду.

3) Підсистема підтримки часу виконання – інтерпретує програми маніпуляції даними, що створюють інтерфейс користувача СУБД.

4) Сервісні програми – забезпечують низку додаткових можливостей обслуговування інформаційної системи. На рис. 1.2 можна побачити наочне уявлення складу СУБД.

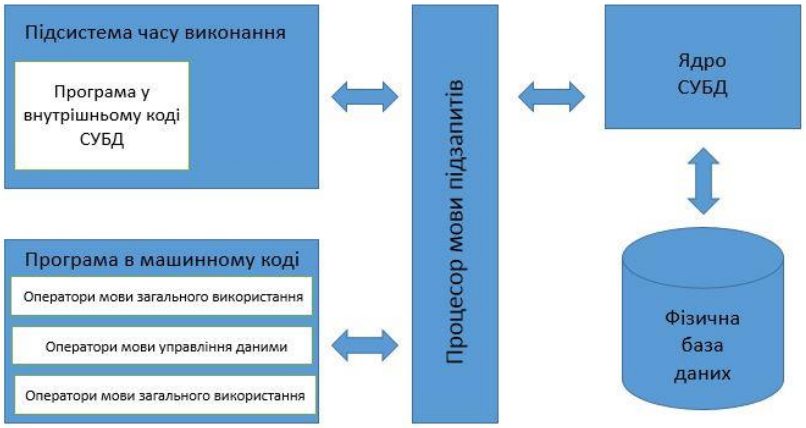


Рисунок 1.2 – Склад системи управління базами даних

Основними функціями СУБД є функції наведені нижче.

1) Керування даними у зовнішній пам'яті – дозволяє зберігати, вилучати, оновлювати інформацію, що зберігається в БД.

2) Керування даними в оперативній пам'яті з використанням дискового кешу дозволяє збільшити швидкість при зверненні до БД.

3) Управління транзакціями – дозволяє скасувати всі виконані зміни та повернути БД у вихідний стан.

4) Журналізація змін та резервне копіювання – дозволяє відновити бази даних після апаратних або програмних збоїв.

5) Підтримка мов БД [2].

Об'єктно-орієнтована модель – це коли в базі даних зберігаються не тільки дані, а й методи їх обробки у вигляді програмного коду. Це перспективний напрямок, який ще не отримав активного поширення через складність створення та використання таких СУБД.

Об’єктно-орієнтована база даних (ООБД) дозволяє програмістам, які працюють з мовами третього покоління, інтерпретувати всі свої інформаційні сутності як об’єкти, що зберігаються в оперативній пам’яті. Додатковий рівень абстракції інтерфейсу забезпечує перехоплення запитів, які звертаються до тих частин бази даних, які знаходяться в постійному зберіганні на диску. Зміни в об'єктах оптимально переносяться з пам'яті на диск [2].

Перевагою ООБД є спрощений код. Програми здатні інтерпретувати дані в контексті мови програмування, якою вони написані. Реляційна база даних повертає значення всіх полів у вигляді тексту, а потім вони перетворюються на локальні

типи даних. В ООБД ця стадія виключена. Методи маніпулювання даними завжди однакові незалежно від того, чи знаходяться дані на диску чи в пам’яті.

Дані в ООБД можуть приймати форму будь-якої структури, яка може бути виражена використовуваною мовою програмування. Відносини між сутностями також можуть бути довільно складними. ООБД керує буфером кешу об’єктів, за потреби переміщуючи об’єкти між буфером і дисковим сховищем.

ООБД вирішує дві проблеми. По-перше, складні інформаційні структури виражені в них краще, ніж в реляційних базах даних, а по-друге, відпадає необхідність перекладу даних з формату, який підтримує СУБД. Наприклад, у реляційній СУБД розмірність цілих чисел може становити одинадцять цифр, а у використовуваній мові програмування – шістнадцять. Програмісту доведеться враховувати цю ситуацію.

Об'єктно-орієнтовані СУБД виконують багато додаткових функцій. Це окупається, якщо зв’язки між даними дуже складні. У цьому випадку продуктивність ООБД вище, ніж у реляційної СУБД. Якщо дані менш складні, додаткові функції зайві. Спеціальні запити підтримуються в моделі об’єктів даних, але їх мова не обов’язково є SQL. Логічне представлення даних може не відповідати реляційній моделі, тому використання мови SQL втратить сенс. Зручніше обробляти об'єкти в пам'яті, виконуючи відповідний пошук.

Великим недоліком об'єктно-орієнтованих баз даних є їх тісний зв'язок з використовуваною мовою програмування. Дані, що зберігаються в реляційній СУБД, можуть бути доступні для будь-якої програми, тоді як, наприклад, об'єкт Java, розміщений в ООБД, буде цікавий тільки для програм, написаних на Java.

Деякі об’єктно-орієнтовані бази даних розроблені для тісної взаємодії з об’єктно-орієнтованими мовами програмування, такими як Python, Java, Visual Basic .NET, C++, Objective-C і Smalltalk; інші мають власні мови програмування. ООСУБД використовують ту саму модель, що й об’єктно-орієнтовані мови програмування.

Приклади ООСУБД – ORION і O2.

Об'єктно-орієнтовані бази даних зазвичай рекомендуються для тих випадків, коли потрібна високопродуктивна обробка даних зі складною структурою.

ООБД має обов'язкові характеристики. Їх вибір ґрунтується на 2 критеріях: система повинна бути об'єктно-орієнтованою і бути базою даних.

Обов'язкові характеристики наведено нижче.

1) Підтримка складних об'єктів. У системі має бути передбачена можливість створення складених об'єктів за допомогою конструкторів складених об'єктів. Необхідно, щоб конструктори об'єктів були ортогональними, тобто будь-який конструктор можна застосувати до будь-якого об'єкта.

2) Підтримка індивідуальності об'єктів. Усі об’єкти повинні мати унікальний ідентифікатор, який не залежить від значень їхніх атрибутів.

3) Підтримка інкапсуляції. Коректна інкапсуляція досягається за рахунок того, що програмісти мають право доступу тільки до специфікації інтерфейсу методів, а дані і реалізація методів приховані всередині об'єктів.

4) Підтримка типів і класів. Необхідно, щоб ООСУБД підтримував принаймні одну концепцію відмінності між типами та класами. Термін «тип» більше відповідає концепції абстрактного типу даних. У мовах програмування змінна оголошується разом із її типом. Компілятор може використовувати цю інформацію, щоб перевірити, що операції, що виконуються над змінною, сумісні з її типом , що допомагає забезпечити правильність програмного забезпечення. З іншого боку, клас є шаблоном для створення об’єктів і надає методи, які можна застосувати до цих об’єктів. Таким чином, концепція «класу» стосується більше часу виконання, ніж компіляції.

5) Підтримка успадкування типів і класів від предків. Підтип або підклас повинен успадкувати атрибути та методи від свого супертипу або суперкласу відповідно.

6) Перевантаження в поєднанні з повним зв'язуванням. Методи повинні застосовуватися до об'єктів різних типів. Реалізація методу повинна залежати від типу об'єктів, до яких застосовується метод. Щоб забезпечити цю функціональність, зв’язування імен методів у системі не має відбуватися до моменту виконання програми.

7) Обчислювальна повнота. Мова обробки даних має бути мовою програмування загального призначення.

Набір типів даних має бути розширюваним. Користувач повинен мати засоби для створення нових типів даних на основі набору попередньо визначених системних типів. Крім того, не повинно бути різниці між способом використання системних і визначених користувачем типів даних.

Бази даних – це сукупність записів різного типу, що містять перехресні посилання [2].

Файл – це сукупність однотипних записів, у яких відсутні перехресні посилання [2].

Крім того, у визначенні немає жодної згадки про архітектуру комп'ютера. Справа в тому, що, хоча в більшості випадків база даних дійсно складається з одного або (частіше) кількох файлів, їх фізична організація істотно відрізняється від логічної. Таблиці можна зберігати в окремих файлах або разом. І навпаки, іноді для зберігання однієї таблиці використовується декілька файлів. Додаткові спеціальні файли зазвичай виділяються для підтримки перехресних посилань і швидкого пошуку.

Тому при роботі з базами даних зазвичай використовують поняття більш високого логічного рівня: запис і таблиця, не заглиблюючись у деталі їх фізичної структури.

Таким чином, сама база даних є просто набором таблиць із перехресними посиланнями. Щоб універсально витягувати з нього групи записів, обробляти, змінювати та видаляти, потрібні спеціальні програми, які називаються СУБД.

За характером використання СУБД поділяються на персональні і багатокористувацькі (СУБД).

До персональних СУБД відносяться VISUAL FOXPRO, ACCESS та ін. До багатокористувацьких СУБД відносяться, наприклад, СУБД ORACLE і INFORMIX.

Багатокористувацька СУБД включає сервер бази даних і клієнтську частину, вони працюють в неоднорідному обчислювальному середовищі, допускаються різні типи комп'ютерів і різні операційні системи. Отже, на базі СУБД можна створити інформаційну систему. Функціонування за технологією клієнт-сервер. Універсальність багатокористувацької СУБД відображається, відповідно, у високій ціні та ресурсах комп'ютера, необхідних для підтримки.

Персональні СУБД – це набір мовних і програмних засобів, призначених для створення, підтримки та використання бази даних [3].

Для обробки команд користувача або операторів програми в СУБД використовуються інтерпретатори команд (оператори) і компілятори. За допомогою компіляторів в ряді СУБД можна отримати автономні додатки – exe-програми.

Забезпечення цілісності бази даних є необхідною умовою успішного функціонування бази даних. Цілісність БД – властивість бази даних, яка означає,

що база даних містить повну та послідовну інформацію. Для забезпечення цілісності бази даних накладаються обмеження цілісності у вигляді певних умов, яким повинні задовольняти дані, що зберігаються в базі даних. Прикладом таких умов є обмеження діапазонів можливих значень атрибутів об'єктів, інформація про які зберігається в базі даних, або відсутність дублюючих записів у таблицях реляційної бази даних.

Безпека СУБД досягається шляхом шифрування прикладних програм, даних, захисту паролем, підтримки рівнів доступу до бази даних, до окремої таблиці.

Розширення користувальницьких можливостей СУБД досягається з'єднанням дистрибутивних систем C і Assembler.

Робота мережі підтримується нажченаведеними засобами.

1) Засоби контролю доступу користувачів до спільних даних, тобто засоби блокування файлів (таблиць), записів, полів, які різною мірою реалізовані в різних СУБД.

2) Засоби механізму транзакцій, що забезпечують цілісність бази даних при роботі в мережі [3].

Тепер розглянемо функції СУБД трохи докладніше.

1) Визначення даних.

СУБД повинна приймати визначення даних (зовнішні схеми, концептуальну схему, внутрішню схему, а також усі пов’язані відображення) в їх вихідній формі та перетворювати ці визначення у форму відповідних об’єктів. Іншими словами, СУБД повинна включати компонент мовного процесора для різних мов визначення даних. СУБД також повинна «розуміти» синтаксис мови визначення даних.

2) Обробка даних.

СУБД повинна мати можливість обробляти запити користувачів на вибір, зміну чи видалення існуючих даних у базі даних або додавання нових даних до бази даних. Іншими словами, СУБД повинна включати компонент процесора мови обробки даних.

Запити мови обробки даних бувають «заплановані» та «не заплановані».

Запланований запит – це запит, який, як очікується, буде потрібний заздалегідь. Адміністратору баз даних може знадобитися налаштувати фізичний дизайн бази даних таким чином, щоб гарантувати достатню продуктивність для таких запитів [3].

Незапланований запит – це, навпаки, особливий запит, необхідність якого не була заздалегідь передбачена. Фізичний дизайн бази даних може підходити або не підходити для конкретного запиту, який розглядається. Загалом, отримання максимально можливої продуктивності для незапланованих запитів є одним із завдань СУБД [3].

3) Безпека та цілісність даних.

СУБД повинна контролювати запити користувачів і припиняти спроби порушити правила безпеки та цілісності, визначені адміністратором бази даних.

4) Відновлення та дублювання даних.

СУБД або інший відповідний компонент програмного забезпечення, який зазвичай називають менеджером транзакцій, повинен здійснювати необхідний контроль над відновленням і реплікацією даних.

5) Словник даних.

СУБД повинна забезпечувати функцію словника даних. Сам словник даних по праву можна вважати базою даних (не базою даних користувача, а системою). Словник «містить дані про дані» (іноді їх називають метаданими), тобто визначення інших об’єктів системи, а не лише «необроблені дані». Зокрема, у словнику зберігатимуться вихідні та об’єктні форми різних схем (зовнішніх, концептуальних тощо) і відображень. Розширений словник також включатиме перехресні посилання, які показуватимуть, наприклад, які програми використовують яку частину бази даних, які звіти потрібні тому чи іншому користувачеві, які термінали підключені до системи тощо. Словник може бути (і навіть повинен бути) інтегрований у базу даних, яку він визначає, що означає, що він повинен містити опис самого себе. Звичайно, слід мати можливість отримати доступ до словника так само, як до іншої бази даних, наприклад, щоб дізнатися, на які програми та/або користувачів вплинуть запропоновані зміни в системі.

6) Продуктивність.

Очевидно, що всі ці функції СУБД повинна виконувати з максимально можливою ефективністю.

База знань – база даних, що містить правила висновку та інформацію про людський досвід і знання в деякій предметній області. У самонавчальних системах база знань також містить інформацію, яка є результатом вирішення попередніх завдань [3].

Сучасні бази знань працюють у поєднанні з інформаційно-пошуковими та пошуковими системами. Це вимагає певної моделі для класифікації понять і певного формату для представлення знань. Ієрархічний спосіб представлення набору понять та їхніх зв’язків у базі знань називається онтологією.

Онтологію певної галузі знань разом з інформацією про властивості конкретних об’єктів часто називають «базою знань». У той же час повноцінні бази знань (на відміну від звичайної бази даних) містять не тільки фактичну інформацію, але й правила логічного висновку, які дозволяють робити автоматичні висновки щодо наявних або нововведених фактів і тим самим виробляти семантичну (змістовну) обробку інформації.

Область штучного інтелекту, яка вивчає бази знань і методи роботи зі знаннями, називається інженерією знань.

База знань – це особливий вид бази даних, призначений для обробки знань (метаданих). База знань містить структуровану інформацію, що охоплює певну область знань для використання кібернетичним пристроєм (або людиною) з певною метою. Сучасні бази знань працюють спільно з інформаційно-пошуковими системами, мають структуру класифікації та формат подання знань [4].

Повноцінні бази знань містять не тільки фактичну інформацію, а й правила логічного висновку, які дозволяють автоматично робити висновки щодо нововведених фактів і, як наслідок, осмислену обробку інформації. Галузь науки про штучний інтелект, яка вивчає бази знань і методи роботи зі знаннями, називається, так само, інженерією знань.

Ієрархічний спосіб представлення набору понять та їхніх зв’язків у базі знань називається онтологією. Онтологію певної галузі знань разом з інформацією про властивості конкретних об’єктів також можна назвати базою знань.

Відмінності наведено нижче.

1) База знань – це семантична модель, яка описує предметну область і дозволяє відповідати на такі питання з цієї предметної області, відповіді на які явно не присутні в базі. Основним компонентом інтелектуальних та експертних систем є база знань [4].

2) База даних – сукупність пов'язаних даних, організованих за певними правилами, що передбачають загальні принципи опису, зберігання та маніпулювання, незалежно від прикладних програм. База даних є інформаційною

моделлю предметної області. Доступ до баз даних здійснюється за допомогою системи керування базами даних (СУБД) [3].

1.2 Проблеми кібербезпеки баз даних

Проблеми безпеки становлять серйозну загрозу для будь-якої системи, тому важливо знати її вразливі місця. Бази даних є дуже привабливою мішенню для хакерів, оскільки вони містять цінну конфіденційну інформацію. Це може варіюватися від фінансової чи інтелектуальної власності до корпоративних і особистих даних користувачів. Кіберзлочинці можуть заробляти гроші, зламуючи сервери компаній та компрометуючи при цьому бази даних. Тому перевірка безпеки бази даних є обов’язковою.

Розглянемо найпоширеніші проблеми.

1) Уразливості до генерування підроблених даних.

Кіберзлочинці можуть підробляти або генерувати дані та надсилати їх у систему, де вони зберігатимуться разом із дійсними даними. Наприклад, якщо ваша виробнича компанія використовує дані датчиків для виявлення неправильних виробничих процесів, зловмисник може проникнути у вашу систему та змусити датчики відображати помилкові результати, наприклад неправильні температури.

Таким чином ви можете ігнорувати тривожні тенденції та втрачати можливості вирішити проблеми до того, як буде завдано серйозної шкоди. Для захисту від таких вразливостей можна використовувати методи виявлення шахрайства.

2) Метод захисту паролем.

Хоча шифрування є добре відомим методом захисту конфіденційної інформації, воно також входить до списку проблем безпеки даних. Незважаючи на можливість і важливість шифрування інформації, цей захід безпеки часто ігнорується. Особисті дані зазвичай зберігаються в хмарі без будь-якого захисту від зламу. Причина такої недбалості проста: постійне шифрування та дешифрування навіть невеликих фрагментів даних уповільнює процес, тим самим втрачаючи важливу перевагу - швидкість.

3) Можливість отримати важливу інформацію.

У системах часто використовується захист його кордонів. Це означає, що всі «точки входу та точки виходу» захищені. Але, що саме ІТ-спеціалісти роблять у вашій системі, залишається загадкою.

Відсутність контролю над рішеннями для роботи з даними може дозволити корумпованим ІТ-фахівцям або шахраям-конкурентам видобувати незахищені дані та продавати їх з метою прибутку. Якщо така інформація стосується запуску нових продуктів/послуг, фінансових операцій компанії або особистої інформації користувачів, ваша компанія може зазнати великих збитків.

Так, дані можна краще захистити, додавши додаткові межі. Крім того, безпеку вашої системи можна покращити за допомогою анонімізації. Якщо у когось є особисті дані, у яких відсутні імена, адреси та номери телефонів ваших користувачів, вони можуть завдати невеликої шкоди.

4) Жодних перевірок безпеки перед розгортанням.

Однією з найпоширеніших причин уразливості бази даних є необережність на етапі розгортання. Хоча функціональне тестування проводиться для забезпечення оптимальної продуктивності, цей тип тестування не може визначити, чи ваша база даних робить те, чого не повинна робити. Тому дуже важливо перевірити безпеку веб-сайту за допомогою різних типів тестів перед повним розгортанням.

5) Викрадені резервні копії баз даних.

Бази даних стикаються з двома типами загроз: зовнішніми та внутрішніми. У деяких випадках компанії борються з внутрішніми загрозами навіть більше, ніж із зовнішніми.

Власники бізнесу ніколи не можуть бути на 100% впевнені в лояльності своїх співробітників, незалежно від того, яке програмне забезпечення для комп’ютерної безпеки вони використовують і наскільки сумлінно вони працюють. Кожен, хто має доступ до конфіденційних даних, може викрасти їх і продати третім особам з метою отримання прибутку. Однак існують можливості для усунення ризиків:

Файли бази даних зашифровані, застосовуються суворі стандарти безпеки, накладаються штрафи за порушення, використовується програмне забезпечення для кібербезпеки, а в команді постійно підвищується обізнаність за допомогою корпоративних зустрічей та індивідуальних консультацій [4].

Водночас, з точки зору практичного застосування, найважливішим питанням є визначення терміну «кібербезпека», який, залежно від його розуміння, впливає на

формування предметних кіл, які він обумовлює на національному та міжнародному рівнях.

Останніми роками кібербезпека стала широко вживаним терміном.

Однак, як і у випадку з багатьма популярними «жаргонами», здається, мало розуміння того, що насправді означає цей термін. Хоча використання терміну в неофіційному контексті не викликає труднощів, воно може створити значні проблеми в контексті організаційної стратегії, бізнес-цілей або міжнародних угод.

З використанням терміну «кібербезпека» почали набирати популярності нові терміни. Він використовувався в попередні роки, але його популярність значно зросла, коли президент США Барак Обама закликав зробити кібербезпеку невід'ємною частиною національної безпеки в 2009 році: «Я закликаю американський народ визнати важливість, відзначити цей момент відповідними заходами , програмами та навчанням для зміцнення нашої національної безпеки та стійкості» [12].

Безпосередній вплив цього прес-релізу на термін ілюструється тенденціями пошуку Google, які значно зросли за цей період. Лінія тренду на графіку показує загальну кількість пошукових запитів за терміном порівняно із загальною кількістю пошукових запитів у Google за певний час. Ми спостерігали постійне зниження пошукових термінів «комп'ютерна безпека» та «інформаційна безпека» порівняно з «кібербезпекою», які зображені на рисунку 1.3.

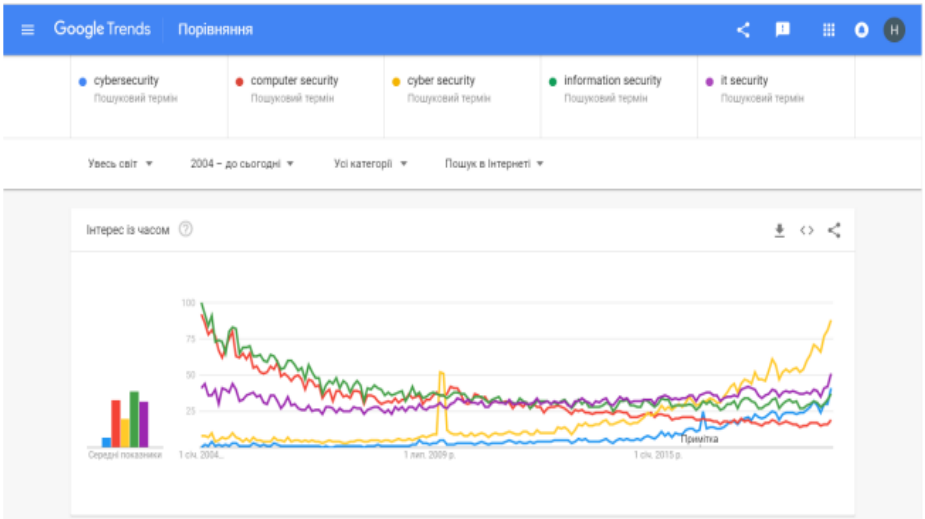


Рисунок 1.3 – Частота використання таких термінів, як «кібербезпека»

У науковому виданні Helen Meyer вперше використала цей термін у книзі «Комп’ютери та безпека». Визначення цього терміну не надано [12].

Оксфордські словники конкретно визначають «кібербезпеку» як стан захисту від злочинного чи несанкціонованого використання електронних даних або заходи, вжиті для цього [12].

«Кібербезпека: захист цифрової інформації, а також цілісності інфраструктури та передачі цифрової інформації. Зокрема, кібербезпека включає серію політик, призначених для захисту мереж, комп’ютерів, програм і даних від атак, пошкоджень або несанкціонованого доступу. Технології , процеси, практики, відповіді та пом’якшення для забезпечення конфіденційності, цілісності та доступності» [12].

1.3 Шифрування даних

Шифрування даних – це метод безпеки, у якому інформація кодується та може бути відновлена або розшифрована лише користувачем із правильним ключем розшифровки. Зашифровані дані, також відомі як зашифрований текст, здаються зашифрованими або нечитабельними для неавторизованих осіб або організацій [5].

Шифрування даних використовується для запобігання доступу зловмисників або небажаних осіб до конфіденційних даних. Важлива лінія захисту в архітектурі кібербезпеки – шифрування – максимально ускладнює використання перехоплених даних. Його можна застосовувати для захисту різноманітних даних, від конфіденційних державних документів до операцій з особистими кредитними картками. Програмне забезпечення для шифрування даних, також відоме як алгоритм шифрування або шифр, використовується для розробки схеми шифрування, яку теоретично можна обійти лише завдяки великій обчислювальній потужності.

Шифрування використовується для зберігання важливої інформації в ненадійних джерелах і передачі її через незахищені канали зв’язку. Ця передача даних складається з двох протилежних процесів:

Перш ніж дані надсилаються по лініях зв’язку або зберігаються, вони шифруються.

Хоча дані все ще можуть бути перехоплені, вони будуть приховані, тому вони не будуть корисними для шпигунів або хакерів.

Різні пристрої в різних мережах шифрують повідомлення, якими вони обмінюються. Шифрування використовується не лише для передачі через Інтернет, але й для операцій через банкомат або мобільний телефон. Алгоритми шифрування захищають усі передані дані.

Щоб відновити оригінальні дані із зашифрованих даних, ви можете застосувати до них процес дешифрування.

Шифр – це пара алгоритмів, які реалізують кожне вказане перетворення. Ці алгоритми застосовуються до даних за допомогою ключів. Ключ шифрування та ключ дешифрування можуть бути різними або однаковими [5].

Секретність другого робить дані недоступними для несанкціонованого читання, а секретність першого запобігає введенню неправдивих даних. Перший метод шифрування використовував той самий ключ, але в 1976 році було виявлено алгоритм, який використовував інший ключ. Зберігати ці ключі в секреті та належним чином ділитися ними між одержувачами є нетривіальним завданням з точки зору захисту конфіденційності переданої інформації - це складне завдання.

В даний час запропоновано велику кількість методів шифрування.

Ці методи в основному поділяються на симетричні та асиметричні, залежно від структури використовуваного ключа. Крім того, методи шифрування можуть мати різну стійкість до шифру та по різному обробляти вхідні дані – блокові шифри та потокові шифри. Криптографія займається всіма цими методами, їх створенням і аналізом.

Само собою зрозуміло, що ключі є фундаментальною частиною збереження конфіденційності інформації, повідомлень або фрагментів даних. Процес шифрування та дешифрування можна розпочати лише за допомогою ключа.

Ключі шифрування/дешифрування можна порівняти зі звичайними паролями, наприклад, тими, які ви використовуєте для електронної пошти. Ключі є невід'ємною частиною процесу кодування та декодування даних.

Ключі шифрування створені як абсолютно унікальні за допомогою набору різних алгоритмів. Ключі шифрування використовуються для кодування або декодування даних. Це означає, що ключ шифрування здатний шифрувати дані в нечитабельні символи та перетворювати ці символи назад у відкритий текст.

Як правило, ключ – це випадковий двійковий файл або справжній пароль. Ключ «доповідає» алгоритму, який шаблон слідувати, щоб перетворити відкритий текст на зашифрований (і навпаки).

Оскільки алгоритм загальнодоступний і доступний кожному, якщо хакер отримає ключ шифрування, він може легко розшифрувати зашифровані дані у відкритий текст.

Симетричні або секретні ключі використовують один і той же ключ для кодування та декодування інформації [5]. Найкраще використовувати під час обміну невеликими обсягами даних один з одним. Хоча симетричне шифрування набагато швидше, ніж асиметричне шифрування, відправник повинен обмінятися ключами шифрування з одержувачем, перш ніж одержувач зможе розшифрувати повідомлення як зображено на рисунку 1.4.

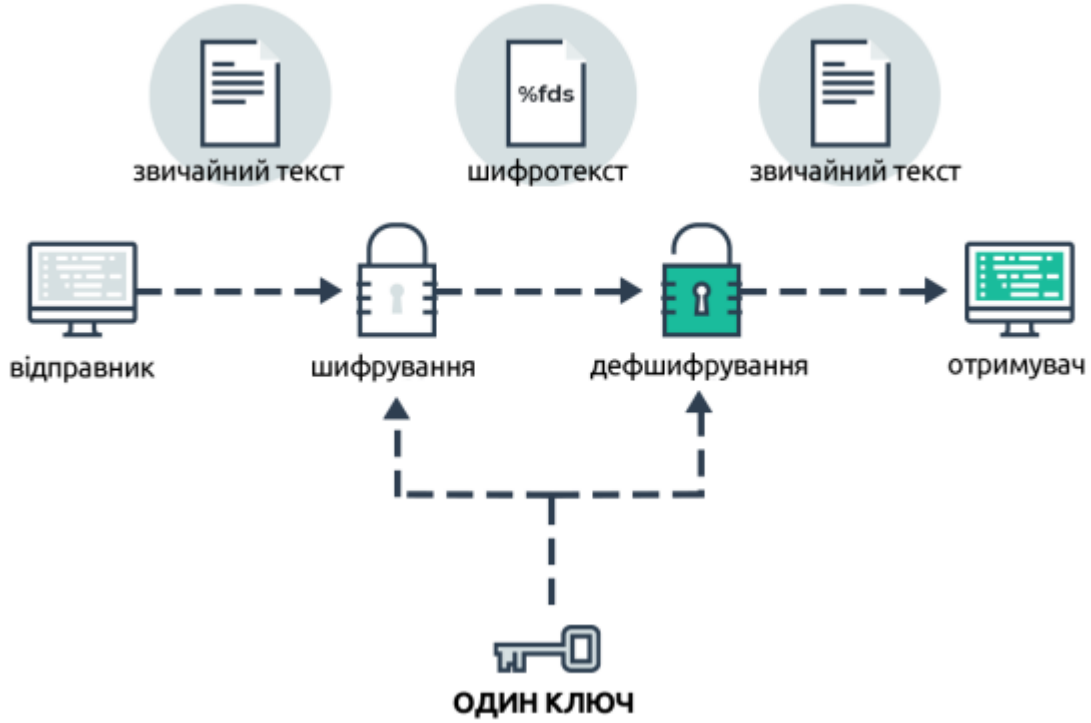


Рисунок 1.4 – Схема шифрування симетричним алгоритмом

Мета шифрування бази даних полягає в тому, щоб захистити інформацію, що зберігається в базі даних, від доступу зловмисників, а також від крадіжки, оскільки після того, як дані зашифровано, вони фактично стають марними. Для шифрування бази даних можна використовувати різні методи та прийоми.

Прозоре шифрування даних (TDE) є прозорим з точки зору користувача:

Йому не потрібен спеціальний ключ для доступу до захищених даних. Методи шифрування захищають інформацію в базі даних (БД), шифруючи базові файли бази даних, а не самі дані. Це запобігає злому та копіюванню даних на інший сервер. Щоб відкрити файл, ви повинні мати оригінальний сертифікат шифрування та головний ключ.

Для безпеки TDE зберігає ключі шифрування у зовнішньому (на відміну від бази даних) модулі безпеки (сховище ключів) [5]. Програми не можна модифікувати, щоб TDE працював належним чином (резервні копії бази даних також зашифровані) [5]. Весь процес шифрування повністю прозорий для додатків, які отримують доступ до бази даних за допомогою Advanced Encryption Standard (AES), який шифрує сторінки файлу, а потім розшифровує інформацію, коли вона надходить у пам’ять. По суті, це шифрування та дешифрування введення/виведення в реальному часі. AES не збільшує розмір зазначеної бази даних. Нижче на рис. 1.5 показана ієрархія TDE.

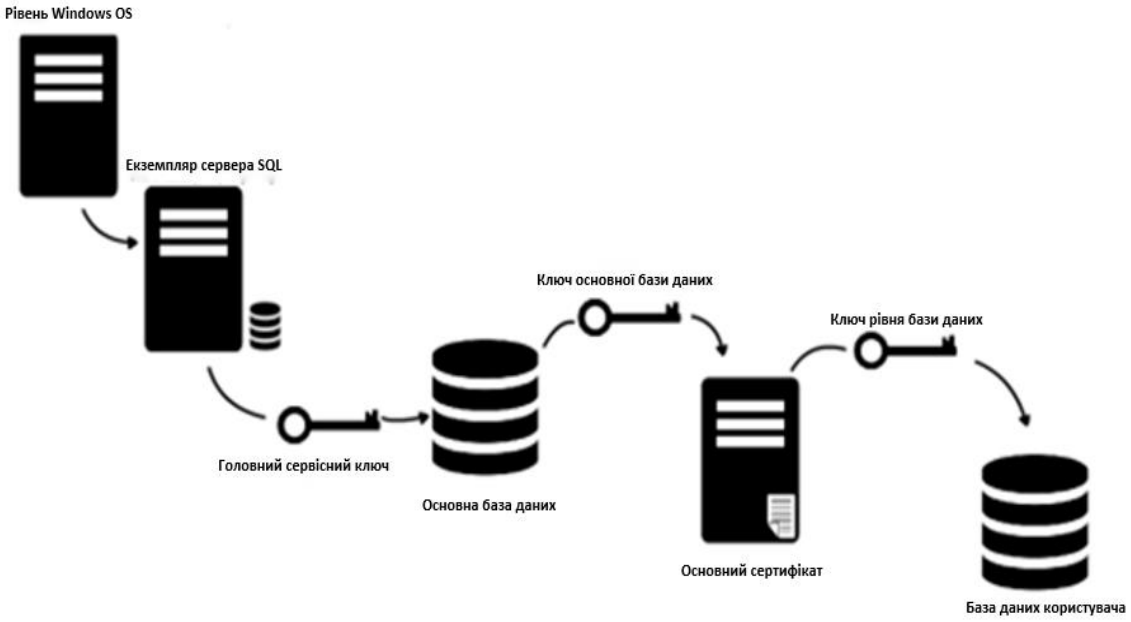


Рисунок 1.5 – Ієрархія Transparent Data Encryption

Після шифрування користувачі можуть переглядати базу даних так само, як якщо б вона не була зашифрована [5].

1) Шифрування на рівні стовпців.

Типовою реляційною базою даних є таблиці, які поділені на стовпці з рядками даних.

Переваги цього методу шифрування: можливість шифрувати окремі стовпці більш корисна, ніж шифрувати всю базу даних; для кожного стовпця в базі даних (окремо) можна використовувати абсолютно унікальний ключ шифрування [5].

Технологія була прийнята кількома компаніями з програмного забезпечення для шифрування по всьому світу, включаючи IBM, MyDiamo (Penta Security), Oracle.

Симетричне шифрування в контексті шифрування бази даних досягається за допомогою методу закритого ключа [5]. Цей закритий ключ змінює інформацію таким чином, що її неможливо прочитати без попереднього розшифрування. Принцип зображено на рисунку 1.6.

Рисунок 1.6 – Схема симетричного шифрування

Якщо користувач знає секретний ключ, інформація шифрується під час збереження та розшифровується під час відкриття. Тому, щоб розшифрувати дані, одержувач повинен мати копію закритого ключа, який використовує відправник.

Асиметричне шифрування, зображене вище на рис. 1.7, розширює симетричне шифрування, включаючи два різні типи ключів у метод шифрування: приватний і відкритий. Відкритий ключ – це секретний ключ, який доступний будь-кому та унікальний для одного користувача, тоді як закритий ключ — це секретний ключ, унікальний і відомий лише одному користувачеві. У більшості випадків відкритий ключ є ключем шифрування, а закритий – ключем дешифрування [5].

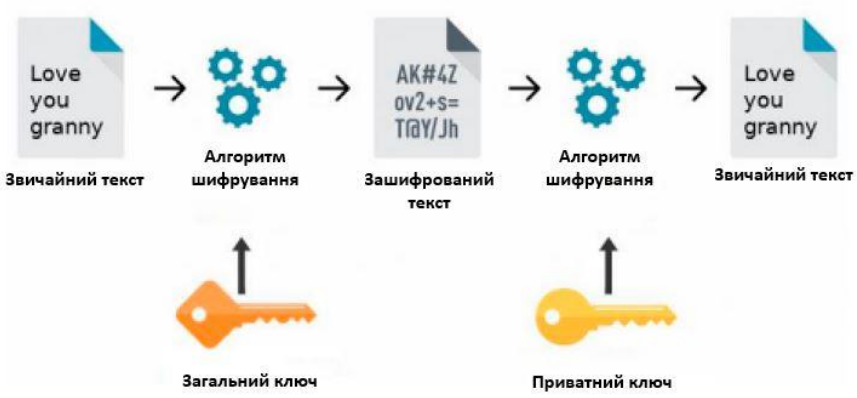


Рисунок 1.7 – Схема асиметричного шифрування

Асиметричне шифрування часто описують як більш безпечне, ніж симетричне шифрування бази даних, оскільки закритий ключ не потрібно використовувати спільно (два окремих ключі обробляють процес шифрування та дешифрування).

2) Шифрування на рівні поля.

Триває експериментальна робота з представлення операцій бази даних (таких як пошук або арифметичні операції) у зашифрованих полях без необхідності дешифровки. Надійне шифрування потребує різного результату щоразу, коли рандомізується. Це називається імовірнісним шифруванням. Прикладом поля, яке потрібно зашифрувати, є номер кредитної картки. Коли поле вибрано, усі дані в цьому полі автоматично шифруються [5].

1.4 Вибір алгоритмів шифрування

На зорі криптографії переважали шифри перестановки (зміна порядку букв у повідомленні) та шифри заміни (системи, що передбачають заміну однієї літери або символу на іншу). Ці класичні шифри забезпечують мінімальний захист [6].

Найвідомішим із ранніх альтернативних шифрів є шифр Цезаря. Суть його в тому, щоб замінити кожну літеру алфавіту іншою літерою, і змінити алфавітний порядок. Гай Юлій Цезар використав цей трьох-бітний шифр зсуву для передачі повідомлень між собою та своїми генералами під час військових кампаній.

Всі перераховані вище шифри досі абсолютно не захищені криптоаналізом з використанням частотного аналізу. Ситуація змінилася з винаходом багатолітерних кодів. Дуже популярний багатолітерний шифр Вігенера, який

використовує ключові слова для керування заміною літер залежно від того, які літери використовуються у ключовому слові. У середині 1800-х років Чарльз Беббідж продемонстрував, що цей тип багатолітерного шифру все ще частково піддається частотному аналізу. Показано, що зберігання ключів у секреті є достатньою умовою захисту інформації традиційними криптографічними схемами. Це фундаментальний принцип криптографії, вперше запропонований та названий на честь Огюста Кекхофса у 1883 році.

З винаходом електроніки, а потім цифрового комп'ютера можливості шифрування розширилися, і з'явилися нові, складніші шифри. Крім того, будь-які дані, які можуть бути подані у двійковій формі на комп'ютері, можуть бути зашифровані, а не тільки письмовий текст, як це було раніше.

Перші ідеї, які лягли в основу асиметричної криптографії, з'явилися в 1976 в роботі Уітфілда Діффі і Мартіна Геллмана «Нові напрямки в сучасній криптографії». У своїй роботі вони пропонують новий спосіб організації секретних комунікацій без попереднього обміну ключами, званий криптографією з відкритим ключем. Цей метод використовує різні ключі для шифрування та дешифрування, що унеможливлює пошук одного, якщо інший існує. Завдяки цьому ключ шифрування може бути оприлюднений, так як це не відбиватиметься на стійкості шифру, тільки ключ дешифрування повинен бути пов'язаний з одержувачем, тому криптосистеми з відкритим ключем називаються асиметричними криптосистемами.

На рис. 1.8 показано блок-схему криптосистеми з відкритим ключем.

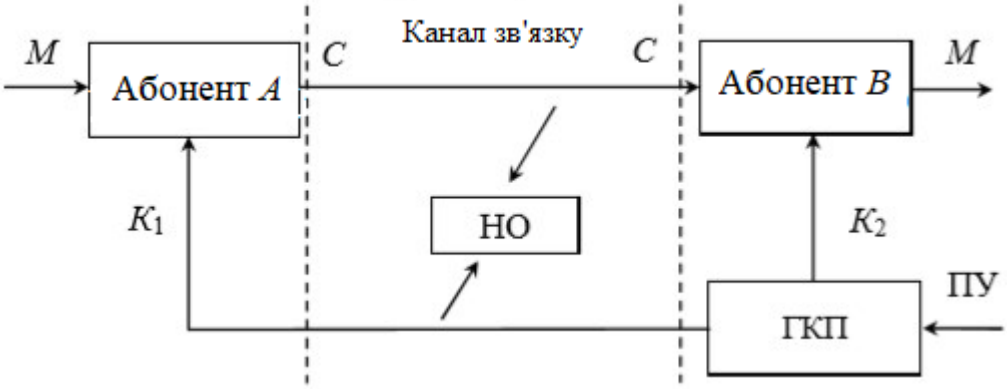


Рисунок 1.8 – Структурна схема асиметричної криптосистеми

GKP, генератор пари ключів, створює пару ключів (K1, K2) з урахуванням початкових умов, які знає лише одержувач повідомлення. Відкритий ключ K1

відправляється відправнику незахищеним каналом зв'язку. Відправник шифрує повідомлення М ключем К1. Зашифрований текст С (шифр) пересилається одержувачу незахищеним каналом зв'язку, який потім розшифровує шифр за допомогою ключа К2, відновлюючи вихідне повідомлення. Канал зв'язку незахищений, тому несанкціонована людина (НЛ), маючи доступ до них, може перехопити криптограму і відкритий ключ К1. Також, НЛ може мати алгоритм шифрування.

Однак єдине, що він не має – це ключ К2, без якого розшифрувати криптограму С не вийде.

Найвідоміші асиметричні криптосистеми: криптосистема RSA, криптосистема Діффі-Хеллмана, криптосистема Меркла-Хеллмана, криптосистема Ель-Гамалю, криптографічні системи на еліптичних кривих, електронний цифровий підпис [7].

Розглянемо основні алгоритми далі.

1) Алгоритм Меркле-Геллмана.

Криптосистема була розроблена Ральфом Мерклем та Мартіном Геллманом у 1978 році. Він став першим алгоритмом шифрування, що відкидав, що набув широкого поширення. Криптосистема Меркла-Хеллмана відноситься до алгоритму “рюкзака”. Спочатку алгоритм використовував лише шифрування повідомлень, але пізніше Аді Шамір модифікував відповідну криптосистему підтримки інструментів цифрового підпису. Безпека алгоритму рюкзака ґрунтується на відомій математичній проблемі упаковки рюкзака. В основі алгоритму лежить ідея шифрування повідомлень шляхом вирішення ряду завдань, пов'язаних із упаковкою рюкзака. Вирішити завдання п'яти елементів дуже просто за допомогою алгоритму рюкзака. Щоб бути практичним, рюкзак повинен містити 250 або більше елементів і кожен елемент повинен бути в діапазоні від 200 до 400 біт. Довжина модуля має бути від 100 до 200 символів. Для отримання таких значень практично використовуються генератори випадкових послідовностей. Ця криптосистема була зламана Шаміром і Циппелем, які змогли виявити її лазівки і відновити послідовність рюкзаків, що швидко росте, через звичайну послідовність рюкзаків. Пізніше було створено більше криптосистем з використанням алгоритму ранцевого укладання, але жодна з них не виявилася надійною та стійкою до зламу.

Суть криптографічного алгоритму Меркла-Хеллмана, що ґрунтується на задачі «наповнення рюкзака», зводиться до наступного [8]. Абоненти, що отримують інформацію, вибирають початковий вектор в формулі (1.1):

$$W = (w_1, w_2, \dots, w_n), \tag{1.1}$$

що складається з елементів (випадкових чисел) w_i, де i = 1,2,..n, щоб задовольнити вимоги послідовності, що швидко росте. Потім одержувач вибирає просте число r, значення якого більше суми та деякого обраного числа (не обов'язково простого числа), що задовольняє умовам формули (1.2) і формули (1.3):

$$r \leq p - 1, \tag{1.2}$$

$$НСД(r, p) = 1. \tag{1.3}$$

Потім він генерує відкритий ключ K за формулою (1.4) нижче:

$$k_i = (w_i \cdot r) \cdot \mathrm{mod}\ p,\ i = 1,2, \dots, n, \tag{1.4}$$

де k_i – відкритий ключ;

w, r, p – прості числа.

Значення ключового елемента передається відправнику повідомлення відкритими каналами зв'язку. Значення початкового вектора W та чисел r та p запам'ятовує лише одержувач повідомлення. Відправник ділить зашифроване повідомлення M на шматки n символів за формулою (1.5):

$$M = (m_1, m_2, \dots, m_n), \tag{1.5}$$

де $m_i = \{1,0\}$ – частина зашифрованого повідомлення;

$i = 1, 2, \dots, n$ – число символів.

Відправник використовує ключ для шифрування повідомлення використовується така формула (1.6):

$$C = K \cdot M = \sum_{i=1}^{n} k_i \cdot m_i, \tag{1.6}$$

де К – шифр;

М – повідомлення.

Потім відправник передає зашифрований текст одержувачу на відкритому каналі. Рівняння рішення в формулі (1.7):

$$r \cdot e = 1 \cdot (\mathrm{mod}\ p), \tag{1.7}$$

де $r \cdot e$ – відкритий канал.

Відносно, тобто знайти обернену величину числа за модулем. Потім отримана криптограма перетворюється за формулою (1.8):

$$C' = (C \cdot e) \cdot \mathrm{mod}\ p, \tag{1.8}$$

де C' – криптограма.

Наша C' повинна мати повну форму в формулі (1.9):

$$C' = (C \cdot e) \cdot \mathrm{mod}\ p = (\sum_{i=1}^{n} k_i \cdot m_i \cdot e) \cdot \mathrm{mod}\ p =$$

$$= (\sum_{i=1}^{n} m_i \cdot w_i \cdot r \cdot e) \cdot \mathrm{mod}\ p = \sum_{i=1}^{n} w_i \cdot m_i. \tag{1.9}$$

Далі необхідно вирішити задачу визначення суми підмножин.

Шифр, запропонований Аді Шаміром, дозволяє обмінюватися секретними повідомленнями відкритими лініями зв'язку без використання захищених каналів і ключів. Цей шифр повністю вирішує проблему обміну закритими для читання повідомлень у ситуаціях, коли абоненти не можуть використовувати закриті лінії зв'язку. Але у схеми Шаміра є очевидна вада, так як повідомлення пересилається три рази між абонентами.

2) Алгоритм RSA.

Через деякий час після появи алгоритму, що відкидається, Меркла-Геллмана був створений перший зрілий алгоритм з відкритим ключем, який можна використовувати для шифрування та цифрових підписів, а саме алгоритм RSA. Його назва утворена від ініціалів його винахідників, а саме Рональда Рівеста, Аді Шаміра та Леонарда Адлемана. Він був створений у 1978 році [8]. Розробникам

удалося ефективно реалізувати ідею наявності секретної односторонньої функції. Надійність алгоритму RSA залежить від складності факторизації великих цілих чисел. Ключі включення та вимкнення є функцією двох дуже великих простих чисел. Відновлення вихідного тексту за допомогою зашифрованого тексту та відкритого ключа еквівалентно розкладанню числа на пару великих простих множників. Шифрування досягається шляхом модульного зведення ступінь великого цілого числа. Дешифровка досягається шляхом обчислення функції Ейлера за заданим великим числом, що вимагає знання розкладання числа на прості множники [8].

Стійкість алгоритму RSA залежить від складності розв'язання задачі факторизації, тобто факторизації великих чисел. Проте з технічного погляду це не так. Залежність безпеки RSA від проблеми факторизації великих чисел є гіпотетичним твердженням, оскільки теоретично можна винайти інший спосіб злому алгоритму RSA.

Суть алгоритму RSA ось у чому. Виберіть два великих простих числа p та q довільно. Обчисліть їх добуток та функцію Ейлера формулою (1.10):

$$\varphi(n) = (p - 1) \cdot (q - 1), \tag{1.10}$$

де $\varphi(n)$ – функція Ейлера;

p, q – прості числа.

Довільно вибираємо просте число e – ключ шифрування, що задовольняє умовам в формулах (1.11) та (1.12):

$$e < \varphi(n), \tag{1.11}$$

$$НСД\big(e, \varphi(n)\big) = 1. \tag{1.12}$$

Обчислюємо число d – ключ дешифрування, що є оберненим числом e формулою (1.13):

$$e \cdot d = 1 \cdot (\mathrm{mod}\ \varphi(n)), \tag{1.13}$$

де e – ключ шифрування;

d – ключ дешифрування.

Пара чисел (e,n) відіграє роль відкритого ключа, поміщеного в загальнодоступний каталог, а числа p та q зберігаються в секреті. Число d є ключовим. Щоб зашифрувати повідомлення, M спочатку розбивається на блоки чисел менше n, тобто p та q, якщо і - 128-бітові прості числа, то n містить не менше 256 біт, тому кожен блок m_i повідомлення повинен складатися приблизно з 256-бітних чисел. Блоки C_i, що становлять зашифроване повідомлення C, матимуть однакову довжину. Формула (1.14) для шифрування виглядає так:

$$C = M^e \cdot (\mathrm{mod}\ n), \tag{1.14}$$

де C – зашифроване повідомлення.

Інформація для дешифрування виходить шляхом виконання операції d-й ступеня по модулю над отриманим зашифрованим текстом C, тобто згідно формулі (1.15):

$$M = C^d \cdot (\mathrm{mod}\ n). \tag{1.15}$$

Тому, що згідно формулі (1.16) маємо:

$$C^d = (\mathrm{mod}\ n) = M^{e \cdot d} \cdot (\mathrm{mod}\ n) = \left(M^{k \cdot (p-1) \cdot (q-1)}\right) \cdot \mathrm{mod}(p \cdot q) = M, \tag{1.16}$$

де M – дешифрована інформація.

Переваги та недоліки алгоритму шифрування RSA.

Криптосистема RSA використовується в різних продуктах, платформах і галузях. В даний час криптосистема RSA вбудована в багато комерційних продуктів, і їх кількість продовжує зростати. Він також використовується операційними системами Microsoft, Apple, Sun та Novell. В апаратних версіях алгоритм RSA використовується у захищених телефонах, мережевих картах Ethernet, смарт-картах і широко застосовується у криптографічних пристроях. Крім того, цей алгоритм включений у всі основні протоколи, що використовуються для безпечного зв'язку в Інтернеті, включаючи S/MIME, SSL та S/WAN, і

використовується багатьма установами, такими як урядові установи, більшість корпорацій, урядові лабораторії та університети. .

Технологією шифрування RSA BESAFE користується приблизно п’ятсот мільйонів користувачів по всьому світу. Оскільки в більшості випадків використовується алгоритм RSA, його можна вважати найпоширенішою криптосистемою з відкритим ключем у світі, а з розвитком Інтернету це число має тенденцію до збільшення.

Основна перевага алгоритму RSA полягає в тому, що наявність відкритого ключа та знання алгоритму шифрування унеможливлює повторення зашифрованої інформації, програма шифрування PGP працює на основі алгоритму RSA, реалізована функція хешування (електронно-цифрового підпису).

Хоча через низьку швидкість шифрування (близько 30 кбіт/с з 512-бітним ключем на процесорі 2 ГГц) повідомлення зазвичай шифруються ефективнішим симетричним алгоритмом і випадковим ключем (сеансовим ключем), і тільки цей ключ використовує RSA. Через необхідність використання антикриптографічних генераторів випадкових чисел для генерації випадкових сеансових ключів для симетричного шифрування та алгоритмів симетричного шифрування, що ефективно протистоять атакам (нині широко використовуються AES, IDEA, Serpent, Twofish), такі механізми мають потенційні вразливості. Шифр RSA може бути зламаний протягом 5-10 років завдяки зростанню потужності процесорів та удосконаленню математичних алгоритмів знаходження простих множників.

3) Алгоритм Рабіна.

Алгоритм Рабіна було створено як модифікацію алгоритму RSA. Його безпека ґрунтується на складності знаходження квадратного кореня за модулем складового числа [8]. Суть алгоритму Рабіна ось у чому. Вибираються два великих простих числа p та q , які можна порівняти з три по модулю чотири. Ці прості числа виступають як закритий ключ, а їх добуток n – як відкритий ключ для формул (1.17) та (1.18):

$$p = 3 \cdot (\mathrm{mod}4) = -1 \cdot (\mathrm{mod}4), \quad (1.17)$$

$$q = 3 \cdot (\mathrm{mod}4) = -1 \cdot (\mathrm{mod}4), \quad (1.18)$$

бо знаємо, що n буде мати значення як у формулі (1.19):

$$n = p \cdot q, \tag{1.19}$$

де n – відкритий ключ.

Припустимо, що фіксовано і завжди дорівнює два, тоді шифрований текст повідомлення M обчислюється формулою (1.20):

$$C = M^2 \cdot (\mathrm{mod}\ n), \tag{1.20}$$

де C – розшифровка криптограми.

Розшифрування криптограми C. Введемо допоміжні значення x та y згідно формули (1.21) та формули (1.22):

$$x = C^1 \cdot (\mathrm{mod}\ p), \tag{1.21}$$

$$y = C^k \cdot (\mathrm{mod}\ q), \tag{1.22}$$

Тоді для x^2 та y^2 отримуємо в формулах (1.23) та (1.24):

$$x^2 = C^{2k} \cdot (\mathrm{mod}\ p) = \left[(M^2)^{\frac{p+1}{4}}\right]^2 \cdot \mathrm{mod}\ p = M^2 \cdot (\mathrm{mod}\ p), \tag{1.23}$$

$$y^2 = C^{21} \cdot (\mathrm{mod}\ q) = \left[(M^2)^{\frac{q+1}{4}}\right]^2 \cdot \mathrm{mod}\ q = M^2 \cdot (\mathrm{mod}\ q). \tag{1.24}$$

Звідки отримуємо чотири системи рівнянь для M_1, M_2, M_3, M_4 формулами (1.25), (1.26), (1.27), (1.28) далі:

$$\begin{cases} M_1 = x \cdot (\mathrm{mod}\ p) \\ M_1 = y \cdot (\mathrm{mod}\ q)' \end{cases} \tag{1.25}$$

$$\begin{cases} M_2 = x \cdot (\mathrm{mod}\ p) \\ M_2 = -y \cdot (\mathrm{mod}\ q)' \end{cases} \tag{1.26}$$

$$\begin{cases} M_3 = -x \cdot (\mathrm{mod}\ p) \\ M_3 = y \cdot (\mathrm{mod}\ q) \end{cases}, \tag{1.27}$$

$$\begin{cases} M_4 = -x \cdot (\mathrm{mod}\ p) \\ M_4 = -y \cdot (\mathrm{mod}\ q) \end{cases}. \tag{1.28}$$

Одним із результатів M_1, M_2, M_3 і M_4 буде повідомлення M. Якщо повідомлення написане словами, легко вибрати правильне M.

Однак, якщо повідомлення являє собою потік випадкових бітів, призначених для генерації ключів цифрового підпису, визначити правильну версію повідомлення M досить складно. Один із способів вирішити цю проблему – додати щось до заголовка повідомлення перед шифруванням.

Перевагами є висока швидкість роботи та проста і зрозуміла реалізація.

Недоліком є те, що можливо підібрати вхідні дані так, що кількість помилкових спрацьовувань буде неприпустимо великою [8].

4) Криптосистема Вільямса.

Х'ю Вільямс оптимізував алгоритм Рабіна, внісши до нього зміни, що усувають неоднозначність прийняття [8].

Суть алгоритму Вільямса ось у чому. Три простих числа p, q, c вибрані так, що згідно формул (1.29), (1.30) та (1.31) маємо:

$$p = 3 \cdot (\mathrm{mod}4), \tag{1.29}$$

$$q = 3 \cdot (\mathrm{mod}4), \tag{1.30}$$

$$c = 3 \cdot (\mathrm{mod}4). \tag{1.31}$$

Нехай P – непарне, більше одиниці число і його розкладання на прості множники (серед $p_1,\ldots,p_n$ може бути рівні). Тоді для довільного числа α символ Якобі визначається рівністю в формулі (1.32):

$$\left(\frac{\alpha}{P}\right) = \left(\frac{\alpha}{p_1}\right) \cdot \left(\frac{\alpha}{p_2}\right) \ldots \cdot \left(\frac{\alpha}{p_n}\right), \tag{1.32}$$

де $\left(\frac{\alpha}{p_n}\right)$ – символи Лежандра.

Також вибирається невелике ціле число s, для якого символ Якобі $J(s, n)$ дорівнює мінус 1. Числа s та n відправляються відправнику по відкритому каналу. Секретний ключ в формулі (1.33):

$$k = \frac{1}{2} \cdot \left[\frac{1}{4} \cdot (p - 1) \cdot (q - 1) \cdot (c - 1) + 1\right], \quad (1.33)$$

де k – секретний ключ.

Для шифрування повідомлення M обчислюється C_1 так, що визначаються проміжні повідомлення формулами (1.34) і (1.35):

$$M' = (S^{C_1} \cdot m) \cdot \mathrm{mod}\ n, \quad (1.34)$$

$$C_2 = M' \cdot (\mathrm{mod}\ 2). \quad (1.35)$$

Криптограма обчислюється так як і за алгоритмом Рабіна формулою (1.36):

$$C = (M')^2 \cdot \mathrm{mod}\ n. \quad (1.36)$$

Одержувач отримує три числа – C, C_1, C_2. Щоб розшифрувати C, одержувач обчислює M'' формулами (1.37) та (1.38):

$$\pm M'' = C^k \cdot (\mathrm{mod}\ n), \quad (1.37)$$

$$M' = (S^{-C_1} \cdot M'') \cdot \mathrm{mod}\ n, \quad (1.38)$$

де правильний знак M'' визначає C_2.

Перевагами алгоритму Вільямса є наступне.

1) Немає необхідності спочатку передавати ключ надійним каналом.

2) Тільки одна сторона знає ключ дешифрування, який повинен зберігатися в секреті (у симетричній криптографії такий ключ відомий обом сторонам і повинен зберігатися в секреті обома сторонами).

3) У великих мережах кількість ключів в асиметричній криптосистемі набагато менша, ніж у симетричній криптосистемі.

Недоліками алгоритму Вільямса є складність зміни алгоритму та більш довгі ключі.

Як висновок, слід зазначити, що, з урахуванням теоретичної частини, криптосистема Вільямса виглядає як найбільш стійка та надійна серед розглянутих вище криптосистем.

1.5 Теоретичний аналіз методів комбінованого шифрування

Безпека є основною частиною інформаційних систем, яку можна досягти за допомогою комбінованої криптографії. Однак основна мета криптографії полягає не тільки в тому, щоб забезпечити конфіденційність, але й у тому, щоб надати рішення для таких проблем, як: зламана цілісність, автентифікація та повнота даних. У криптографічних системах, ключовий термін відноситься до числового значення, яке використовується алгоритмом для зміни інформації, роблячи це так, що інформація була захищена та видима лише особам, які мають відповідний ключ для розкриття змісту. В наш час вчені продовжують пошук нових алгоритмів.

Однак це питання – дуже складна справа, тому що необхідно враховувати багато факторів, таких як: безпека, характеристики алгоритму, швидкість і складність.

Огляд криптографії з відкритим ключем не є повним без згадки про симетричний або секретний ключ. Причина цього полягає в тому, що криптографічні системи з відкритим ключем здебільшого не мають позитивних характеристик. Основними характеристиками симетричних криптографічних систем, є те, що той самий ключ використовується як для шифрування, так і для дешифрування. Через цю симетрію ключ потрібно зберегти таємний і ділиться лише між двома сторонами.

Поєднання симетричного та асиметричного шифрування може призвести до кращих результатів щодо шифрування.

У цьому процесі необхідно враховувати те, що один алгоритм слабший за інший. Комбінований метод шифрування дозволяє поєднати переваги високої секретності, що забезпечуються асиметричними криптосистемами з відкритим ключем, з перевагами високої швидкості роботи, наявними симетричними

криптосистемами з секретним ключем. При такому підході криптосистема з відкритим ключем застосовується для шифрування, передачі та подальшого розшифрування тільки секретного ключа симетричної криптосистеми. Симетрична криптосистема використовується для шифрування та передачі вихідного відкритого тексту. В результаті криптосистема з відкритим ключем не змінює симетричну криптосистему з секретним ключем, а лише доповнює її, дозволяючи підвищити всю захищеність переданої інформації. Наведемо далі алгоритм дії.

Наприклад, якщо користувач А1 захоче передати зашифроване комбіноване повідомлення М користувачеві А2, то він буде дотримуватись алгоритму дій зазначеного нижче.

1) Створити симетричний ключ, який називається в цьому методі сеансовим ключем К1.

2) Зашифрувати повідомлення М на сеансовому ключі К1.

3) Зашифрувати сеансовий ключ К1 на відкритому ключі К2 користувача А2 і на своєму секретному ключі К3.

4) Передати по відкритому каналу зв'язку на адресу користувача А2 зашифроване повідомлення разом із зашифрованим сеансовим ключем.

5) Дії ж користувача при отриманні зашифрованого повідомлення та зашифрованого сеансового ключа повинні бути зворотніми.

6) Розшифрувати на своєму секретному ключі К4 і відкритим ключем К2 користувача А1 сеансовий ключ К1.

7) За допомогою отриманого сеансового ключа К1 розшифрувати та прочитати повідомлення М.

При використанні комбінованого методу шифрування можна бути впевненим у тому, що тільки користувач А2 зможе правильно розшифрувати ключ К1 і прочитати повідомлення М.

Таким чином, при комбінованому методі шифрування застосовуються криптографічні ключі як симетричних, так і асиметричних криптосистем. Очевидно, вибір довжини ключів для кожного типу криптосистеми слід здійснювати таким чином, щоб злочиннику було дуже складно атакувати будь-який механізм захисту комбінованої криптосистеми.

Прикладом алгоритму для симетричного шифрування може бути всім відомий алгоритм DES.

DES – блоковий алгоритм, тобто при шифруванні вихідне повідомлення переводиться в двійковий код, а потім розбивається на блоки і кожен блок окремо зашифровується (розшифровується). За стандартом (прийнятий у 1977 році) розмір блоку DES дорівнює шістдесят чотири біти, тобто використовуючи восьми-бітове кодування ASCII, що застосовувалось в ті часи, отримаємо в одному блоці – вісім символів.

Тепер же в основному використовується шістнадцяти-бітове кодування Юнікоду (UTF-16).

Отже, для того щоб зашифрувати повідомлення алгоритмом DES, необхідно виконати наступну послідовність кроків.

1) Довести вихідне повідомлення до такого розміру (у бітах), щоб воно ділилося на розмір блоку (наприклад, сто двадцять вісім біт).
2) Поділити вихідне повідомлення на блоки.
3) Довести довжину ключа до половини блоку.
4) Перевести ключ у бінарний формат (у нулі та одиниці).
5) Провести над кожним блоком пряме перетворення мережею Фейстеля на протязі шістнадцяти раундів. Після кожного раунду необхідно виконувати циклічне зрушення ключа на задану кількість символів.
6) З'єднати всі блоки разом, у такий спосіб отримаємо повідомлення, зашифроване алгоритмом DES.

Розшифровка DES здійснюється аналогічно. Використовується зворотнє перетворення мережею Фейстеля [8].

Мережа Фейстеля використовується в алгоритмі DES для зашифрування (пряме перетворення мережею) та розшифрування (зворотнє перетворення). Ці перетворення зображені на рисунках 1.9 та 1.10 відповідно.

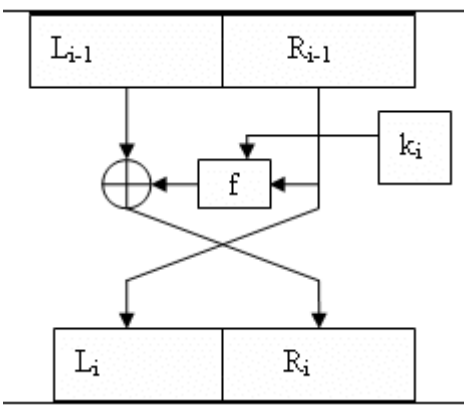


Рисунок 1.9 – Пряме перетворення мережею Фейстеля

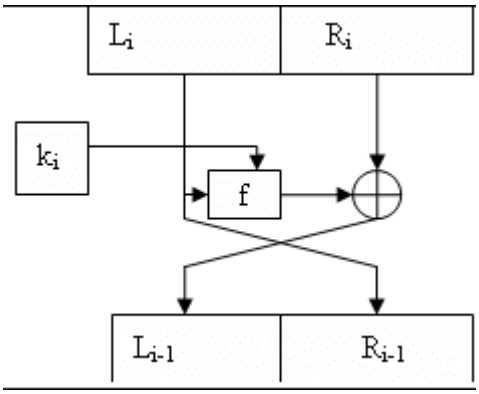


Рисунок 1.10 – Зворотнє перетворення мережею Фейстеля

Щоб було зрозуміліше, розглянемо один раунд прямого перетворення мережею Фейстеля.

На і-й ітерації вихідний блок ділиться навпіл – ліва частина позначається L, права R. Над R та ключем і обчислюється якась обрана логічна функція f (ми будемо використовувати XOR). Потім виконується обчислення логічної операції «виключає або» над L і раніше обчисленим значенням функції (L xor f). Старе значення R переноситься у ліву частину блоку, а праву частину заноситься значення L xor f. І остання операція раунду - потрібно виконати циклічне зрушення ключа: ключ і плюс один буде дорівнювати ключу і, котрий в свою чергу буде більше shiftKey (при розшифровці ключ і мінус один буде дорівнювати ключу і, котрий в свою чергу буде менше shiftKey), де shiftKey – кількість символів, на яку необхідно зрушити ключ.

Як бачимо алгоритм доволі простий і швидко буде виконуватись, хоча його криптостійкість доволі слаба і його треба застосовувати в парі з асиметричним шифруванням, наприклад шифруванням Вільямса на базі алгоритму RSA.

Зробити поетапне подвійне асиметричне шифрування також можливо, але за умови, що використовуються одні й ті самі методи шифрування і базуються на загальних взаємодоступних параметрах для шифрування, наприклад одному і тому самому симетричному ключі, або розподілі цього ключа на рівні частини.

Для остаточного і повного висновку, щодо вибору комбінації алгоритмів для шифрування, зробимо порівняльну табличку 1.1 нижче.

Таблиця 1.1 – Порівняння криптосистем та методів шифрування

Назва	Швидкість шифрування	Розмір ключа	Стійкість

алгоритму			
DES	290 кілобіт в секунду	128 біт	Слабка
RSA	30 кілобіт в секунду	2024 біт	Висока
AES	1750 кілобіт в секунду	256 біт	Слабка
Вільямса	30 кілобіт в секунду	4096 біт	Висока

Як бачимо, з урахуванням теоретичної частини, криптосистема Вільямса виглядає як найбільш стійка та надійна серед проаналізованих асиметричних методів, а метод AES, як кращий серед симетричних. Проте, зважаючи на більшу складність реалізації AES, та практично непомітну різницю у швидкості в запланованій інформаційній системі, було вирішено використовувати метод шифрування DES.

Отже, під час проектування інформаційної системи буду використовувати комбінацію з криптосистеми Вільямса та методу шифрування DES.

2 ВИБІР ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ РОЗРОБКИ

Мов програмування багато, тому вибір має базуватися на основі сфери, де вона буде використовуватися.

2.1 Вибір мови програмування

Серед мов програмування, які використовуються для реалізації криптосистем, розглянемо найпопулярніші мови, які відповідають тенденціям використання в розвитку криптографії: Python, C, C++, C#, Java, JavaScript.

Давайте розглянемо плюси і мінуси цих мов і порівняємо їх.

1) Мова Python.

Python – це високорівнева, універсальна, об’єктно-орієнтована, інтерпретована мова програмування [17]. Він призначений для підвищення продуктивності розробників і здатності читати код. Також можна помітити мінімалістичний синтаксис ядра Python, а стандартна бібліотека містить велику кількість корисних функцій.

Python підтримує велику кількість різноманітних парадигм програмування, зокрема: процедурне, функціональне, об’єктно-орієнтоване, імперативне та аспектно-орієнтоване програмування [17]. Ключові архітектурні особливості включають динамічний тип, автоматичне керування пам’яттю, повну інтроспекцію під час виконання, механізми обробки винятків, підтримку розширених структур даних і багатопотокове обчислення. Підтримка поділу програм на модулі та їх об'єднання в пакети. Крім того, є інтерактивний режим роботи.

Більшість сучасних платформ підтримують інтерпретатор CPython, який можна розширити за допомогою функцій і типів даних, розроблених на C, C++ або інших мовах, які можна викликати з C. Розповсюджується за безкоштовною ліцензією, Python можна використовувати та поширювати без обмежень у будь-якому додаток.

Мова Python розвивається дуже швидко. Версії з оновленими мовними функціями випускаються в середньому кожні два-три роки. Python є універсально стабільною мовою. Він широко використовується багатьма великими компаніями в проектах у різних сферах, як основна мова програмування та для створення

розширень та інтеграційних програм. В останні роки Python перевершив за популярністю інші найпопулярніші мови програмування, такі як C, C++ і Java.

Мова сильно змінилася з початку свого розвитку в 1980-х роках. Поточна версія, Python 3.0, була представлена в 2008 році і має більш структурований дизайн, ніж її попередник, уникаючи дублювання модулів і конструкцій. На даний момент останньою версією є Python 3.8.

Мова Python має нижчевказані переваги.

1) Велика кількість корисних модулів забезпечують можливість роботи з мережевими протоколами, серверами та клієнтами, кросплатформними додатками, графічними інтерфейсами та інструментами для роботи з математичними функціями.

2) Портативність – мова доступна майже на всіх відомих платформах, таких як Microsoft Windows, усі варіанти UNIX, Mac OS, iPhone OS та Android, а також чудово сумісний із системами SCM [19].

3) Продуктивність – інтеграція процесів, покращені можливості управління та системи модульного тестування допомагають підвищити швидкість і продуктивність більшості програм.

4) Простота інтеграції – мова має потужні елементи керування, наприклад, виклик функцій безпосередньо з C, C++ або Java через Python. Python також підтримує XML та інші мови розмітки. Додатковим плюсом є те що Python широко використовується в сучасній технології WPF [20].

5) Простий синтаксис.

6) Допускаються цілі числа довільного розміру.

Недоліки мови Python:

1) Оскільки Python є інтерпретованою мовою, швидкість виконання програм, написаних на ньому, низька порівняно з неінтерпретованими мовами.

2) Відсутність статичної типізації не дозволяє реалізувати механізм перевантаження функцій на етапі компіляції.

3) Помилки не виявляються під час компіляції, а лише під час виконання, що ускладнює тестування програм.

2) Мова C.

Мова C – універсальна, компільована, процедурна, статично типізована імперативна мова програмування [17]. Це одна з найпопулярніших мов

програмування за кількістю написаних програм, хоча спочатку вона була призначена тільки для написання системного програмного забезпечення.

Компілятори мови С існують багатьох операційних систем. Вони дуже прості та забезпечують спрямованість мови до мінімалізму. Мова С забезпечує низькорівневий доступ до оперативної пам'яті без особливої динамічної підтримки.

Багато мов програмування високого рівня мають компілятори, інтерпретатори та бібліотеки, написані мовою С. Крім того, деякі з цих мов використовують мову С як проміжну мову.

Як основний засіб розширення мова використовує бібліотеки, що являють собою набори функцій.

Серед переваг мови С слід зазначити наступні: швидкість виконання та реалізація на широкому спектрі апаратних платформ.

Недоліки цієї мови:

1) Складність мови, яка сприяє написанню складного, проблематичного та заплутаного коду.

2) Різні припущення в мові дозволяють компілювати програми з великою кількістю невиявлених помилок, що призводять до непередбачуваної поведінки програми.

3) Необроблена модульна підтримка.

4) Відсутність контролю над адресною арифметикою, контролем над змінними та ініціалізацією динамічної пам'яті призводить до крихкості програми.

5) Відсутність можливості вирішувати багато прикладних задач.

Далі одразу розглянемо мову програмування С++.

3) Мова С++.

Мова С++ – статично типізована компільована мова програмування загального призначення [17]. Підтримує кілька парадигм програмування, таких як процедурне, об'єктно-орієнтоване та універсальне програмування.

Мова використовується для системного програмування, написання драйверів, серверних та клієнтських програм, відеоігор. Мова синтаксично схожа мовою С. Мова С++ є розширеною версією мови С у багатьох відношеннях, включаючи додавання об'єктно-орієнтованих можливостей за рахунок введення класів, що забезпечують інкапсуляцію, успадкування та поліморфізм.

Стандартна бібліотека мови С++ заснована на стандартній бібліотеці шаблонів, яка надає такі важливі інструменти, як контейнери та ітератори.

Переваги мови C++.

1) Програми мовою C++ практично не поступаються програмам, написаним мовою C.

2) Мова підходить для розробки різних систем та платформ.

3) Адреси, порти та пам'ять можуть оброблятися на низькому рівні.

4) Підтримує різні стилі програмування.

Серед недоліків мови C++ важливо відзначити нижчевказані пункти.

1) Зменшує безпеку програми та призводить до помилок, багів та можливості непередбачуваної поведінки програми (аналогічно мові C).

2) Необроблена модульна підтримка.

3) Функціональне програмування не підтримується.

Недоліки C++ частково успадковані від мови-пращура C та викликані вихідною вимогою максимально можливої сумісності з C. Приклади цих недоліків:

Препроцесор, успадкований від C, дуже примітивний.

Погана підтримка модульності (насправді у класичному C немає модульності на рівні мови, її забезпечення перекладається на компонувальника). При включенні великої кількості модулів, включаючи інтерфейси зовнішніх модулів, вставка препроцесором в заголовні файли (#include) може серйозно уповільнити компіляцію (оскільки результуючий файл, що обробляється компілятором, дуже великий).

4) Мова C#.

Мова C# – це об'єктно-орієнтована мова програмування для платформи Microsoft .NET Framework із безпечною системою типів та C-подібним синтаксисом [17]. Мова підтримує строгу статичну типізацію, навантаження операторів, поліморфізм, делегування, властивості, події, анонімні функції, ітератори. Мова C# успадкувала багато функцій від своїх попередників і не підтримує деякі моделі, які виявилися проблематичними.

Сьогодні мова програмування C# є однією з найпотужніших і широко затребуваних мов, і вона швидко розвивається. C# був створений як компонентна мова програмування, призначена для повторного використання компонентів, що ви створюєте.

Серед переваг мови C# варто зазначити наступне.

1) C# – потужна об'єктно-орієнтована мова з можливістю узагальнення та успадкування.

2) C# створюється паралельно з .NET Framework і повністю враховує всі можливості.

3) C# містить найкращі можливості мов C/C++.

4) З .NET Framework, створеної поверх операційної системи, програмісти на C# можуть отримати ті ж переваги віртуальної машини, що й програмісти на Java, не кажучи вже про те, що середовище виконання CLR є компілятором проміжної мови, а віртуальна машина Java – пристроєм інтерпретатора байт-коду.

5) Потужна бібліотека фреймворку підтримує зручну побудову різних типів додатків мовою C#, дозволяючи легко створювати веб-сервіси та інші типи компонентів, достатньо простого зберігання та вилучення інформації з баз даних та інших сховищ даних.

6) Поєднання реалізації побудови надійного та ефективного коду є дуже важливим фактором успіху C#.

Недоліки у C# такі: пріоритетна орієнтованість на платформу Windows; мова безкоштовна тільки для невеликих фірм, індивідуальних програмістів, стартапів та учнів. Великій компанії купівля ліцензійної версії цієї мови обійдеться в велику ціну.

5) Мова Java.

Мова Java – строго типізована об'єктно-орієнтована мова програмування [17]. Програми, написані цією мовою, перетворюються на байт-код, який виконується віртуальною машиною Java – програмою, яка обробляє байт-код, передає інструкції пристрою та діє як інтерпретатор.

Java майже повністю запозичує синтаксис C та C++, а також об'єктну модель C++. Однак це унеможливлює виникнення конфліктних ситуацій і знижує відповідальність програміста за правильне функціонування програми. Через те, що основна увага приділяється незалежності від платформи, Java має менше низькорівневих функцій, більш повільне виконання та підвищене використання пам'яті порівняно з C та C++.

Переваги мови Java: архітектурна незалежність та переносимість; захист від низькорівневих помилок; висока продуктивність; автоматичне керування пам'яттю.

Недоліки мови: низька швидкість виконання програми та широке використання ресурсів пам'яті.

6) Мова JavaScript.

JavaScript – це прототипна динамічна об'єктно-орієнтована мова програмування [17]. Він підтримує об'єктно-орієнтований, функціональний та імперативний стилі програмування. JavaScript найчастіше використовується для сценаріїв веб-сторінок, але ця мова також підходить для написання веб-додатків, серверних, настільних і мобільних додатків та сумісна з фреймворком React [18].

Його архітектурні особливості включають автоматичне керування пам'яттю, слабку динамічну типізацію, прототипне програмування та функції як першокласні об'єкти. Існує великий вибір бібліотек, які роблять використання мови ще простіше.

Переваги мови JavaScript: високо абстрактний; кросбраузерність; автоматичний набір тексту та складання сміття; обробка винятків; спрощений синтаксис.

Недоліки мови: програми працюють повільно, тому що мова є інтерпретатором; складність введення в експлуатацію; запуск програм у браузері.

Отже, порівняємо можливості перелічених вище мов та їх придатність для написання криптографічних систем. Python, JavaScript, Java та C# мають захист від програмних порушень низького рівня. Перші два мають спрощений синтаксис у порівнянні з C-подібними мовами, але страждають від зниження швидкості виконання. C# наділений найбільш підходящими і в той же час найпростішими функціями для роботи з математичними функціями та великими числами, які необхідні для створення криптосистем.

На підставі проаналізованих даних було прийнято рішення використовувати мову C#, оскільки вона має найзручніші інструменти для швидкої та якісної розробки криптосистем.

2.2 Вибір середовища розробки

Оскільки C# одна із найпоширеніших мов програмування, він має багато інтегрованих середовищ розробки з різними функціями. Бо доволі висока продуктивність є однією з основних вимог щодо розробки програм.

Важливим критерієм вибору середовища розробки програми є аналізування. Для дослідження було обрано інтегровані середовища розробки MS Visual Studio, JetBrains Clion, Qt Creator, NetBeans та Eclipse SDK, оскільки вони є найбільш потужними та широко використовуються.

1) Microsoft Visual Studio.

Microsoft Visual Studio – це інтегроване середовище, призначене для розробки програмного забезпечення з безліччю додаткових інструментів.

Продукти MS Visual Studio дозволяють розробляти консольні програми, програми з графічним інтерфейсом, веб-додатки для платформ. Інтерфейс середовища наведено на рисунку 2.1.

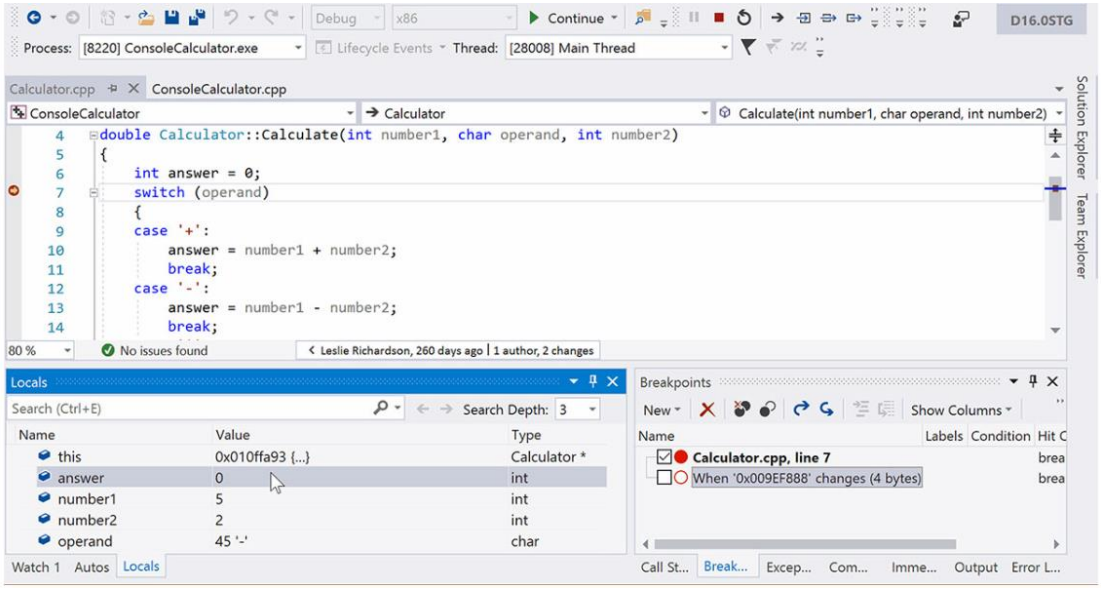


Рисунок 2.1 – Інтерфейс Microsoft Visual Studio

MS Visual Studio включає вбудований інструмент для редагування вихідного коду програми та рефакторингу коду. Вбудований налаштовувач MS Visual Studio працює на двох рівнях: рівень налагодження вихідного коду та рівень налагодження машинного коду. Інші вбудовані інструменти включають форми з графічним інтерфейсом та зручні редактори для створення програм з інтерфейсами, включаючи веб-редактор та редактор класів. Це середовище розробки підтримує можливість створення та підключення плагінів (надбудов), що розширюють функціональні можливості. MS Visual Studio підтримує системи контролю версій вихідного коду (такі як Visual SourceSafe або Subversion) та включає безліч інструментів, що дозволяють редагувати та проектувати код з використанням об'єктно-орієнтованих мов програмування.

2) JetBrains Clion.

JetBrains CLion – інтелектуальне інтегроване середовище розробки програмного забезпечення, призначене для розробки додатків мовами C і C ++ [17].

JetBrains Clion підтримує розробку програмного забезпечення на платформах Windows, OS X та Linux. Інтерфейс середовища наведено на рисунку 2.2.

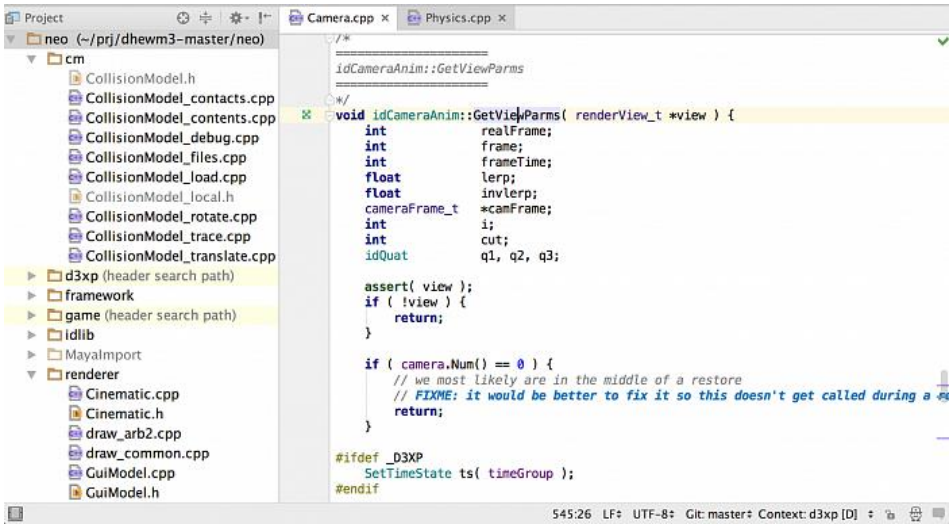


Рисунок 2.2 – Інтерфейс JetBrains CLion

Багатофункціональне середовище розробки — JetBrains CLion включає вбудований відладчик, безліч шаблонів коду та інтерфейси для популярних систем керування версіями, таких як Subversion, Git, GitHub, Mercurial, CVS, Perforce та TFS. CLion забезпечує автоматичне завершення та автоматичне форматування коду, аналіз коду стрічок для виявлення потенційних проблем та пропозиції способів їх усунення, систему складання, що підтримує кросплатформові проекти CMake, а також різні рефакторинги коду.

3) Qt Creator.

Qt Creator – це повністю інтегроване середовище розробки програмного забезпечення. Інтерфейс середовища наведено на рисунку 2.3.

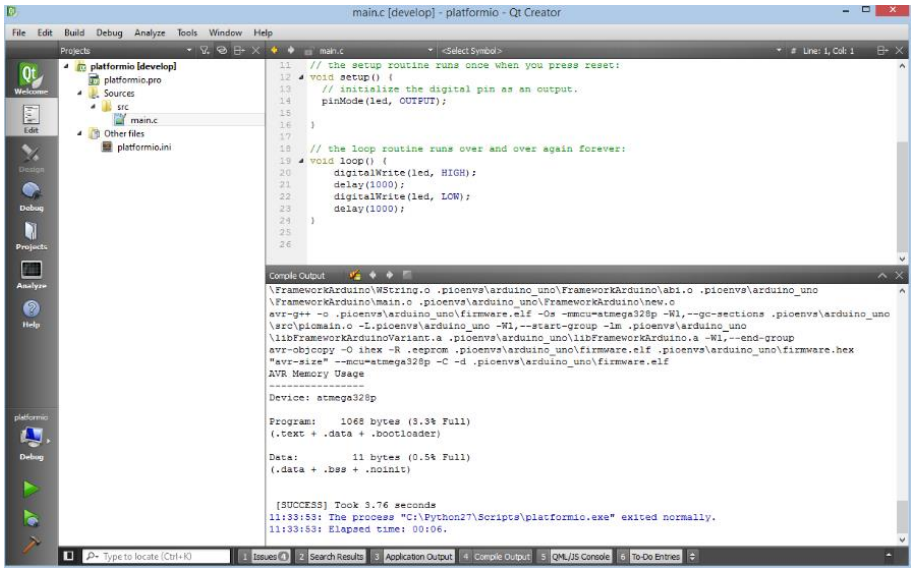


Рисунок 2.3 – Інтерфейс Qt Creator

Qt Creator призначено для розробки програмного забезпечення з використанням мови програмування C#. Також є PyQt для програмування на Python та PHP-Qt для PHP.

Qt Creator інтегрується з кросплатформовими системами автоматизації складання: qmake і CMake, включаючи редактор коду Qt Designer, який дозволяє проектувати і створювати графічні інтерфейси користувача (GUI) з віджетів Qt. Середовище включає безліч корисних інструментів, таких як підтримка систем контролю версій (Subversion, Git, GitHub, Mercurial та CVS) та Qt. Qt Creator включає вбудований налагоджувач для налагодження звичайних програм C#, а також включає можливість підключення мобільного пристрою до комп'ютера та налагодження програм, що працюють на ньому.

4) NetBeans.

NetBeans – це безкоштовне середовище з відкритим вихідним кодом для розробників програмного забезпечення [10]. NetBeans надає безліч різних інструментів для створення професійного програмного забезпечення з використанням Java, C/C++, PHP, Python, JavaScript, Groovy, Ruby та інших. Інтерфейс середовища наведено на рисунку 2.4.

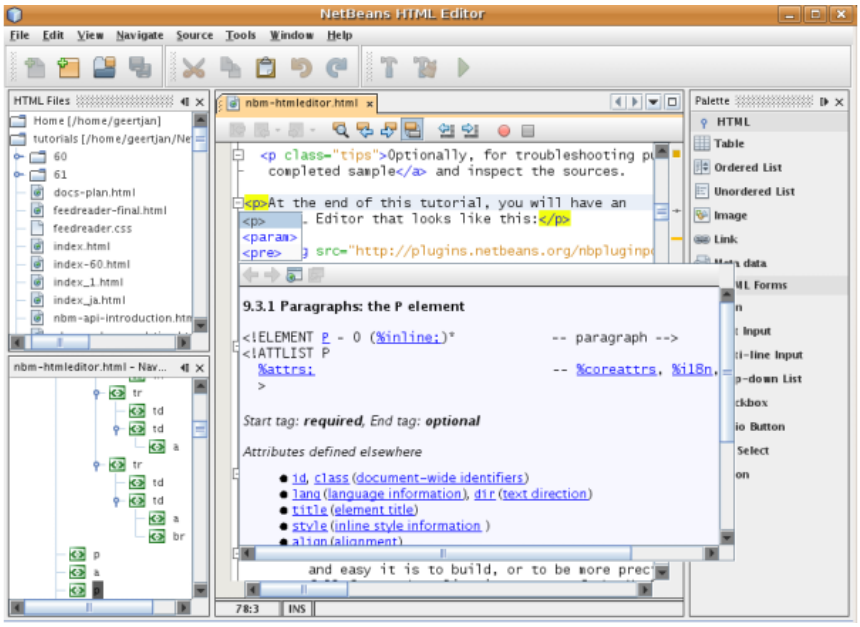


Рисунок 2.4 – Інтерфейс середовища NetBeans

Середовище розробки NetBeans включає такі функції: рефакторинг коду; профільування; виділення різних синтаксичних конструкцій за кольором; автозаповнення у під'їздах різних будівель; безліч шаблонів коду.

У середовищі NetBeans спочатку має бути встановлена необхідна версія J2EE SDK або Sun JDK.

NetBeans надає готові інструменти керування версіями для Subversion, Mercurial та Git. Редактор середовища IDE та можливості перетягування також дозволяють швидко та ефективно розробляти графічні інтерфейси додатків.

5) Комплект розробки Eclipse.

Eclipse – це безкоштовне інтегроване середовище розробки програмного забезпечення з відкритим кодом. Eclipse має велику і велику систему модулів (надбудов), що підключаються, які забезпечують робоче середовище з використанням таких мов програмування, як C, C++, PHP, Perl, Python, Ruby, Ada або COBOL. Інтерфейс середовища наведено на рисунку 2.5.

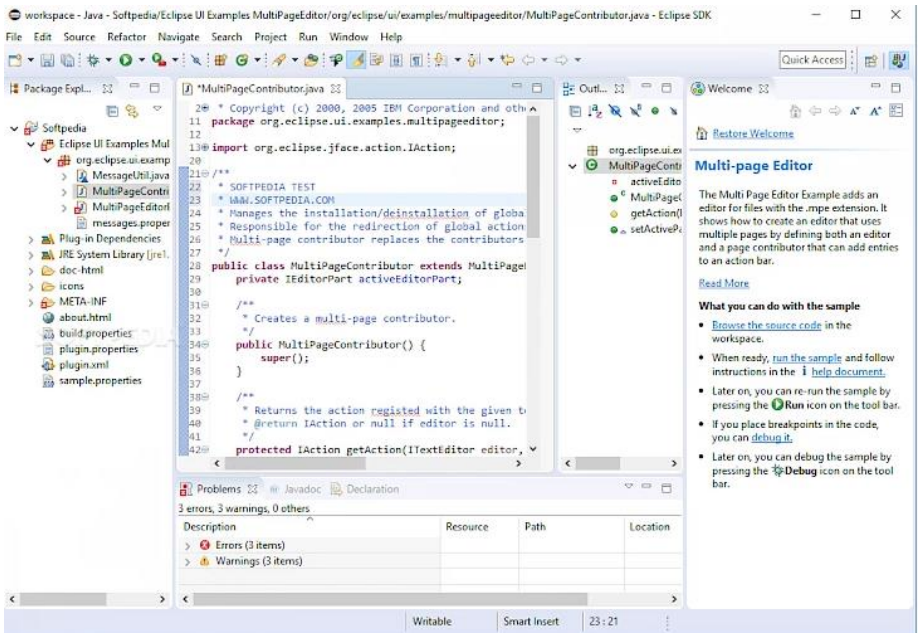


Рисунок 2.5 – Інтерфейс Eclipse

Eclipse дозволяє створювати кросплатформене програмне забезпечення для Microsoft Windows, Mac OS X, дистрибутивів Linux і навіть для Solaris.

Така система допомагає розробникам створювати потужні програми користувача з красивим графічним інтерфейсом на основі CSS (каскадних таблиць стилів). Інтегроване середовище розвитку.

Eclipse надає середовищем розробки Java всі інструменти, необхідні для розробки високоякісних програм.

У результаті вивчення розглянутих середовищ розробки програмування мовою C# було створено порівняльну таблицю 2.1.

Для розробки програми використовується середовище Microsoft Visual Studio. Середовище розробки Microsoft Visual Studio було обрано через безліч важливих функцій розробки, таких як підтримка аналізу коду C#, налагодження та сумісність із застарілими проектами.

Для цього проекту на розгляд було винесено реляційну СУБД. За способом розподіленості була обрана локальна СУБД, а за способом доступу – клієнт-серверна. Далі розглянемо системи управління базами даних, що існують на даний момент.

Кожна система підтримує різні моделі та структури баз даних. Ця модель і визначає, як СУБД оперуватиме даними. Найпопулярнішою моделлю бази даних є реляційна модель.

Таблиця 2.1 – Порівняння середовищ розробки

	MS Visual Studio	JetBrains Clion	Qt Creator	Eclipse	NetBeans
Редактор інтерфейсу користувача	+	–	+	+	+
Вбудований відладчик		+	+	+	+
Підтримка C#	+	+	+	+	+
Встановлене ПЗ та наявність ліцензії	+	–	+	–	–
Опис використання	+	–	+	+	–
Сумісність із попередніми проектами	+	–	+	–	–

Реляційна модель має математичний спосіб структурування, зберігання та використання даних. Завдяки десятиліттям розробки, реляційні СУБД досягли високого рівня відмовостійкості та продуктивності. Незважаючи на великі обмеження у формуванні та управлінні даними, реляційні бази даних зберігають широкі можливості з налаштування та надають великий функціонал. Найбільш популярні реляційні СУБД на сьогоднішній день: SQLite - потужна СУБД, що вбудовується; MySQL – найпопулярніша СУБД; PostgreSQL – професійна СУБД, що вільно-розповсюджується [11].

Розглянемо докладніше кожну з наведених вище СУБД.

1) SQLite.

SQLite – файлова, що легко вбудовується БД, яка має потужну бібліотеку і великий набір інструментів для простої обробки даних різних видів [14].

Переваги наступні: файлова – вся БД складається з одного файлу; швидка та потужна – звернення відбуваються безпосередньо до файлів; стандартизована – підтримка SQL; підходить для розробки та тестування.

Недоліками є: відсутність можливості у користувача управління та відсутність можливості збільшення продуктивності.

2) MySQL.

MySQL – найпопулярніша і найпростіша у використанні з усіх серверних БД. Має широкий функціонал і у вільному доступі [16].

Перевагами є: простота – наявність сторонніх інструментів, що полегшують початок роботи з MySQL; має широкий функціонал, незважаючи на те, що не повністю відповідає SQL-стандартам; наявність вбудованих функцій забезпечують безпечну роботу; швидкість та масштабованість - MySQL досить легко працює з великими обсягами даних та легко масштабується.

Недоліками є: надійність, деякі операції реалізовані менш надійно, ніж в інших СУБД та застій у подальшій розробці.

3) PostgreSQL.

PostgreSQL – професійна СУБД, яка орієнтується на відповідність стандартам та розширюваності [15].

Перевагами є: можливість програмного розширення за рахунок збережених процедур; об'єктно-орієнтованість – PostgreSQL є не тільки реляційною БД, а й об'єктно-орієнтованою; є повністю SQL-сумісною БД.

Недоліками є: продуктивність – в операціях читання може поступатися іншим СУБД та має низьку популярність через свою складність.

При розгляді різних СУБД, які підходять для досягнення поставлених завдань, було детально розглянуто рішення, що стосується реляційної моделі. Як продукт реляційної моделі була обрана СУБД MySQL, завдяки хорошій системі безпеки пакету, стабільній роботі та високій швидкодії [10].

3 ПРОЕКТУВАННЯ І РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Розглянемо алгоритм Вільямса (криптосистема Вільямса), що лежить в основі вдосконаленого методу, розробленого у свій час на базі RSA [11].

3.1 Модель шифрування бази даних

Для будь-якої системи бажано мати доказ того, що її злом має обчислювальну складність, подібну до складності завдання, яку в даний момент не можна вирішити за заданий час. Як і RSA, криптосистема Вільямса спирається на припущення складності факторизації великих чисел. Показано, що з будь-якому зламу шифротекста з допомогою криптоаналізу, треба лише відкритий ключ розкласти на множники. Тобто, розв'язати рівняння щодо p, q, c в формулі (3.1):

$$p \cdot q \cdot c = n. \tag{3.1}$$

Обчислювальна складність завдання факторизації невідома, але вважається досить високою.

Основний принцип цього методу полягає у зведенні у ступінь числа виду як у формулі (3.2) нижче:

$$\alpha = a + b \cdot \sqrt{c}, \tag{3.2}$$

де a, b, c — цілі числа, а чи не просто цілі числа, як у алгоритмі RSA.

Визначимо функцію формулами (3.3) та (3.4):

$$X_i(\alpha) = \frac{(\alpha^i + \alpha^{-i})}{2}, \tag{3.3}$$

$$Y_i(\alpha) = \frac{b \cdot (\alpha^i + \alpha^{-i})}{(\alpha + \bar{\alpha})}, \tag{3.4}$$

де α – сполучене альфа-число, що дорівнює формулі (3.5):

$$a - b \cdot \sqrt{c}. \tag{3.5}$$

Тоді функція, що дозволяє висловити, у скільки разів число α матиме такий вигляд згідно формули (3.6):

$$\alpha^i = X_i(\alpha) + \sqrt{c} \cdot Y_i(\alpha). \tag{3.6}$$

Математична суть методу полягає так: нехай n – добуток двох простих непарних чисел p і q, a, b, c – цілі числа, що задовольняють співвідношенню з формули (3.7):

$$a^2 - c \cdot b^2 = 1 \cdot (\mathrm{mod}\ n). \tag{3.7}$$

Нехай символи Лежандра в формулах (3.8) та (3.9) нижче:

$$\delta_p = \left(\frac{c}{p}\right), \tag{3.8}$$

$$\delta_q = \left(\frac{c}{q}\right), \tag{3.9}$$

задовольняють рівнянням у формулах (3.10) та (3.11):

$$\delta_p = -p \cdot (\mathrm{mod}\ 4), \tag{3.10}$$

$$\delta_q = -q \cdot (\mathrm{mod}\ 4). \tag{3.11}$$

Крім того, з формули (3.12) для НСД маємо:

$$\mathrm{НСД}(c \cdot b, n) = 1. \tag{3.12}$$

І для знаку Якобі з формули (3.13) маємо:

$$\left(\frac{2 \cdot (a+1)}{n}\right) = 1. \tag{3.13}$$

Тепер позначимо в формулі (3.14):

$$m = \frac{(p-\delta\cdot p)\cdot(q-\delta\cdot q)\cdot(c-\delta\cdot c)}{4}. \quad (3.14)$$

Беремо e та d, що відповідають рівнянню у формулі (3.15):

$$e \cdot d = \frac{(m+1)}{2\cdot(\mathrm{mod}\ 2)}. \quad (3.15)$$

Тоді, при цьому припущенні, згідно із формулою (3.16) маємо:

$$a^{2\cdot e\cdot d} = \pm a \cdot (\mathrm{mod}\ n). \quad (3.16)$$

Використовуючи цю лему, генеруються ключі шифрування.

Спочатку вибираються три простих непарних числа p, q, c обчислюється їх добуток n. На вибір число c вибирається таким, щоб символи Лежандра δ_p і δ_q задовольняли умовам, сформульованим у лемі. Число s також вибирається на вибір таким чином, щоб знак Якобі і НСД були як у формулі (3.17) та формулі (3.18):

$$\left(\frac{s^2-c}{n}\right) = -1, \quad (3.17)$$

$$\mathrm{НСД}(s, n) = 1. \quad (3.18)$$

Далі обчислюємо число m за формулою (3.19):

$$m = \frac{(p-\delta_p)\cdot(q-\delta_q)\cdot(c-\delta_c)}{4}. \quad (3.19)$$

Число d вибираємо в такий спосіб, де НСД за формулою (3.20):

$$\mathrm{НСД}(d, m) = 1. \quad (3.20)$$

Розраховуємо e за формулою (3.21):

$$e = \frac{m+1}{2} \cdot d^{-1} \cdot (\mathrm{mod}\ m). \quad (3.21)$$

Отримуємо ключі n, e, c, s - відкриті, p, q, m, d - закриті.

Отже, шифрування відбувається в такий спосіб. Повідомлення задаються як число M, і воно лежить в інтервалі від одного до n. Наші γ і b_1 обчислюються, якщо знак Якобі буде згідно формулі (3.22):

$$\left(\frac{M^2-c}{n}\right) = 1, \quad (3.22)$$

то b_1 дорівнюватиме нулю, і наш γ буде як у формулі (3.23):

$$\gamma = M + \sqrt{c}. \quad (3.23)$$

Інакше b_1 дорівнюватиме одиниці, і наш γ буде як у формулі (3.24):

$$\gamma = (M + \sqrt{c}) \cdot (s + \sqrt{c}). \quad (3.24)$$

Число α виходить за формулою (3.25):

$$\alpha = \frac{\gamma}{\bar{\gamma}}, \quad (3.25)$$

де $\bar{\gamma}$ буде за формулою (3.26):

$$\bar{\gamma} = M - \sqrt{c}. \quad (3.26)$$

Далі обчислюємо за формулами (3.27) та (3.28):

$$X_e(\alpha) = \frac{(\alpha^e + \bar{\alpha}^e)}{2}, \quad (3.27)$$

$$Y_e(\alpha) = \frac{(\alpha^e - \overline{\alpha^e})}{(\alpha^e + \bar{\alpha})}. \quad (3.28)$$

З них отримуємо формулу (3.29):

$$E = \frac{X_e(\alpha)}{Y_e(\alpha)}. \quad (3.29)$$

Тоді b_2 розраховується за формулою (3.30):

$$b_2 = \alpha \cdot \mathrm{mod}\ 2. \quad (3.30)$$

Отже, три числа (E, b_1, b_2) надсилаються у вигляді зашифрованого тексту.

Дешифровка відбувається в такий спосіб. Спочатку обчислюється параметр α за формулою (3.31):

$$\alpha^{2\cdot e} = \frac{E+\sqrt{c}}{E-\sqrt{c}}. \quad (3.31)$$

Далі визначаються параметри X, Y, E, що позначаються формулою (3.32):

$$\alpha^{2ed} = X_d(\alpha^{2e}) + Y_d(\alpha^{2e}) \cdot \sqrt{c} = \pm\alpha'. \quad (3.32)$$

Число b_2 показує, який символ вибрано, а число b_1 - чи слід множити результат на відповідь з формули (3.33):

$$\frac{s-\sqrt{c}}{s+\sqrt{c}}. \quad (3.33)$$

В результаті всього ми отримаємо число α'. Вихідний текст повідомлення M можна отримати за формулою (3.34):

$$M = \frac{\alpha'+1}{\alpha'-1} \cdot \sqrt{c}. \quad (3.34)$$

Структурну схему класичного методу шифрування Вільямса представлено на рисунку 3.1

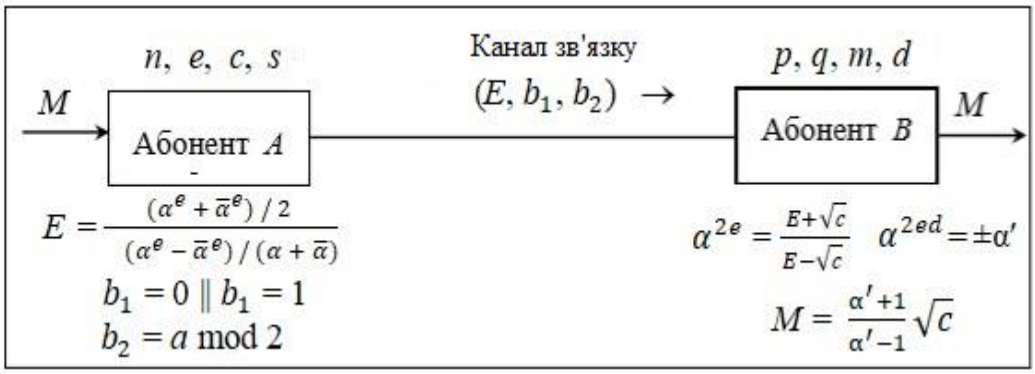


Рисунок 3.1 – Блок-схема класичного методу шифрування Вільямса

Після генерації ключа основне обчислення відбувається при множенні числа α до ступеня, що відбувається за кілька операцій множення за модулем, число яке обчислюється за такою формулою (3.35):

$$O \cdot (\log n), \tag{3.35}$$

де O – операція.

Кожне обчислення відбувається за кілька операцій складання з формули (3.35) тощо. Операції збільшуються з однаковою швидкістю. Тобто вся операція відбувається згідно формулі (3.36):

$$O \cdot (\log^3 n). \tag{3.36}$$

Що, як остаточний результат, дорівнює часу зведення в ступінь цілого числа за модулем, що використовується в алгоритмі шифрування RSA.

3.2 Аналіз варіантів використання системи

Пропонований метод шифрування Вільямса реалізований у вигляді окремого модуля, що містить функції для реалізації необхідних алгоритмів. Також реалізовані модулі генерації великих простих чисел, модулі математичних алгоритмів, що використовуються в процесі шифрування, та модулі прямого перетворення чисел на байти.

Згенеровані числа використовують у функції генерації ключа шифрування.

Щоб вибрати випадкове число, замість використання алгоритму генерації випадкових чисел вибираємо використання заданого початкового значення числа та перевіряємо, чи воно відповідає критеріям, поступово збільшуючи його. Такий підхід значно підвищує швидкість роботи функції.

Функція приймає довжину ключа як аргумент. Кожне із трьох простих чисел становить одну третину довжини ключа. Тому загальний обсяг даних та довжина ключа, використовуваного для обчислень, не збільшуються по довжині порівняно з класичними методами шифрування, зберігаючи при цьому підвищену криптостійкість [11].

Алгоритми використовуються для створення ключів, які дозволяють обчислювати математичні функції, такі як символ Лежандра та його узагальнення, символ Якобі. Це мультиплікативні функції, які часто використовують у теорії чисел. Ці функції реалізовані окремих модулях разом з іншими математичними операціями. Після генерації ключ передається методам шифрування.

Систему можна використовувати для шифрування різних текстових даних, використовуючи наявну базу даних, або ж заповнюючи базу даних в режимі реального часу. Втім, слід враховувати, що збільшення полів у таблицях нашої бази даних впливає і на клієнтську частину системи, тож розширення таблиць буде потрібне і на боці клієнта в його додатку. Поточна інформаційна система добре зможе підійти для шифрування даних пов’язаних із поточним станом оцінок студентів. В майбутньому система буде дороблена з додаванням дешифратора, автоматичною генерацією ключів, а також сучасним графічним інтерфейсом. Все ж таки, в цій роботі інформаційна система представлена більше для аналітично-наукового прикладу на основі аналізу різних методів шифрування баз даних.

3.3 Проектування внутрішньої будови

Для реалізації веб-додатка була обрана стандартна клієнт-серверна архітектура, що має три чітко розділені рівні: рівень подання даних (інтерфейс користувача), серверна частина, що забезпечує логічну обробку подій, і база даних як рівень управління даними [13]. Ця архітектура була обрана з урахуванням не лише вимог до швидкості роботи програми на пристроях користувачів, але й з урахуванням забезпечення розподілу рівнів доступу до інформації. Також варто відзначити, що було обрано клієнт-серверну архітектуру з товстим сервером і

тонким клієнтом. Такий розподіл логіки обробки інформації обумовлений вимогою підтримки сайту будь-яким браузером і з будь-якого пристрою, тому мінімізується використання ресурсів пристроїв користувачів. Крім того, ця архітектура забезпечує віддалене зберігання даних, незалежно від сеансу користувача або роботи сервера. Схематично архітектура програми представлена рисунку 3.2.

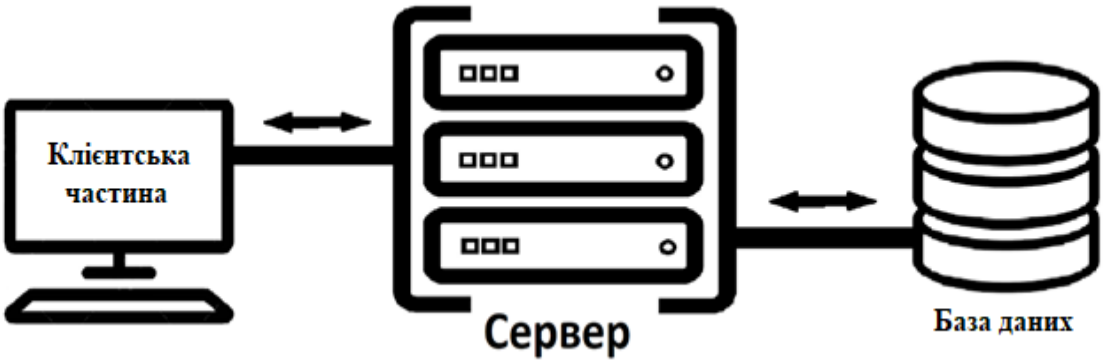


Рисунок 3.2 – Схема архітектури програми

У системі керування контентом застосовується СУБД MySQL. База даних складається із схеми shifr.

Усі дані зберігаються на локальному сервері в окремих базах даних, як і зазначено на рисунку 3.3.

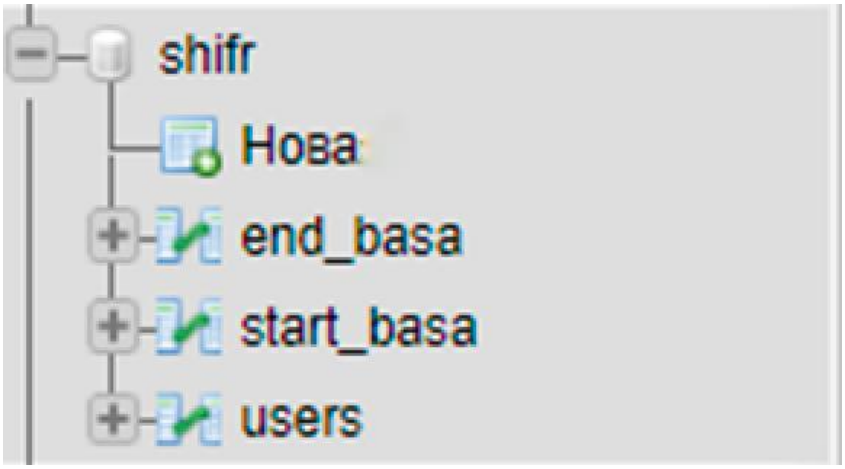


Рисунок 3.3 – Структура бази даних shifr

Опис кожної структури БД, які були створені для застосування наведено на рисунку 3.4, рисунку 3.5, рисунку 3.6.

+ Параметр

id	login	pass
1	admin	admin
2	user1	1234

Рисунок 3.4 – Структура бази даних users

+ Параметр

id	name	sername	mark
1	name1	sername1	100
2	name2	sername2	60
3	name3	sername3	80

Рисунок 3.5 – Структура бази даних start_basa

+ Параметр

id	name	sername	mark
1	59534648101	38494959534648101	100100115
2	59534648102	38494959534648102	99100
3	59534648103	38494959534648103	109100

Рисунок 3.6 – Структура бази даних end_basa

Для підключення до бази даних було використано наступний код програми.

Створення класу database (DB) для підключення MySQL бази даних на рис. 3.7 далі.

```
class DB
{
```

Рисунок 3.7 – Клас database для підключення MySQL бази даних

Прописання шляху для підключення MySQL бази даних на рисунку 3.8.

```
MySqlConnection connection = new MySqlConnection("server=localhost;
port = 3306;
username=root;
password=root;
database=shifr");
```

Рисунок 3.8 – Шлях для підключення MySQL бази даних

Створення умови для підключення до сервісу на рисунку 3.9.

```
public void openConnection()
{
if (connection.State == System.Data.ConnectionState.Closed)
connection.Open();
}
```

Рисунок 3.9 – Умова для підключення до сервісу

Створення умови для відключення від сервісу на рисунку 3.10.

```
public void closeConnection()
{
if (connection.State == System.Data.ConnectionState.Open)
connection.Close();
}
public MySqlConnection getConnection()
{
 return connection;
}
}
```

Рисунок 3.10 – Умова для відключення від сервісу

Для отримання даних із бази даних в програмі була розроблена наступна функція new database (newDB) зображена на рисунку 3.11.

```
DB db = new DB();
DataTable table = new DataTable();
MySqlDataAdapter adapter = new MySqlDataAdapter();
```

Рисунок 3.11 – Функція new database для отримання даних із бази даних

Команда виклику даних із бази даних зображена на рисунку 3.12.

```
MySqlCommand command = new MySqlCommand("SELECT * FROM `users`",
db.getConnection());
adapter.SelectCommand = command;
```

Рисунок 3.12 – Команда виклику даних із бази даних

Команда заповнення першої таблиці в інтерфейсі програми зображена на рисунку 3.13 далі.

```
adapter.Fill(table);
```

Рисунок 3.13 – Команда заповнення першої таблиці в інтерфейсі програми

Для додавання в базу даних інформації було використано наступну частину коду з використанням команди MySqlCommand зображену на рисунку 3.14.

```
DB db2 = new DB();
DataTable table2 = new DataTable();
MySqlDataAdapter adapter2 = new MySqlDataAdapter();
```

Рисунок 3.14 – Команда MySqlCommand для додавання в базу даних інформації

Безпосередня команда заповнення бази даних для полів порядкового номеру, ім’я, прізвища та оцінки зображена на рисунку 3.15.

```
MySqlCommand command2 = new MySqlCommand("INSERT INTO `start_basa` (`id`,
`name`, `sername`, `mark`)
VALUES (@id, @name, @sername, @mark)", db2.getConnection());
```

Рисунок 3.15 – Безпосередня команда заповнення бази даних

Присвоювання значення для поля порядкового номеру зображено на рисунку 3.16.

```
command2.Parameters.Add("@id", MySqlDbType.VarChar).
Value = id;
```

Рисунок 3.16 – Присвоювання значення для поля порядкового номеру

Присвоювання значення для поля імені зображено на рисунку 3.17.

```
command2.Parameters.Add("@name", MySqlDbType.VarChar).
Value = name;
```

Рисунок 3.17 – Присвоювання значення для поля імені

Присвоювання значення для поля прізвища зображено на рисунку 3.18.

```
command2.Parameters.Add("@sername", MySqlDbType.VarChar).
Value = sername;
```

Рисунок 3.18 – Присвоювання значення для поля прізвища

Присвоювання значення для поля оцінки зображено на рисунку 3.19.

```
command2.Parameters.Add("@mark", MySqlDbType.VarChar).
Value = mark;
adapter2.SelectCommand = command2;
```

Рисунок 3.19 – Присвоювання значення для поля оцінки

Команда заповнення другої таблиці в інтерфейсі програми зображено на рисунку 3.20.

```
adapter2.Fill(table2);
```

Рисунок 3.20 – Команда заповнення другої таблиці в інтерфейсі програми

Для видалення із бази даних використовуємо команду MySqlCommand зображену на рисунку 3.21.

```
string id = textBox6.Text;
DB db2 = new DB();
DataTable table2 = new DataTable();
MySqlDataAdapter adapter2 = new MySqlDataAdapter();
```

Рисунок 3.21 – Команда MySqlCommand для видалення із бази даних

Безпосередня команда видалення з бази даних усіх полів за вказаним значенням порядкового номеру зображена на рисунку 3.22.

```
MySqlCommand command2 = new MySqlCommand("DELETE FROM `start_basa`
WHERE `start_basa`.`id` = @id", db2.getConnection());
command2.Parameters.Add("@id", MySqlDbType.VarChar).Value = id;
adapter2.SelectCommand = command2;
adapter2.Fill(table2);
```

Рисунок 3.22 – Безпосередня команда видалення з бази даних

3.4 Розробка модулю шифрування

Розробка модулю шифрування розпочинається в першу чергу з підключення всіх необхідних бібліотек для коректного виконання функцій та команд програми. Підключення бібліотек зображено на рисунку 3.23.

```
using MySql.Data.MySqlClient;
using MySqlX.XDevAPI.Relational;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;
```

Рисунок 3.23 – Підключення всіх необхідних бібліотек

Наступним кроком було зроблено окремі функції для шифрування.

Перша функція шифрує строку вбудованим в неї спрощеним симетричним алгоритмом DES (згідно документації від Майкрософт) відштовхуючись додатково від довжини шифрованого слова, вона шифрує запис таким чином, щоб його можна було розшифрувати лише за умови авторизації в програмі [3]. Вона виглядає наступним чином на рисунку 3.24.

```
public string shifr1(string toEncrypt)
{
char[] key = { 'U', 'T', 'C' };
string output = "";
```

Рисунок 3.24 – Функція шифрування алгоритмом Data Encryption Standard

За рахунок циклу for відбувається проходження по всіх елементах та відбувається шифрування даних в базі даних зображеного на рисунку 3.25.

```
for (int i = 0; i < toEncrypt.Length; i++)
output += toEncrypt[i] ^ key[i % key.Length];
 return output;
}
```

Рисунок 3.25 – Цикл функції шифрування алгоритмом Data Encryption Standard

А код початку другої функції на основі шифрування Вільямса (покращеного RSA) наведено на рисунку 3.26.

```
public string shifr2(string text)
{
string s = text;
```

Рисунок 3.26 – Початок функції на основі шифрування Вільямса

Перетворення трьох простих заданих текстом чисел в еквівалентне 32-бітове ціле число для обчислення ключів шифрування зображено на рисунку 3.27.

```
int k = Convert.ToInt32(textBox8.Text);
int n = Convert.ToInt32(textBox9.Text);
int K = Convert.ToInt32(textBox10.Text);
string ans = "";
```

Рисунок 3.27 – Перетворення трьох простих заданих текстом чисел

Далі за рахунок циклів for, зображених на рисунку 3.28, відбувається проходження по всіх елементах та відбувається шифрування даних в базі даних, з урахуванням трьох простих чисел, а не двох, як в класичному алгоритмі RSA.

```
for (int i = 0; i < K - 1; ++i)
s += '0';
string str = "";
for (int i = 0; i < K; ++i)
str += "0";
for (int i = 0; i < s.Length; i += k)
{
for (int j = 0; j < k; ++j)
str = s[i + j] + str;
string s2 = "";
for (int j = 0; j < K; ++j)
s2 += str[j];
```

Рисунок 3.28 – Цикли для проходження по елементах для шифрування

Далі йде розрахунок, зображений на рисунку 3.29, числа Лежандра згідно вказаних в умовах значень для наших простих чисел k, n, K, необхідних для виконання методу шифрування Вільямса.

```
if (k == 1 && n == 2 && K == 3){
ans += Convert.ToString((Convert.ToInt32(s2[0]) - 48 + Convert.ToInt32(s2[1]) - 48 + Convert.ToInt32(s2[2]) - 48) % 2);
ans += Convert.ToString((Convert.ToInt32(s2[0]) - 48 + Convert.ToInt32(s2[2]) - 48) % 2);}
if (k == 1 && n == 2 && K == 4){
ans += Convert.ToString((Convert.ToInt32(s2[0]) - 48 + Convert.ToInt32(s2[1]) - 48 + Convert.ToInt32(s2[2]) - 48 +
Convert.ToInt32(s2[3]) - 48) % 2);
ans += Convert.ToString((Convert.ToInt32(s2[0]) - 48 + Convert.ToInt32(s2[2]) - 48 + Convert.ToInt32(s2[2]) - 48) % 2);}
if (k == 1 && n == 2 && K == 5){
ans += Convert.ToString((Convert.ToInt32(s2[0]) - 48 + Convert.ToInt32(s2[2]) - 48 + Convert.ToInt32(s2[3]) - 48 +
Convert.ToInt32(s2[4]) - 48) % 2);
ans += Convert.ToString((Convert.ToInt32(s2[0]) - 48 + Convert.ToInt32(s2[1]) - 48 + Convert.ToInt32(s2[4]) - 48) % 2);}
if (k == 1 && n == 3 && K == 3){
ans += Convert.ToString((Convert.ToInt32(s2[0]) - 48 + Convert.ToInt32(s2[1]) - 48 + Convert.ToInt32(s2[2]) - 48) % 2);
ans += Convert.ToString((Convert.ToInt32(s2[0]) - 48 + Convert.ToInt32(s2[1]) -48 + Convert.ToInt32(s2[2]) - 48) % 2);
ans += Convert.ToString((Convert.ToInt32(s2[0]) - 48 + Convert.ToInt32(s2[2]) - 48) % 2);}
if (k == 1 && n == 3 && K == 4){
ans += Convert.ToString((Convert.ToInt32(s2[0]) - 48 + Convert.ToInt32(s2[1]) - 48 + Convert.ToInt32(s2[2]) - 48 +
Convert.ToInt32(s2[3]) - 48) % 2);
ans += Convert.ToString((Convert.ToInt32(s2[0]) - 48 + Convert.ToInt32(s2[2]) - 48 + Convert.ToInt32(s2[3]) - 48) % 2);
ans += Convert.ToString((Convert.ToInt32(s2[0]) - 48 + Convert.ToInt32(s2[1]) - 48 + Convert.ToInt32(s2[2]) - 48) % 2);}
}
return ans;
}
```

Рисунок 3.29 – Розрахунок числа Лежандра згідно вхідних параметрів

Після виконання шифрування методами DES та Вільямса ми зможемо отримати кінцевий результат комбінованого методу шифрування.

3.5 Розробка графічного інтерфейсу

Перед запуском програми потрібно запустити локальний сервер. У нашому випадку це MAMP котрий наведено на рисунку 3.30.

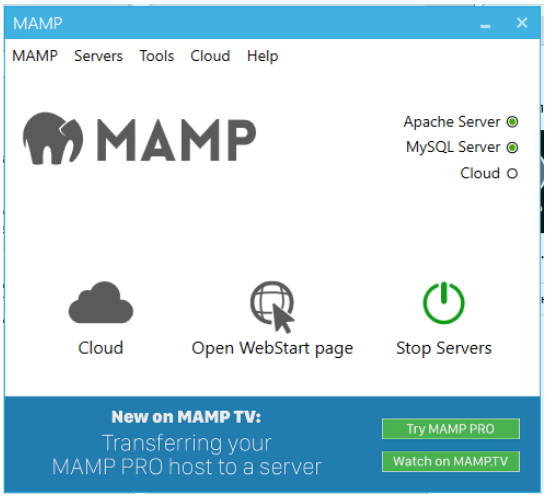


Рисунок 3.30 – Запуск локального сервера

Для запуску програми необхідно перейти наступним шляхом та запустити Base_Shifr.exe як наведено на рисунку 3.31.

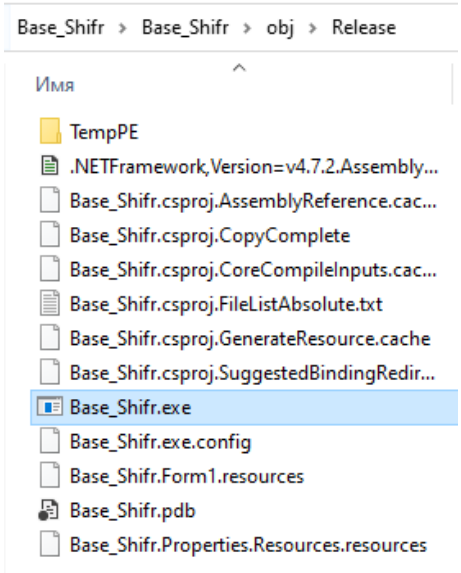


Рисунок 3.31 – Шлях до файлу

Запустивши програму, перед нами постане наступне вікно як і на рис. 3.32 нижче.

Рисунок 3.32 – Стартове вікно програми

Далі ввівши логін «admin» та пароль «admin» користувач ввійде як адміністратор, як в прикладі на рисунку 3.33, або інший логін чи пароль користувач увійде як звичайний користувач, як в прикладі на рисунку 3.34.

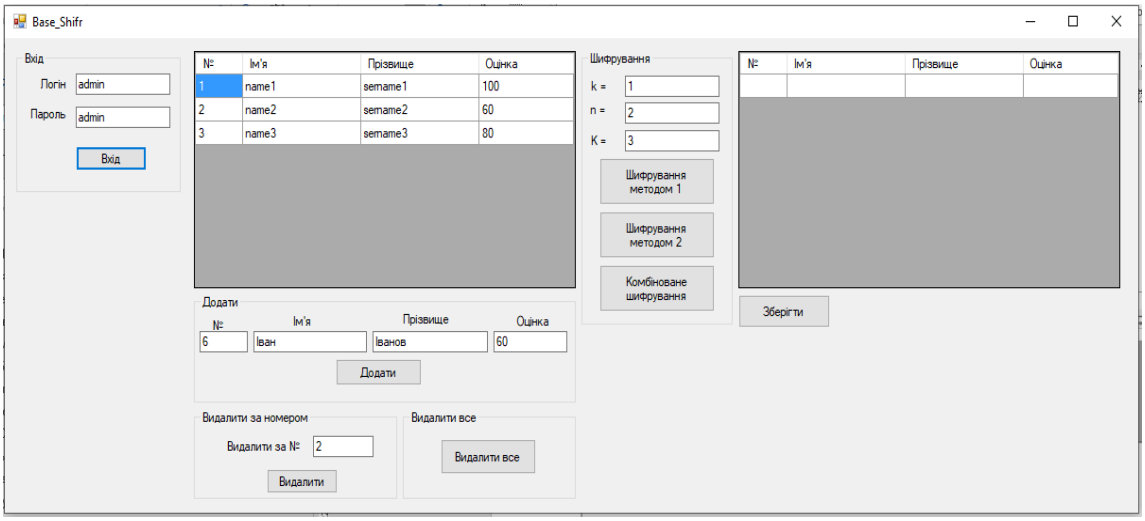


Рисунок 3.33 – Вхід адміністратора

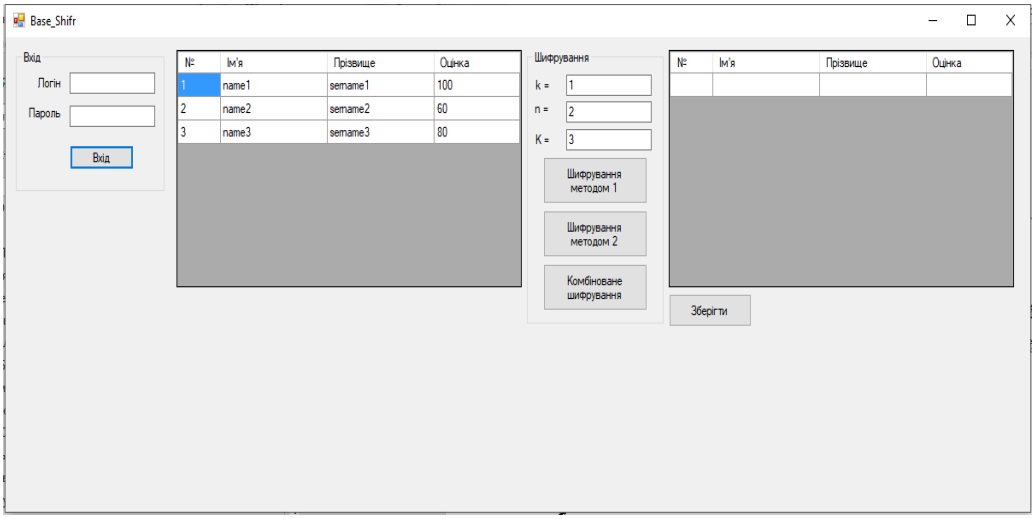


Рисунок 3.34 – Вхід звичайного користувача

В ролі адміністратора реалізовано наступні можливості.

1) Додавання запису в масив даних, як в прикладі на рисунку 3.35.

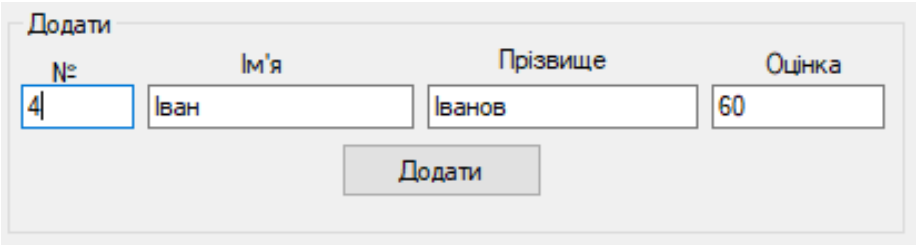


Рисунок 3.35 – Додавання даних

2) Видалення даних за номером, як в прикладі на рисунку 3.36.

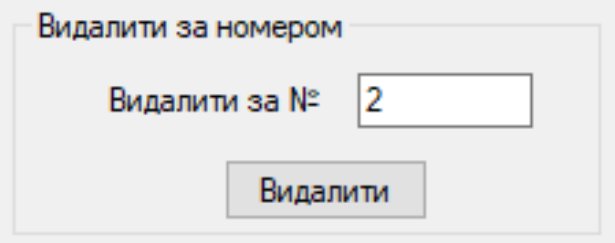


Рисунок 3.36 – Видалення даних за номером

3) Видалення всіх даних, як в прикладі на рисунку 3.37.

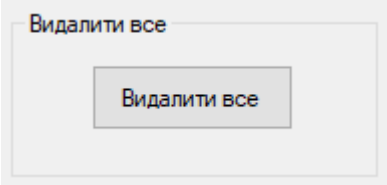


Рисунок 3.37 – Видалення всіх даних

Для шифрування даних, були створені окремі кнопки з відповідними написами, натиснувши на які відбувається шифрування даних, зазначені далі на рис. 3.38, а також відповідні параметри k, n, K для шифрування методом Вільямса (покращений RSA).

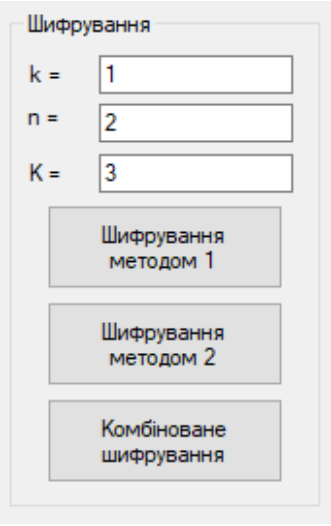


Рисунок 3.38 – Кнопки для шифрування

Для відображення бази даних було створено таблицю dataGridView, зазначену на рисунку 3.39.

№	Ім'я	Прізвище	Оцінка
1	name1	sername1	100
2	name2	sername2	60
3	name3	sername3	80

Рисунок 3.39 – Таблиця з даними

А також таблиця для відображення даних після шифрування, зазначена на рисунку 3.40.

№	Ім'я	Прізвище	Оцінка
1	59534648101	38494959534648101	100100115
2	59534648102	38494959534648102	99100
3	59534648103	38494959534648103	109100

Рисунок 3.40 – Таблиця з зашифрованими даними

3.6 Тестування системи

Під час тестування усі виникаючі помилки одразу ж виправлялись. Нижче наведено декілька прикладів тестування роботи програми.

Починалось тестування з додавання значень до бази даних, як і зазначено на рисунку 3.41.

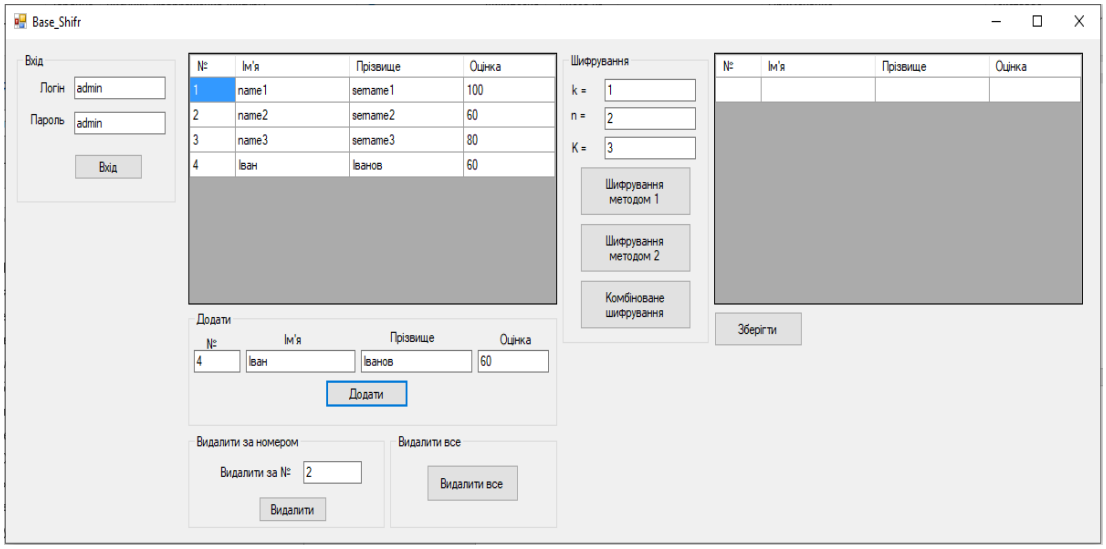


Рисунок 3.41 – Тестування додавання записів

Наступним кроком було тестування видалення за номером, як і зазначено на рисунку 3.42.

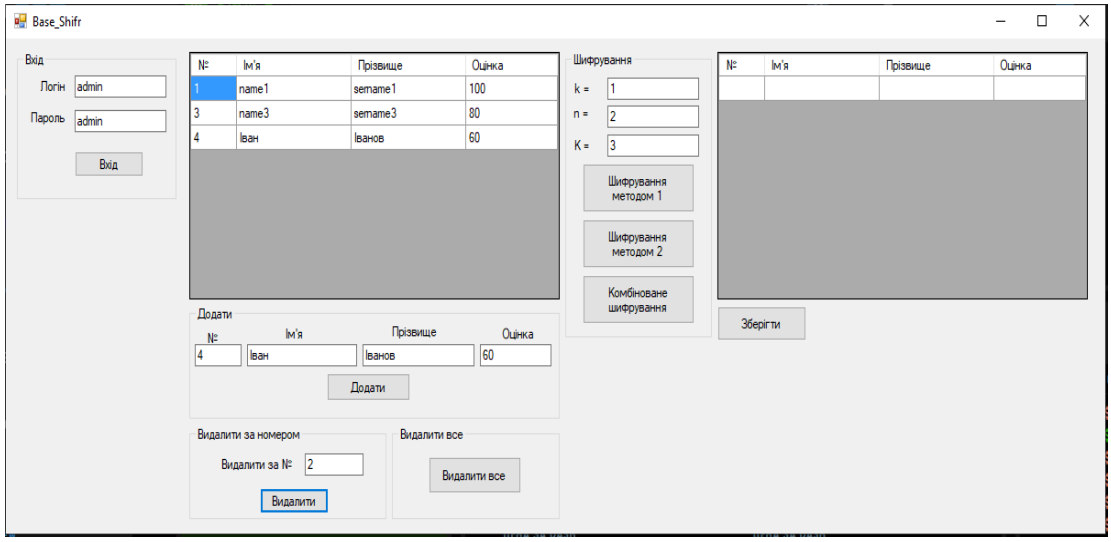


Рисунок 3.42 – Тестування видалення за номером

Також було протестована можливість видалення всіх даних. Тестування наведено на рисунку 3.43.

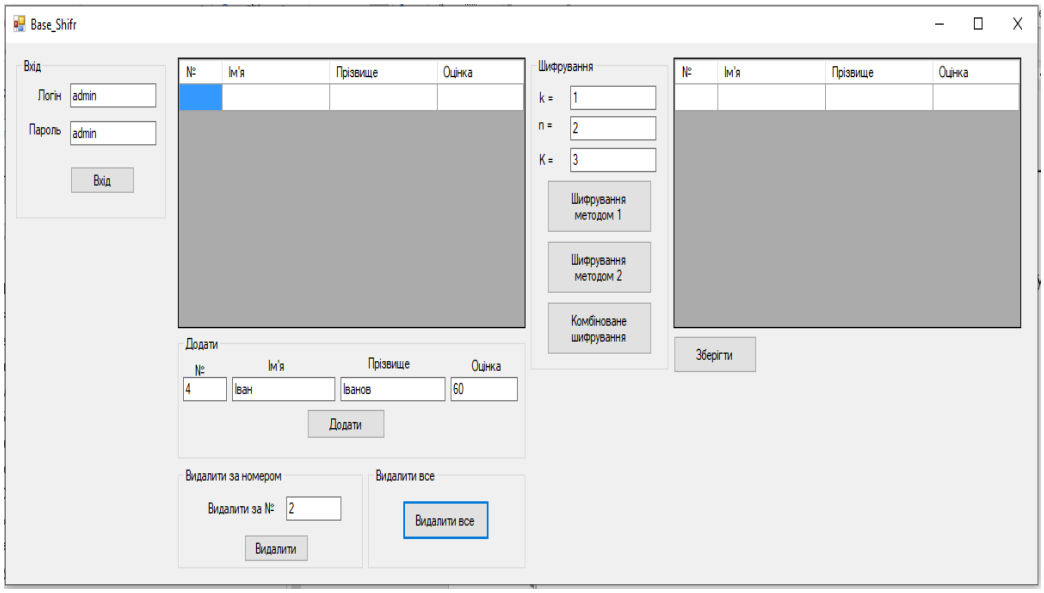


Рисунок 3.43 – Тестування видалення всіх даних

Далі перейдемо до тестування шифрування даних різними методами. Приклади шифрування однієї бази даних наведено на рисунку 3.44 та рисунку 3.45. Тестування першим методом шифрування наведено на рисунку 3.44.

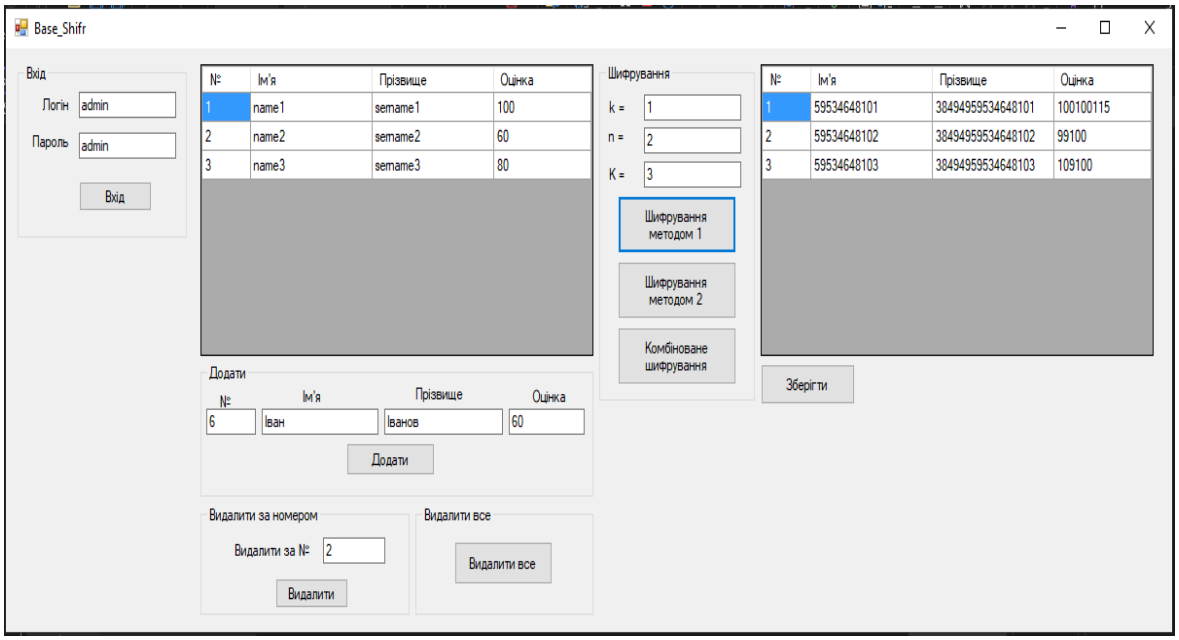


Рисунок 3.44 – Тестування шифрування першим методом

Тестування другим методом шифрування наведено на рисунку 3.45.

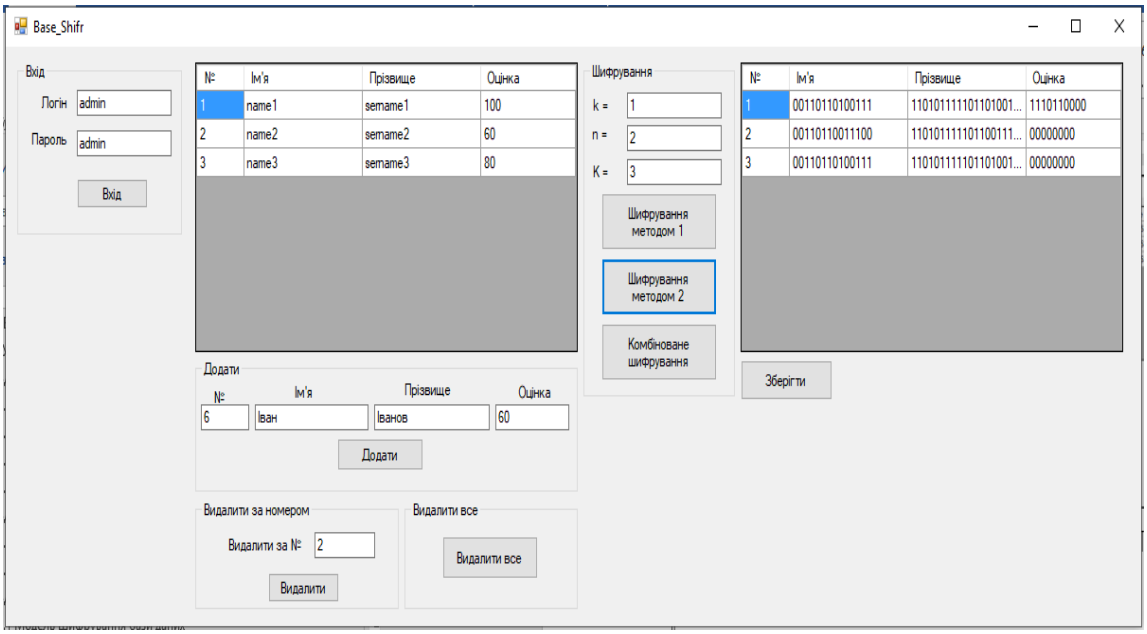


Рисунок 3.45 – Тестування шифрування другим методом

Тестування комбінованим методом шифрування даних наведено на рисунку 3.46.

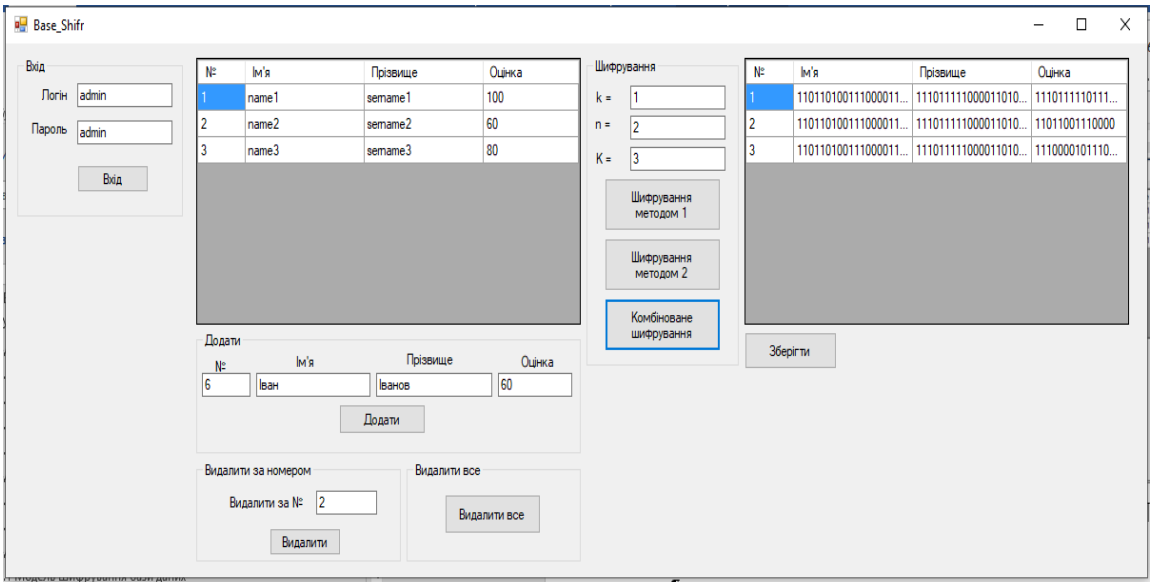


Рисунок 3.46 – Тестування комбінованого шифрування

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було зроблено аналіз сучасних концепцій створення баз даних, детально проаналізовано методи проектування баз даних, алгоритми створення баз даних та шифрування бази даних. Під час огляду та порівняння найпопулярніших СУБД було обрано систему базу даних MySQL, тому що ця СУБД повністю відповідає критеріям розробки бази даних для обраної теми, а також має в арсеналі програми процес створення бази даних, що є набагато зручнішим. Після цього було реалізовано проектування та розробку «Бази облікових даних для студентів та абітурієнтів», що складалось з наступних шагів.

1) Аналіз вимог для подальшого створення концептуальної моделі.
2) Визначено основні сутності та атрибути бази даних.
3) Створено логічний дизайн і визначено необхідні базові типи дані (реляційна база даних).
4) Реалізована база даних на основі MySQL.
5) Проаналізовано та реалізовано методи шифрування бази даних.

У ході роботи були проаналізовані класичні методи шифрування, а також існуючі модифікації класичних методів шифрування. Наведено порівняльний аналіз класичних симетричних та асиметричних шифрів. Докладно проаналізовано метод шифрування Вільямса (криптосистема Вільямса), його слабкі сторони та недоліки. Проаналізовано інструменти та прийоми, використані для аналізу методу Вільямса.

Проаналізований та відтворений метод шифрування Вільямса та результати дослідження класичного методу шифрування RSA показують, що метод шифрування Вільямса є стійкішим від RSA, але доволі складний у зміні алгоритму та потребує більше ресурсів. Також був проаналізований та відтворений метод шифрування DES, як частина комбінованого шифрування.

Серед недоліків методу Вільямса – зниження стійкості до прямих атак на основі n-факторингу відкритих ключів у два рази, а метод DES – не криптостійкий.

Відповідним застосуванням методу Вільямса є довгі ключі шифрування. Розроблено комбінований метод шифрування, який складається з асиметричного методу шифрування Вільямса та симетричного методу шифрування DES. Таким

чином, комбінований метод зберігає свої переваги перед класичним методом. Комбінований метод шифрування забезпечує підвищену швидкість роботи та підвищену стійкість до атак на основі підбору шифротексту. Він також є достатньо стійким до атак за рахунок декомпозиції відкритого ключа через його великий розмір довжини та використання симетричного і асиметричного шифрування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Database Encryption in SQL Server 2008. URL: https://learn.microsoft.com/en-us/previous-versions/sql/sql-server-2008 (дата звернення: 28.02.2023).
2. Database Encryption: Risks and Solution. URL: https://www.thalesesecurity.com/solutions/by-technology-focus/database-encryption (дата звернення: 28.02.2023).
3. Description of Symmetric and Asymmetric Encryption. URL: https://support.microsoft.com/en-us/kb/246071 (дата звернення: 28.02.2023).
4. Encryption techniques for hardware-based data storage security systems. URL: https://www.computerweekly.com/feature/Encryption-techniques-for-hardware-based-datastorage-security-systems (дата звернення: 28.02.2023).
5. Symmetric and Asymmetric Encryption. URL: https://hackernoon.com/symmetricand-asymmetric-encryption-5122f9ec65b1 (дата звернення: 01.03.2023).
6. Yoon C. E. Earthquake detection through computationally client similarity search. *Science Advances.* 2015. Vol. 4, No 12. P. 1 – 7.
7. Rodrigues M. Describing and comparing big data querying tools. *Conference on Information Systems and Technologies*: 10 – 11 mar. 2017. Madrid, 2017. P. 84 – 112.
8. Transparent Data Encryption (TDE). URL: https://msdn.microsoft.com/enus/library/bb934049.aspx (дата звернення: 02.03.2023).
9. Ліберман Я. Прозоре шифрування баз даних з використанням Microsoft SQL Server 2008. URL: http://rsdn.org/article/d/liberman (дата звернення: 02.03.2023).
10. Begoli E. Apache calcite: A foundational framework for optimized query processing over heterogeneous data sources. *International Conference on Management of Data*: 1 – 3 jul. 2019. Rome, 2019. P. 56 – 190.
11. Грайворонський М.В. Сучасні підходи до забезпечення кібернетичної безпеки в наш час. *Теоретичні і прикладні проблеми фізики, математики та інформатики*: зб. матеріалів доп. учасн. XVII Міжнар. наук.-практ. конф., м. Київ, 19 жовт. 2019 р. Київ, 2019. С. 1 – 8.
12. Kramer F. Cyberpower and National Security: Policy Recommendations for a Strategic Framework. *National Defense University Press*. 2009. Vol. 9, No 4. P. 214 – 588.
13. Doersch C. What makes Paris look like Paris? *Communications of the ACM*. 2015. Vol. 12, No 2. P. 18 – 84.

14. СУБД Postgres. URL: https://postgres.com/ (дата звернення 04.03.2023).

15. SQLite. URL: https://www.sqlite.org/index/ (дата звернення 04.03.2023).

16. MySQL Pro. URL: https://www.mysql.com/ (дата звернення 04.03.2023).

17. F. Chang F. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems*. 2008. Vol. 1, No 17. P. 12 – 42.

18. Dobrynin I.S., Poliezhaiev A.V., Zhuravka A.V. React Framework. *Conference of Free and Open Software*: 16 – 18 nov. 2021. Kharkiv, 2021. P. 9.

19. Lemeshko O.V., Poliezhaiev A.V., Zhuravka A.V. Free Version Management Systems. *Conference of Free and Open Software*: 16 – 18 nov. 2021. Kharkiv, 2021. P. 10 – 11.

20. Snegurov A.V., Poliezhaiev A.V., Zhuravka A.V. Advantages of WPF Platform. *Conference of Free and Open Software*: 16 – 18 nov. 2021. Kharkiv, 2021. P. 11 – 12.