КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет автоматизації інформаційних технологій

Кафедра інформаційних технологій

(назва випускової кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

на тему:

ЗАХИСТ ДАНИХ ВІД АТАК ТИПУ “ЛЮДИНА ПОСЕРЕДЕНІ” (MITM) У ХМАРНИХ СЕРВІСАХ

Тімохович Владислав Геннадійович

Київ 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ

Завідувачка кафедри ІТ

д.т.н., доцент Гончаренко Т.А.

„___” _____2025 року

КВАЛІФІКАЦІЙНА РОБОТА

ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему: «**ЗАХИСТ ДАНИХ ВІД АТАК ТИПУ “ЛЮДИНА ПОСЕРЕДЕНІ” (MITM) У ХМАРНИХ СЕРВІСАХ**»

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач
Тімохович Владислав Геннадійович
122 «Комп’ютерні науки»
(спеціальність)
Інформаційні управляючі системи і технології
(освітня програма)
Групи КН-21-2
Керівник Рябчун Ю.В.
(прізвище та ініціали)
Доктор філософії
(вчене звання, науковий ступінь)
Рецензент к.т.н., доц. Баліна О.І.
(Прізвище та ініціали)

Ідентичність підтверджую

Київ, 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет:	Автоматизації інформаційних технологій
Випускова кафедра:	Інформаційних технологій
Освітній ступінь:	Бакалавр
Спеціальність:	Комп’ютерні науки
Освітня програма:	Інформаційні управляючі системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТ

Тетяна ГОНЧАРЕНКО

_____

„___”_____2025 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ ВИПУСКНОЇ РОБОТИ НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

	Тімохович Владислав Геннадійович

1. Тема роботи ЗАХИСТ ДАНИХ ВІД АТАК ТИПУ “ЛЮДИНА ПОСЕРЕДЕНІ” (MITM) У ХМАРНИХ СЕРВІСАХ

затверджена наказом ректора КНУБА № 235/23/25 від «14» лютого 2024 року

2. Керівник роботи	Рябчун Юлія Володимирівна, PhD

3. Строк подання Здобувачем роботи до захисту_____

4. Зміст пояснювальної записки за розділами:

Р.1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Р.2 ПРОЕКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Р.3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Р.4 ЕРГОНОМІКА ІТ ТА ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ

5. Робота викладена на 87 аркушах, містить 1 додаток, 3 таблиці, 11 рисунків, список використаної літератури із 30 найменувань.

6. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1	20.01.2025
Розділ 2	15.02.2025
Розділ 3	20.03.2025
Розділ 4	10.05.2025
Остаточне оформлення роботи	25.05.2025
Направлення роботи для перевірки на плагіат	25.05.2025
Попередній захист роботи на випусковій кафедрі	26.05.2025
Направлення роботи на рецензування	26.05.2025

7. Консультанти розділів атестаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта	Перевірив	
		дата	підпис
Розділ 1	Рябчун Ю.В.,доц.каф.ІТ	20.01.2025	
Розділ 2	Рябчун Ю.В.,доц.каф.ІТ	15.02.2025	
Розділ 3	Рябчун Ю.В.,доц.каф.ІТ	20.03.2025	
Розділ 4	Рябчун Ю.В.,доц.каф.ІТ	10.05.2025	

8. Дата видачі завдання листопад 2024 р.

Зав. кафедри			Гончаренко Т.А.
	(підпис)		(прізвище та ініціали)
Керівники			Рябчун Ю.В.
	(підпис)		(прізвище та ініціали)
Здобувач			Тімохович В.Г.

АНОТАЦІЯ

Тімохович В.Г. Захист даних від атак типу “людина посередені” (MITM) у хмарних сервісах.

Атестаційна випускна робота бакалавра за спеціальністю 122 «Комп’ютерні науки», освітня програма «Інформаційні управляючі системи та технології». – Київський національний університет будівництва та архітектури. – Київ, 2025.

У дипломній роботі розглядаються питання забезпечення безпеки даних у хмарних сервісах, зокрема захист від атак типу "людина посередині" (MITM). Проаналізовано сучасні методи шифрування та протоколи передачі даних, які мінімізують ризики перехоплення та модифікації інформації. Досліджено роль сертифікатів SSL/TLS, механізмів багатофакторної автентифікації та безпечних з'єднань у запобіганні подібним атакам. Окрему увагу приділено методам шифрування та розроблено програму для шифрування даних. У роботі також розглянуто потенційні вразливості та запропоновані шляхи їх усунення.

Робота викладена на 87 аркушах, містить 1 додаток, 3 таблиці, 11 рисунків, список використаної літератури із 30 найменувань.

Ключові слова: Visual Studio, C#, MITM, IDS, UML, Windows Form, API.

SUMMARY

Timokhovych V.G. Data protection against man-in-the-middle attacks (MITM) in cloud services.

Bachelor's thesis for a bachelor's degree in specialty 122 “Computer Science”, educational program “Information Management Systems and Technologies.” - Kyiv National University of Construction and Architecture - Kyiv, 2025.

The thesis deals with the issues of data security in cloud services, in particular, protection against man-in-the-middle (MITM) attacks. Modern encryption methods and data transmission protocols that minimize the risks of interception and modification of information are analyzed. The role of SSL/TLS certificates, multi-factor authentication mechanisms, and secure connections in preventing such attacks is explored. Special attention is paid to encryption methods and a program for data encryption is developed. The paper also discusses potential vulnerabilities and suggests ways to eliminate them.

The work is presented on 87 pages, contains 1 appendix, 3 tables, 11 figures, and a list of 30 references.

Keywords: Visual Studio, C#, MITM, IDS, UML, Windows Form, API.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Visual Studio - Інтегроване середовище розробки;

C# - Об’єктно-орієнтована мова програмування;

MITM - це вид атаки, коли зловмисник здійснює атаку, щоб витягти інформацію з вашого з’єднання з інтернетом;

IDS - це критично важливий інструмент для забезпечення кібербезпеки в сучасному світі

UML - універсальна мова моделювання**;**

Windows Forms - інтерфейс програмування додатків (API);

API - це спосіб взаємодії комп'ютерних програм між собою.

ВСТУП

Метою даної роботи є розробка та впровадження ефективної моделі захисту даних у хмарних сервісах, яка мінімізує ризики MITM-атак і забезпечує безпечну передачу інформації. Створення програми один із методів шифрування текстів.

Досліджуючи тему, хочемо системно проаналізувати проблему, оцінити можливі ризики та запропонувати ефективні рішення. Завдання дослідження формуються на основі аналізу проблематики захисту даних у хмарних сервісах від атак типу "людина посередині" (MITM). Основну увагу буде приділено забезпеченню конфіденційності, цілісності та доступності інформації, яка передається через відкриті мережі.

Завдання дослідження:

1) Теоретичне обґрунтування сутності атаки типу "людина посередині" (MITM), її механізми та особливості у контексті хмарних сервісів.

2) Аналіз ризиків і вразливостей.

3) Огляд методів захисту та шифрування даних

4) Розробка пропозицій оптимальних методів та інструментів для захисту хмарних сервісів від MITM-атак.

5) Розробити алгоритм або методику, що забезпечує підвищення рівня безпеки передавання даних.

Об’єкт дослідження – системи захисту даних у хмарних обчисленнях, включаючи методи шифрування, протоколи безпечного з'єднання та механізми автентифікації користувачів.

Предмет дослідження – методи та технології захисту даних у хмарних сервісах від атак типу "людина посередині" (MITM), зокрема використання криптографічних протоколів (TLS/SSL), багатофакторної автентифікації, сертифікатів безпеки та механізмів виявлення перехоплення даних.

Розділ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

Хмарні сервіси стають все більш важливими в сучасному світі, оскільки вони забезпечують зручний доступ до обчислювальних ресурсів і зберігання даних. Зі збільшенням використання хмарних технологій збільшується і обсяг даних, що зберігаються та обробляються в хмарних службах. Тому забезпечення надійності табезпеки даних, що зберігаються та обробляються в хмарних сервісах, є головним пріоритетом [1].

Хмарні технології стають ключовим елементом у досягненні надійності пропонуючи значні переваги в хмарних середовищах (ХС), оптимізації обробки даних, прийнятті рішень, економічній ефективності, продуктивності, масштабованості тощо. Втім, лише перехід на хмарні технології не здатен повною мірою вирішити питання безпеки та конфіденційності даних. Натомість розвиток технологій та хмарних рішень веде до зростання потенціалу загроз. Сьогодні, більше ніж коли-небудь, важливо забезпечити кваліфіковане управління та контроль за хмарними сервісами, зокрема високий рівень взаємодії між ІТ-фахівцями, експертами з управління ризиками та спеціалістами з кібербезпеки для ефективного протистояння сучасним загрозам і дотримання нормативних вимог.

1.1 Аналіз предметної області

Хмарні сервіси є одним із важливих компонентів сучасних інформаційних технологій, котрі забезпечують готовність, масштабованість та гнучкість засобів для роботи з даними. Приватні особи та корпоративні суб’єкти використовують їх для перерахування, зберігання та передачі інформації. Тому порушники користуються відкритим очима всіх хмарних середовищ та інтернет-з’єднань.

Однією з основних загроз є кібератаки, що можуть призвести до втрати конфіденційної інформації, порушення роботи критичних інфраструктур, а також нанесення фінансових збитків. Інші загрози включають соціальний інжиніринг, шахрайства, віруси та інші види зловмисного програмного забезпечення.

Загрози в кібербезпеці постійно зростають, оскільки технології швидко розвиваються, і зловмисники постійно вдосконалюють свої методи [2].

Деякі основні загрози в кібербезпеці:

1. ***Кібератаки*:** зловмисники можуть використовувати різноманітні методи для вторгнення в комп'ютерні системи, мережі чи програми з метою збору конфіденційної інформації або завдання шкоди [2].

2. ***Віруси, черв'яки та троянські коні*:** це види шкідливого програмного забезпечення, які можуть самостійно поширюватися та завдавати шкоди комп'ютерним системам, викрадаючи чи вносячи зміни в їх функціонал [2].

3. ***Фішинг*:** атаки, які використовують соціальний інженерінг для отримання конфіденційної інформації, такої як паролі чи банківські реквізити, шляхом видавання себе за довірених осіб або організацій [2].

4. ***DDoS-атаки*:** атаки на відмову в обслуговуванні спрямовані на перевантаження ресурсів мережі чи серверів, щоб призвести до відмови в обслуговуванні для закінчення користувачів [2].

5. ***Розкрадання даних:*** Незаконне отримання, використання чи розголошення конфіденційної інформації, такої як особисті дані або комерційна таємниця [1].

6. ***Кібертероризм*:** використання кібератак для створення паніки, нанесення шкоди критичним інфраструктурам або викликання загроз національній безпеці [2].

7. ***Кібертероризм:*** використання кібератак для створення паніки, нанесення шкоди критичним інфраструктурам або викликання загроз національній безпеці [2].

8. ***Недостатня кібергігієна:*** відсутність усвідомлення та належних практик безпеки серед користувачів і персоналу організацій може викликати безпекові проблеми [2].

9. ***Загрози Інтернету речей (IoT):***нестача безпеки в пристрояхIoT може призвести до атак на підключені до Інтернету пристрої, включаючи вбудовані системи автомобілів, побутові пристрої та інше [2].

Можна зробити висновок, що захист від цих загроз вимагає комплексного підходу, який включає технічні, організаційні та культурні заходи безпеки.

Основна проблема дослідження полягає в недостатньому рівні захисту даних у хмарних сервісах під час передачі між клієнтом і сервером, що створює вразливості до атак типу "людина посередині" (MITM) [2].

Атака “людина посередині” (MITM) – це вид атаки, коли зловмисник здійснює атаку, щоб витягти інформацію з вашого з’єднання з інтернетом [1].

Такі атаки дозволяють зловмисникам:

1. Перехоплювати передані дані, зокрема конфіденційну інформацію, логіни, паролі, фінансові дані, тощо.
2. Модифікувати або підмінювати дані без відома користувача, що може призвести до втрати даних чи їх компрометації.
3. Обманювати користувача за допомогою фальшивих сертифікатів або переправлення трафіку через зловмисні вузли, створюючи ілюзію безпечного підключення.

Основними причинами вразливості хмарних сервісів є :

1. Недостатня захищеність мережевих протоколів, наприклад, слабкі конфігурації SSL/TLS.

2. Відсутність додаткових механізмів автентифікації, таких як багатофакторна автентифікація.

3. Використання публічних або небезпечних Wi-Fi-мереж, що створює можливості для MITM-атак.

Проаналізувати сучасний стан розвитку хмарних технологій і їхню вразливість до MITM-атак.

Розглянути типи MITM-атак (ARP-спуфінг, підробка SSL-сертифікатів, HTTP-проксі тощо) та їх специфіку в хмарних середовищах.

Аналізуючи предметну область, слід окреслити основні проблеми:

1) Як мінімізувати ризики перехоплення даних у хмарних середовищах?
2) Які технології найкраще впроваджувати для боротьби з MITM-атаками?
3) Як забезпечити баланс між безпекою, продуктивністю та вартістю?

1.2 Формулювання проблеми

На основі проведеного аналізу формується конкретна проблема: як захистити дані користувачів у хмарних сервісах від ризику їх перехоплення та підміни зловмисниками? Проблема стосується захищеного передавання даних, автентифікації сторін, перевірки сертифікатів і шифрування даних, які є базовими аспектами захисту від MITM-атак.

Визначити основні точки входу та сценарії MITM-атак у хмарних сервісах.

Проаналізувати основні ризики для конфіденційності, цілісності та доступності даних у хмарному середовищі.

Захист від кіберзагроз включає в себе використання різноманітних інструментів та стратегій. Розглянемо деякі з найважливіших інструментів захисту, які використовуються для забезпечення кібербезпеки, а саме :

1. ***Антивірусне програмне забезпечення (Антивіруси)*** **[2]*:*** ці програми призначені для виявлення, блокування та вилучення зловмисних

програм, таких як віруси, черв'яки, троянці та інші види шкідливого програмного забезпечення. Вони оновлюються регулярно для розпізнавання нових загроз.

2. ***Системи виявлення вторгнень (IDS) та Системи запобігання вторгнень (IPS)*** **[2]*:*** IDS виявляють аномальну або підозрілу активність в мережі, попереджаючи адміністратора про можливий вторгнення. IPS вживають заходів для блокування або обмеження доступу зловмисникам.

3. ***Шифрування даних*** **[2]*:*** шифрування використовується для захисту конфіденційної інформації шляхом перетворення її в незрозумілий для сторонніх код. Такий підхід важливий для захисту даних під час їх передачі чи зберігання.

4. ***Множинний фактор аутентифікації*** **[2]*:*** для підвищення рівня захисту, системи використовують множинний фактор аутентифікації, де користувач повинен пройти кілька етапів перевірки, таких як введення пароля та використання підтверджень на мобільних пристроях.

5. ***Системи резервного копіювання та відновлення (Backup & Recovery)*** **[2]*:*** регулярне створення резервних копій важливої інформації дозволяє відновити дані у випадку кібератаки або випадкового видалення.

6. ***Системи аналізу журналів (Log Analysis)*** **[2]*:*** Аналіз журналів дозволяє виявляти надзвичайні або підозрілі події в комп'ютерній системі, що може свідчити про кібератаку.

7. ***Кібергігієна та організаційні заходи*** **[2]*:*** важливо розвивати культуру кібербезпеки серед персоналу, проводити навчання з питань безпеки та встановлювати політики захисту інформації. Забезпечення правильного використання та захисту інформації на рівні користувача. Це включає в себе навчання персоналу щодо безпеки в інтернеті та усвідомлення можливих загроз

8. ***Міжнародне співробітництво*** **[2]*:*** розвиток міжнародних стандартів та угод для спільного протидії кіберзагрозам та вирішення проблеми кібернепоширення на світовому рівні.

Сучасні інструменти, що використовуються зловмисниками для здійснення MITM-атак.

Можна зробити висновок, що ці інструменти, разом зі спільними зусиллями користувачів, організацій та міжнародної спільноти, є ключовими для забезпечення ефективного захисту в інтернет-просторі.

1.3 Актуальність теми

У зв’язку з цим, на сучасному етапі розвитку технологій значну частину організацій та користувачів передають і зберігають дані у хмарних сервісах, оскільки такі системи дозволяють максимально зручно, доступно і масштабовано зберігати дані. Водночас з розвитком хмарних технологій росте й імовірність кіберзлочини, а саме атаки типу «людина посередині», що стають дедалі більш субтельними та складними у відстеженні. Актуальність захисту інформації у хмарних сервісах приймає ще більші оберти через ряд факторів:

1. Збільшення обсягів даних. Щодо даного аспекту, можна зазначити, що з ростом обсягу існує більше і яскравіше інформації, яка є джерелом прибутку для загроз. Масові дані пріоритетніше;

2. Популярність відаленої роботи. Рішення про розбиття кооперативів власного середовища діяльності для різних мереж або ж надання первісних даних далекому джерелу може спричинити одноманітне поводження.

На даний момент Україна зіштовхнулася з непростою ситуацією в країні через військові дії. На міжнародної конференції з кібербезпеки, яка на початку грудня 2024 року відбулася у Києві, війну в Україні назвали першою в історії війною у кіберпросторі [3]. Але тематика захисту даних від атак типу "людина посередині" (MITM) залишається актуальною в Україні не тількичерез військові дії ,а є ще багатоключових факторів:

- Зростання цифровізації та хмарних технологій. В Україні активно впроваджуються хмарні сервіси для бізнесу, державного сектору та освіти. Усе більше підприємств і держустанов переходять до використання хмарних платформ для зберігання та обробки даних. Чим більше використовується онлайн-ресурсів, тим більша ймовірність кібератак.
- Збільшення кібератак через військові дії.Україна є однією з основних мішеней для кібератак через війну. Багато атак мають на меті перехоплення конфіденційної інформації. Зловмисники можуть використовувати MITM для перехоплення даних у публічних Wi-Fi-з'єднаннях, VPN або навіть у хмарних середовищах, щоб отримати доступ до критичної інформації.
- Низька обізнаність користувачів. Багато користувачів в Україні не знають про ризики MITM-атак і використовують незахищені мережі. Недостатнє шифрування даних або неправильна конфігурація хмарних сервісів можуть стати критичною вразливістю.
- Актуальність для державного сектору. Урядові установи використовують хмарні сервіси для зберігання важливих даних, включаючи персональну інформацію громадян. Вразливість до атак MITM може призвести до витоку чутливої інформації та поставити під загрозу національну безпеку.
- Недостатнє впровадження сучасних стандартів захисту. Не всі компанії в Україні впроваджують новітні технології, такі як TLS 1.3, DNSSEC або Zero Trust. Використання застарілих протоколів (наприклад, HTTP замість HTTPS) створює можливості для атак.
- Міжнародні вимоги щодо кібербезпеки. Для виходу на міжнародний ринок або співпраці з закордонними партнерами українські компанії повинні відповідати сучасним стандартам захисту даних. Захист від MITM є важливою складовою цих стандартів.
- Відповідність до законодавства. Україна поступово адаптує своє законодавство до вимог ЄС (наприклад, GDPR), що передбачає високий рівень

захисту даних. Захист від MITM-атак є важливим елементом забезпечення конфіденційності та цілісності інформації.

Отже, таким чином, захист від MITM-атак у хмарних сервісах має стратегічне значення для України, враховуючи сучасні кіберзагрози, економічний розвиток та необхідність захисту даних як у державному, так і в приватному секторах.

1.3.1 Аналіз основних технологій і практик захисту від атак "людина посередині" (MITM):

1. SSL/TLS (Secure Sockets Layer / Transport Layer Security) [4]:

Призначення:

- Шифрує передані дані між клієнтом і сервером.
- Автентифікує сервер (а іноді й клієнта) за допомогою цифрових сертифікатів.

Захист від MITM:

- Перешкоджає перехопленню трафіку та його підміні.
- Використовує механізми перевірки сертифікатів (PKI, OCSP, HSTS), що допомагає уникнути атак на підроблені сайти.

Уразливості:

- Атаки на довірені сертифікати (наприклад, компрометація центру сертифікації).
- SSL Stripping (зниження рівня безпеки до HTTP).
- Використання слабких алгоритмів шифрування або застарілих версій TLS (наприклад, TLS 1.0).

Рекомендації:

- Використовувати TLS 1.2 або вище.
- Включати HSTS (HTTP Strict Transport Security).
- Регулярно оновлювати сертифікати та перевіряти їх надійність.

2. VPN (Virtual Private Network) [5]

Призначення:

- Шифрує весь трафік між пристроєм і VPN-сервером.
- Ховає реальний IP-адрес користувача.

Захист від MITM:

- Запобігає перехопленню трафіку в незахищених мережах (наприклад, публічний Wi-Fi).
- Створює захищений тунель між користувачем і сервером.

Уразливості:

- Використання ненадійних або зламаних VPN-серверів.
- Витоки DNS або IP (DNS Leak, WebRTC Leak).
- Компрометація протоколу (наприклад, PPTP вразливий до атак).

Рекомендації:

1. Використовувати протоколи OpenVPN, WireGuard або IKEv2/IPSec.

2.Увімкнути функцію «Kill Switch» (автоматичне відключення інтернету при розриві VPN-з'єднання).

3.Переконатися, що VPN не витікає DNS-запити.

<u>3. Фільтрація DNS (DNSSEC, DoH, DoT)</u> [6]:

Призначення:

- Перевіряє автентичність DNS-запитів.
- Запобігає атакам на рівні доменних імен (наприклад, DNS Spoofing).

Захист від MITM:

- DNSSEC (DNS Security Extensions) використовує цифрові підписи для перевірки автентичності записів.
- DoH (DNS over HTTPS) і DoT (DNS over TLS) шифрують DNS-запити, що робить їх невидимими для атакуючих.

Уразливості:

- Багато провайдерів не підтримують DNSSEC.
- DoH/DoT може бути заблокований на рівні мережевого фільтрування.

- Можливі атаки через компрометовані DNS-сервери.

Рекомендації:

- Використовувати DNSSEC для перевірки достовірності доменів.
- Налаштувати DoH/DoT у браузері або на рівні операційної системи.
- Використовувати надійні DNS-сервери (Google DNS, Cloudflare DNS, Quad9).

Для ефективного захисту від MITM-атак необхідно використовувати комбінацію заходів:

- Використовувати тільки зашифровані протоколи (TLS 1.2+, VPN).
- Захищати DNS-запити (DNSSEC, DoH, DoT).
- Включати багатофакторну автентифікацію, бажано без SMS.
- Регулярно перевіряти сертифікати та оновлювати системи безпеки.

1.3.2 Аналіз основних підходів до моніторингу та виявлення аномальної активності в мережевому трафіку:

1. Системи виявлення вторгнень (IDS)

Мережеві IDS (NIDS) [7]:

Призначення:

Аналізують трафік у режимі реального часу та шукають підозрілі патерни.

Приклади: Snort, Suricata, Zeek (Bro).

Методи аналізу:

Сигнатурний аналіз [7] (порівнює трафік зі знаними шаблонами атак).

Аналіз на основі поведінки (виявляє нетипову активність).

Переваги:

- Висока ефективність для відомих загроз.
- Виявляє атаки в режимі реального часу.

Недоліки:

- Не може виявити нові атаки (при сигнатурному аналізі).

- Можливі хибнопозитивні спрацьовування.

Хостові IDS (HIDS) [7].

Призначення:

Моніторять поведінку на конкретному пристрої (сервері, комп'ютері).

Приклади: OSSEC, Wazuh.

Методи аналізу:

- Контроль логінів операційної системи.
- Моніторинг змін у файлах та реєстрі.

Переваги:

- Детальна інформація про загрози всередині системи.
- Може виявляти внутрішні атаки.

Недоліки:

- Не захищає від атак на рівні мережі.

2. **Аналіз мережевого трафіку (NTA – Network Traffic Analysis) [8].**

Призначення:

Виявлення аномалій у трафіку шляхом аналізу поведінки.

Приклади: Darktrace, Vectra AI, Cisco Stealthwatch.

Методи аналізу:

- Виявлення нетипових патернів трафіку.
- Використання машинного навчання (ML) для виявлення аномалій.
- Аналіз ентропії трафіку (перевірка різноманітності запитів).

Переваги:

- Добре підходить для захисту від APT (Advanced Persistent Threats).

Недоліки:

- Високий рівень хибнопозитивів.
- Потрібна обробка великої кількості даних.

3. Використання Honeypots (пастки для атакуючих) [9].

Призначення:

Імітація вразливих систем для виявлення атак.

Приклади: Cowrie (SSH-пастка), Dionaea (пастка для експлойтів), T-Pot (комплексна honeypot-система).

Методи аналізу:

- Логування спроб атак.
- Аналіз шкідливого програмного забезпечення.

Переваги:

- Дає інформацію про нові атаки та техніки.
- Допомагає відволікати атакуючих від реальних ресурсів.

Недоліки:

- ❖ Не запобігає атакам, а лише їх реєструє.
- ❖ Потрібне правильне налаштування, щоб не дати зловмисникам доступ до мережі.

1.3.3 Розглянемо один ізтипів MITM-атак(ARP-спуфінг)

1. ARP- spoofing [10] - мережева атака канального рівня, при якій зловмисник надсилає підроблені повідомлення протоколу ARP або різновид мережевої атаки типу MITM що застосовується в мережах з використанням протоколу ARP, в основному атака застосовується в мережах Ethernet (рис.1.1). Атака заснована на недоліках протоколу ARP. Аналіз безпеки протоколу ARP показує, що, перехопивши на атакуючому вузлі усередині даного сегмента мережі широкомовний ARP-запит, можна послати помилковий запит – ARP відповідь, в якому можливо оголосити себе потрібним вузлом (наприклад, маршрутизатором), і в подальшому активно контролювати мережевий трафік вузла, впливаючи. Протокол ARP є абсолютно незахищеним. Він не володіє ніякими способами перевірки автентичності пакетів: як запитів, так і відповідей. Ситуація стає ще більш складною, коли може використовуватися мимовільний ARP. Незважаючи на ефективність мимовільного ARP, він є особливо небезпечним, оскільки з його допомогою можна запевнити віддалений вузол в

тому, що MAC-адресу будь-якої системи, що знаходиться з нею в одній мережі, змінився і вказати, яку адресу використовується тепер.

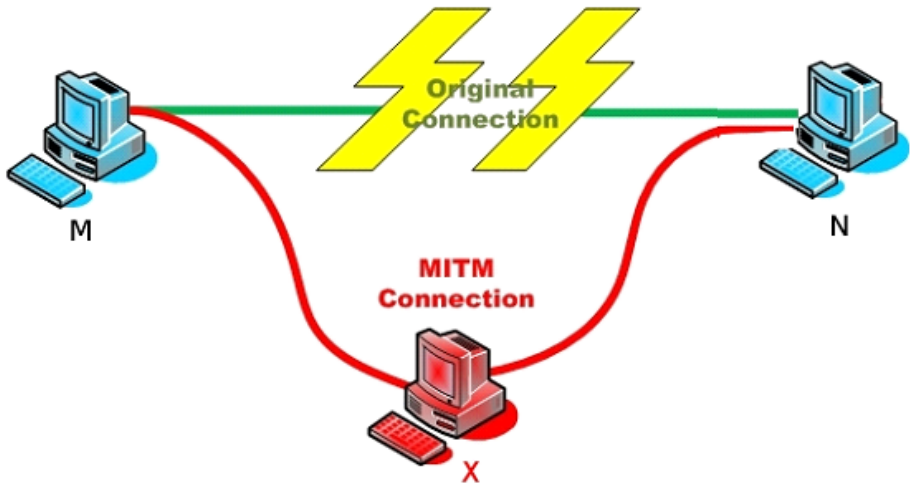

Рисунок 1.1. ARP-spoofing

Опис атаки [10]:

1. Два комп'ютери M та N в локальній мережіEthernet обмінюються повідомленнями. Зловмисник X, що знаходиться в цій мережі, хоче перехоплювати повідомлення між цими вузлами. До застосування атаки ARP spoofing на мережевому інтерфейсі вузла M ARP-таблиця містить IP та MAC-адресу вузла N, а в мережевому інтерфейсі вузла N ARP-таблиця містить IP та MAC-адресу вузла M.

2. Під час атаки ARP spoofing вузол X (зловмисник) відсилає дві ARP-відповіді (без запиту) до вузлів M та N. ARP-відповідь вузла M містить IP-адресу N й MAC-адресу X. В свою чергу ARP-відповідь вузла N містить IP адресу M й MAC-адресу X.

3. Так як комп'ютери M та N підтримують мимовільний ARP, то після отримання ARP-відповіді, вони змінюють свої ARP-таблиці, тепер ARP-таблиця M містить MAC-адресу X яка прив'язана до IP-адреси N,

ARP-таблиця N також містить MAC-адресу X котра в свою чергу прив'язана до IP-адреси M.

4. Починає виконуватися атака ARP spoofing та всі пакети між M та N проходить через X. Наприклад, якщо M хоче передати пакет комп'ютеру N, то M дивиться в свою ARP-таблицю, знаходить запис IP-адреси вузла N, бере звідти MAC-адресу та передає пакет. Пакет знаходиться в інтерфейсі X, аналізується, після чого перенаправляється до вузла N.

Отже, потрібновикористовувати шифрування для запобігання атаки ARP spoofing в локальній мережі можна використовувати протоколи шифрування даних, для захисту переданої інформації від зловмисника. Наприклад такі протоколи як PPPoE або IPSec.

1.4 Кіберзахист в хмарнихсередовищах

На даний момент Україна додатково зіштовхується зі своїми унікальними випробуваннями, зумовленими війною, що триває, та постійним кіберпротистоянням з ворогом. Це змушує країну не лише реактивно адаптуватися до цифрової ери, але й вимагає особливої уваги до питань національної безпеки та розробки стратегій кіберстійкості. Попри екзистенційні виклики, країна продемонструвала непохитність та рішучість у своєму прагненні до технологічного прогресу. І хоча зусилля уряду та приватного сектору,спрямовані на підтримку цифрової трансформації для покращення якості життя громадян та створення міцного фундаменту для подолання кіберзагроз та забезпечення стійкості критично важливих інфраструктур, є значними, та перед країною все ще стоїть чимало викликів та завдань для реалізації повного потенціалу цифрової безпеки та інновацій у державному секторі.

Основні аспекти:

- Урядові установи можуть зіштовхнутися з унікальними труднощами та перешкодами в сфері захисту своїх хмарних середовищ.
- Необхідність стратегічного підходу до використання хмарних технологій стала особливо очевидною після їх прискореного впровадження під час пандемії та повномасштабного вторгнення.
- Сьогодення вимагає від фахівців з безпеки переглянути традиційні підходи управління безпекою та захисту критично важливих активів, зокрема персональних даних громадян та конфіденційної державної інформації.
- Якщо організація ще не зробила необхідних кроків для безпеки власних хмарних рішень, вона створює умови для нових атак.

Перешкоди на шляху до прогресу:

- Довіра до зовнішніх провайдерів. Для урядових організацій довіра до зовнішніх постачальників технологій є критично важливою. Саме вони, як правило, мають доступ до конфіденційних даних і документів, компрометація яких може підірвати довіру громадськості або становити загрозу національній безпеці. У багатьох випадках саме недовіра утримувала уряди від перенесення критично важливих даних у хмару.
- Функції та обов'язки. Для успішної міграції необхідне чітке розуміння розподілу обов'язків між організаціями та їхніми провайдерами хмарних послуг, зокрема щодо безпеки даних. Ініціативи з переходу на хмарні технології зазнають невдачі через брак чіткого розуміння обов'язків та розподілу відповідальності, а також через відсутність нових професійних знань і навичок, необхідних для використання можливостей хмарних технологій всередині організації.
- Цифровий суверенітет та резидентність даних. Багато країн прагнуть цифрового суверенітету, їхнє небажання віддавати інфраструктуру та дані зовнішнім хмарним сервісам лише посилилося через вплив пандемії на економічну стабільність, і геополітичну напруженість. Сьогодні провайдери

хмарних технологій відкривають спеціалізовані центри обробки даних у все більшій кількості країн, щоб вирішити цю проблему. Крім того, діють такі ініціативи, як GAIA-X, пропозиція ЄС щодо створення наступного покоління інфраструктури даних для Європи: безпечної та федеративної екосистеми, яка підтримує цифровий суверенітет і водночас сприяє інноваціям. Наразі в цій ініціативі вже беруть участь понад 300 організацій, вона продовжує розвиватися.

- Регулювання. З нормативно-правової точки зору, Загальний регламент ЄС про захист даних (GDPR) встановлює високі стандарти конфіденційності даних для провайдерів хмарних послуг. Інші країни, зокрема Бразилія та Індія, наслідують цей приклад, приймаючи власні закони. Уряди різних країн запроваджують різні стандарти, що створює нові труднощі для провайдерів хмарних рішень.
- Передові рішення для державного сектору. Сьогодні хмарні технології стали мейнстрімом, а інноваційні хмарні сервіси, платформи та інфраструктура як ключові елементи нової цифрової економіки допомагають забезпечити високий рівень масштабованості, гнучкості та стійкості. Хмарні рішення допомагають розкрити потенціал органів державної влади, які прагнуть підвищити продуктивність праці, ефективність і знайти нові способи задовільнити очікування споживачів, що стрімко змінюються.

Багато організацій, зокрема органи державної влади, все ще перебувають на ранніх стадіях переходу до хмарної інфраструктури як послуги (IaaS), намагаючись вирішувати проблеми із застарілою архітектурою, відповідністю вимогам щодо конфіденційності даних та розподілом обов'язків між провайдерами хмарних послуг і організацією. Деякі структури можуть бути більш прогресивними та впроваджують все більш популярну нині платформу як послугу (PaaS).

Сьогодні майже кожна установа використовує хмарні програмні рішення як послугу (SaaS) для забезпечення стандартних офісних інструментів,

онлайн-навчання, корпоративних платформ управління персоналом тощо.[1]

Розділ 2. МЕТОДИ ЗАХИСТУ ВІД АТАК MITM

2.1. Обґрунтування вибору методів

Щоб забезпечити захист даних для хмарної служби перед "людина посередині " (MITM), було обрано наступні методи та технології:

1. **Протокол шифрування (TLS/SSL)** – забезпечує надійне шифрування вихідних даних між клієнтами та серверами [4].

2. **Багатофакторна автентифікація (MFA)** – це додатковий рівень безпеки, в якому код вводиться іншим пристроєм для перевірки надійності користувачів [11].

Протоколи шифрування – це багато правил та процедур, що забезпечують безпеку зв'язку та обміну даними між двома або більше сторонами по мережі [12]. Це включає використання технологій шифрування для забезпечення конфіденційності, цілісності та надійності переданої інформації. Криптографічні протоколи необхідні для забезпечення безпечного та приватного спілкування в різних областях, включаючи інтернет-банкінг, електронну комерцію, безпечні повідомлення та безпечне надсилання файлів.

2.1.1 Криптографічні протоколи

Криптографічні протоколи забезпечують захист переданих даних за допомогою різноманітних технік і механізмів. Основні складові та процеси, які при цьому задіяні, описані нижче.

1. **Обмін ключами [13].** Першим етапом створення безпечного каналу зв’язку є передача криптографічних ключів між учасниками процесу. Для цього протоколи використовують різні методи безпечної генерації і передачі ключів.

2. **Шифрування [13].** Після обміну ключами дані, призначені для передачі, шифруються за їх допомогою. Завдяки шифруванню інформація перетворюється у форму, яку складно зрозуміти стороннім особам. Серед

основних алгоритмів, застосовуваних у криптографічних протоколах, слід відзначити AES (Advanced Encryption Standard) і RC (Rivest Cipher).

3**. Перевірка цілісності [13].** Для гарантування незмінності переданих даних використовуються хеш-функції та цифрові підписи. Хеш-функції генерують унікальне значення для кожного пакета даних, що дає можливість перевірити, чи були вони змінені. Цифрові підписи забезпечують автентичність і цілісність повідомлень за допомогою методів асиметричного шифрування та приватного ключа відправника.

4. **Аутентифікація[13**]. Цей процес дозволяє підтвердити особи сторін, що обмінюються інформацією, і виключити ризик участі шахраїв. Для аутентифікації використовуються цифрові сертифікати, паролі або методи біометричної перевірки.

5. **Захищений канал [13].** Після успішного обміну ключами і проведення аутентифікації передача даних здійснюється через безпечний канал. Це забезпечує конфіденційність і захист переданої інформації від прослуховування чи модифікації. Криптографічні протоколи роблять перехоплення або зміну даних надзвичайно складними для зловмисників.

2.1.2 MFA — багатофакторна аутентифікація

Багатофакторна автентифікація (MFA) — це метод перевірки особистості [11], який потребує від користувача надання щонайменше двох різних доказів для отримання доступу до облікового запису. Лише після введення усієї необхідної інформації відкривається доступ до системи. Серед таких доказів можуть бути номер телефону, електронна пошта або відповідь на секретне запитання, відоме лише вам (рис.2.1).

Один із найпоширеніших факторів автентифікації — ваш номер телефону. У рамках використання багатофакторної автентифікації ви спочатку вводите ім’я користувача та пароль, а потім додатковий унікальний код, надісланий через текстове повідомлення на ваш мобільний телефон. Це

підтверджує не лише те, що ви знаєте свої облікові дані, але й що у вас є зареєстрований пристрій, здатний отримувати коди такого типу.

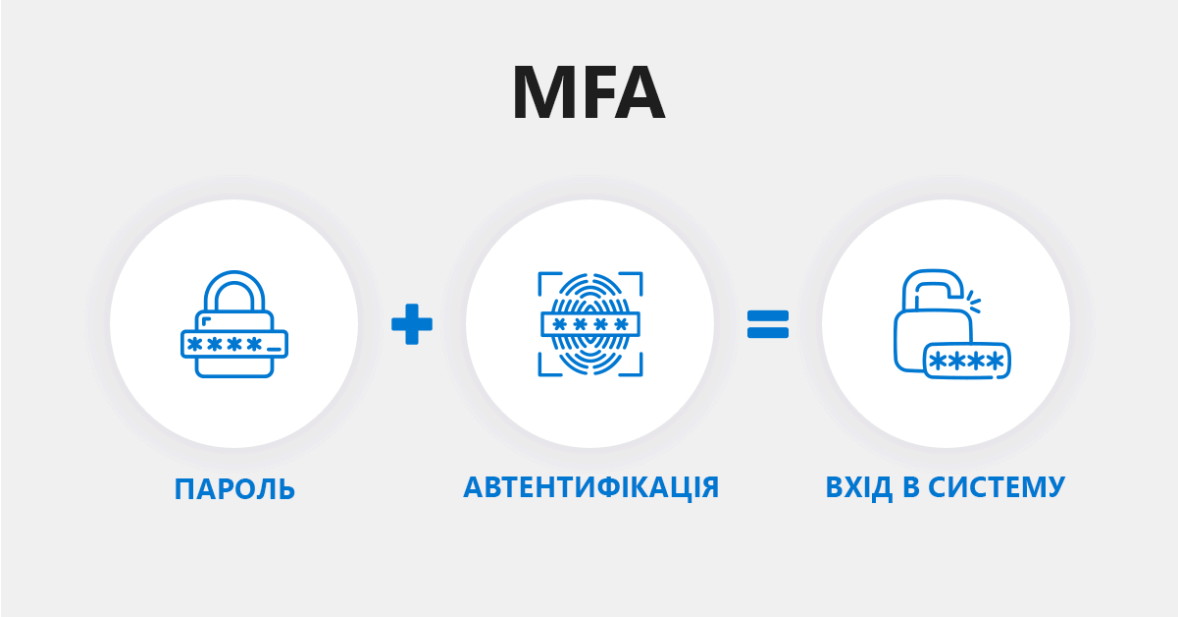

Рисунок 2.1 Багатофакторна аутенфікація

2.2 Методи шифрування

Більш детальніше зупинимося на методах шифруваннях.

Метод шифрування Цезаря [14]. **Шифр Цезаря** – це метод зсувного шифрування, де кожна буква в тексті зміщується на фіксовану кількість позицій в алфавіті. Він використовувався для приватного спілкування римським імператором Юлієм Цезарем. Принцип дії полягає у переміщенні циклу алфавіту, а важливим є кількість символів, відкладених (рис.2.2).

Якщо зіставити кожному символу алфавіту його порядковий номер (нумеруючи з 0), то шифрування і дешифрування можна виразити формулами:

$$y = (x + k)mood(n);$$

$$x = (y - k)mood(n),$$

де (x) — символ відкритого тексту, (y) — символ шифрованого тексту, (n) —потужність алфавіту, а (k) — ключ.

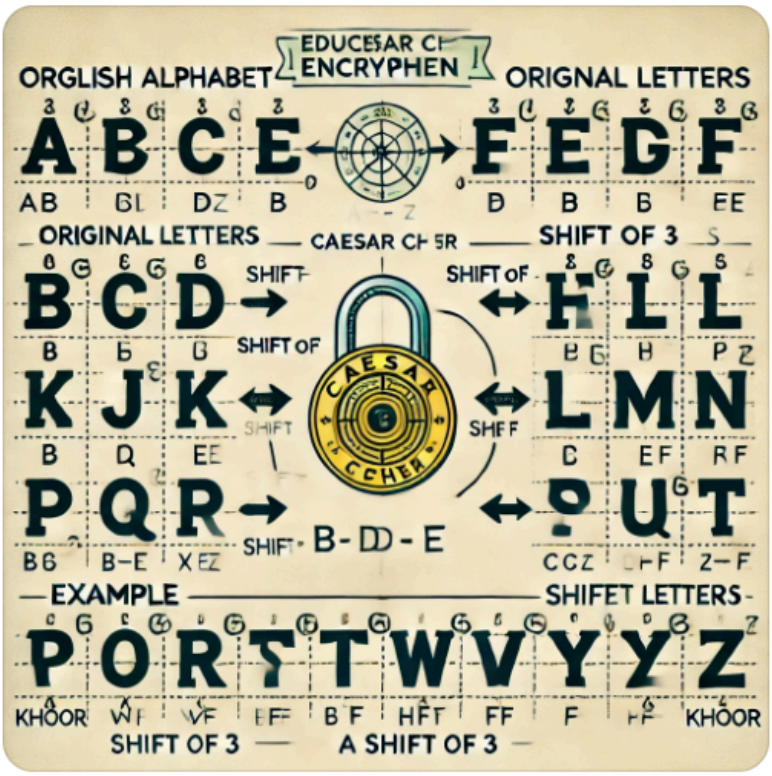

Рисунок 2.2. Схема роботи Шифру Цезаря, яка демонструє зсув літер на 3 позиції

Наочний спосіб зрозуміти, як працює цей шифр показано на рисунку рис.2.3.

1. Вихідний текст (відкритий текст):

nginx Копіювати Редагувати

HELLO

2. Ключ (зсув на 3 позиції):

css Копіювати Редагувати

A → D, B → E, C → F, ..., X → A, Y → B, Z → C

3. Зашифрований текст (шифртекст):

nginx Копіювати Редагувати

KHOOR

Рисунок 2.3. Приклад шифрування слова "HELLO" → "KHOOR"

Можна помітити, що суперпозиція двох шифрувань на ключах k_1 і k_2 є просто шифруванням на ключі $k_1 + k_2$. Більш загально, множина шифруючих перетворень шифру Цезаря утворює групу Z_n.

Шифр Вернама [14] у криптографії, система симетричного шифрування, винайдено в 1917 році співробітниками AT&T Мейджором Джозефом Моборном і Гільбертом Вернамом. Шифр Вернама (One-Time Pad – Одноразовий блокнот) є єдиною системою шифрування, для якої доведена абсолютна криптографічна стійкість.

Принцип роботи. Для утворення шифр тексту повідомлення об'єднується операцією XOR з ключем (названим одноразовим блокнотом або шифроблокнотом). Відкритий текст (plaintext) XOR-иться з випадковим ключем такої ж довжини. Ключ використовується лише один раз, після чого знищується (звідси назва "одноразовий блокнот"). Дешифрування відбувається так само – XOR з тим самим ключем.

Формула:

C=P⊕KC = P \oplus KC=P⊕K P=C⊕KP = C \oplus KP=C⊕K

де, P – відкритий текст (plaintext), K – випадковий ключ (random key), C – шифротекст (ciphertext), ⊕ – побітова операція XOR (виключне АБО).

При цьому ключ повинен володіти трьома критично важливими властивостями:

- Бути справді випадковим;
- Збігатися з розміром з заданим відкритим текстом;
- Застосовуватися тільки один раз.

Вади шифру Вермана:

 Для роботи шифру Вернама необхідна дійсно випадкова послідовність нулів та одиниць (ключ). За визначенням, послідовність, отримана з використанням будь-якого алгоритму, є не зовсім випадковою, а псевдовипадковою. Тобто, потрібно отримати випадкову послідовність неалгорітмічно (наприклад, використовуючи радіоактивний розпад ядер, створений електронним генератором білий шум або інші досить випадкові події). Щоб зробити розподіл гранично близьким до рівномірного, випадкову послідовність зазвичай проганяються через хеш-функцію на кшталт MD5.

 Проблемою є таємна передача послідовності та збереження її в таємниці. Якщо існує надійно захищений від перехоплення канал передачі повідомлень, шифри взагалі не потрібні: секретні повідомлення можна передавати з цього каналу. Якщо ж передавати ключ системи Вернама за допомогою іншого шифру (наприклад, DES), то отриманий шифр буде захищеним рівно настільки, наскільки захищений DES. При цьому, оскільки довжина ключа така ж, як і довжина повідомлення, передати його не простіше, ніж повідомлення. Шифроблокнот на фізичному носії можна вкрасти або скопіювати.

 Можливі проблеми з надійним знищенням використаної сторінки. Цьому схильні як паперові сторінки блокнота, так і сучасні електронні реалізації з використанням компакт-дисків або флеш-пам’яті.

 Якщо третя сторона якимось чином дізнається повідомлення, вона легко відновить ключ і зможе підмінити повідомлення на інше такої ж довжини.

 Шифр Вернама чутливий до будь-якого порушення процедури шифрування. Наприклад, контррозвідка США часто розшифровувала радянські та німецькі послання через неточності генератора випадкових чисел (програмний генератор псевдовипадкових чисел у німців і друкарка, що б’є по клавішах, в СРСР). Бували випадки, коли одна і та ж сторінка блокнота застосовувалася двічі — США також розшифровувати такі послання.

Наочний спосіб зрозуміти, як працює шифр показано на рисунку рис.2.4.

1. Вхідні дані:

- Відкритий текст: **"HELLO"**
- Ключ (випадковий): **XMCKL**
- Бінарне представлення (ASCII):

```ini
H = 01001000   X = 01011000
E = 01000101   M = 01001101
L = 01001100   C = 01000011
L = 01001100   K = 01001011
O = 01001111   L = 01001100
```

2. Виконання операції XOR:

```mathematica
01001000 ⊕ 01011000 = 00010000 (P)
01000101 ⊕ 01001101 = 00001000 (H)
01001100 ⊕ 01000011 = 00001111 (C)
01001100 ⊕ 01001011 = 00000111 (B)
01001111 ⊕ 01001100 = 00000011 (G)
```

Рисунок 2.4. Приклад шифрування слова "HELLO"

Якщо ми напишемо KOD у кодуванні ASCII, то для комп'ютера це буде послідовність із трьох чисел, а кожне число – це набір бітів (табл.2.1):

01001011, 01001111,01000100.

Таблиця 2.1

Приклад шифру слова KOD

Буква	Код вASCII	Біти даних
K	75	01001011
O	79	01001111
D	68	01000100

Схему роботи Шифру Вернама, що показує процес XOR-шифрування та розшифрування показано на рисунку 2.5. Вона наочно демонструє, як вихідний текст перетворюється у двійковий код, обробляється випадковим ключем і потім дешифрується назад.

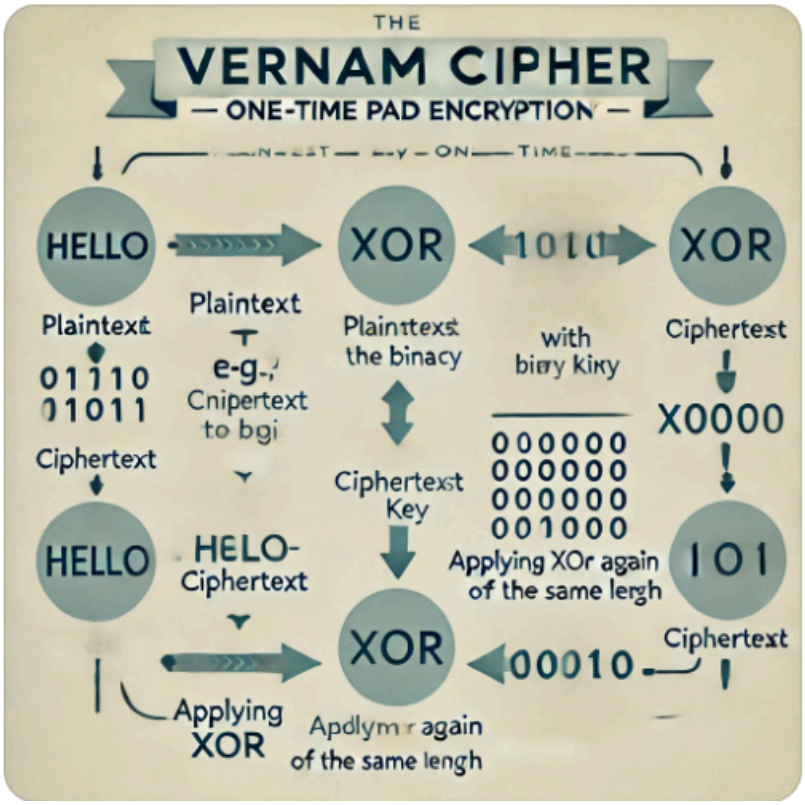

Рисунок 2.5. Схема роботи Шифру Вернама

Мінуси шифру Вернама. Якби цей шифр не мав недоліків, то всі просто використовували б його, але тепер цей код рідко використовується.

Щоб зламати цей код, потрібно мати секретний ключ з обох боків і не можна булоперехопити його.

Навіть якщо можна передати кілька ключів до майбутнього закриття кожної сторони, вони повинні підтримувати повну фізичну безпеку, щоб їх не можна було вкрасти, спостерігати, втрачати чи розгублено.

Цей код рідко використовується завдяки цим важливим вимогам безпеки.

Шифр Віженера — поліалфавітний шифр (на відміну від моноалфавітного шифру Цезаря), який у якості ключа використовує слово [15]. Якщо пронумерувати літери алфавіту від 0 до 32 (а → 0, б → 1, в → 2, …).Для цього методу є спеціальнатаблицяпо якій можна розгадати яке слово було зашифроване.

Основна ідея шифру:

- Кожна літера відкритого тексту шифрується окремим зсувом, який визначається ключовим словом.
- Використовується таблиця Віженера (або матриця Цезаря), де кожен рядок – це зсув на відповідну кількість позицій.

Матриця Віженера складається з 26 рядків, кожен з яких – це алфавіт, зміщений на 1 позицію вниз (рис.2.6). Кожен новий рядок – це зсув на +1 від попереднього.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
…																									

Рисунок 2.6. Матриця Віженера

Ось за допомогою цієї таблиці на рисунку 2.7 можна буде розпізнати слово яке зашифрували.

	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я
а	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я
б	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а
в	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б
г	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в
ґ	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г
д	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ
е	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д
є	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е
ж	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є
з	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж
и	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з
і	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и
ї	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і
й	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї
к	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й
л	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к
м	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л
н	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м
о	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н
п	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о
р	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п
с	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р
т	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с
у	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т
ф	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у
х	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф
ц	ц	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х
ч	ч	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц
ш	ш	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч
щ	щ	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш
ь	ь	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ
ю	ю	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь
я	я	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю

Рисунок 2.7. Таблиця Віженера

Шифр Віженера є одним із класичних методів симетричного шифрування, що використовує багатоалфавітну заміну. Він був розроблений у XVI столітті й довгий час вважався надзвичайно складним для розкриття. Основна ідея цього шифру полягає у використанні ключового слова для багаторазового зміщення символів вихідного тексту. На відміну від шифру простої заміни при використанні шифру Віженера однаковим буквам у відкритому тексті можуть відповідати різні букви у криптотексті. Це значно ускладнює частотний криптоаналіз.[16]

Математично шифр Віженера можна описати наступними формулами:

$$\text{Encrypt}(m_n) = (Q + m_n + k_n) \% Q;$$

$$\text{Decrypt}(c_n) = (Q + c_n - k_n) \% Q.$$

де m_n - позиція символу відкритого тексту, k_n - позиція символу ключа шифрування, Q - кількість букв в алфавіті, c_n - позиція символу зашифрованого тексту.

Принцип роботи шифру Віженера:

1.Використання таблиці Віженера (таблиці зміщених алфавітів);

2.Процес шифрування: застосування ключового слова до відкритого тексту;

3.Процес дешифрування: відновлення початкового тексту за допомогою ключа.

Криптоаналіз шифру Віженера

- Метод Касіскі як один із перших способів злому шифру;
- Використання статистичних методів аналізу;
- Застосування сучасних алгоритмів для дешифрування.

Переваги та недоліки шифру

1.Простота реалізації та використання;

2.Низька стійкість до криптоаналізу при коротких ключах;

3.Неможливість забезпечення надійного захисту інформації в сучасних умовах.

Шифр Віженера відноситься до симетричного шифрування (рис.2.8).

Рисунок 2.8. Симетричне шифрування

Симетричне шифрування - схема шифрування [17], у якій ключ шифрування та ключ дешифрування збігаються або один легко обчислюється з іншого та навпаки. Використовує один криптографічний ключ як для шифрування, так і для розшифрування. Простота використання одного ключа робить процес простим.

Переваги симетричного шифрування:

Симетричне шифрування має помітні переваги, насамперед своєю простотою. Використання одного ключа для шифрування і розшифрування спрощує процес. Ба більше, під час шифрування значних обсягів даних симетричне шифрування виявляється ефективним вибором. До додаткових переваг належать:

- Швидкість: алгоритми симетричного шифрування працюють значно швидше, ніж їхні асиметричні аналоги, про що ми ще поговоримо.
- Обчислювальна потужність: обчислювальні ресурси, необхідні для симетричного шифрування, порівняно нижчі.
- Мінімальний вплив на швидкість Інтернету: симетричне шифрування не чинить істотного впливу на швидкість передачі даних через Інтернет.

Процес шифрування:

1. Визначити ключ (наприклад, "KEY").
2. Повторити ключ, щоб він відповідав довжині відкритого тексту.
3. Для кожної букви відкритого тексту знайти відповідну букву з ключа.
4. Застосувати шифрування, використовуючи таблицю Віженера, де букви з відкритого тексту зміщуються на позицію, визначену буквою з ключа.

Процес дешифрування:

1. Використовувати той же ключ.
2. Повторити ключ, щоб він відповідав довжині зашифрованого тексту.
3. Для кожної букви зашифрованого тексту знайти відповідну букву з ключа.
4. Застосувати зворотне зміщення, щоб отримати відкритий текст.

2.3. Проект програмного забезпечення

Для проектування програмного забезпечення було використано об’єктно-орієнтовану методологію.

Об'єктно-орієнтоване проектування - це методологія проектування, що поєднує процес об'єктної декомпозиції і прийоми подання логічної і фізичної, а також статичної і динамічної моделей проектованої системи [18]. Використання об'єктно-орієнтованої методології зараз нерозривно пов'язано з використанням мови UML (Unified Modeling Language - уніфікована мова моделювання) , що являє собою систему позначень, яка базується на діаграмах і призначена для моделювання систем на основі об'єктно-орієнтованого підходу.

Об'єктно-орієнтоване проектування (ООП) відіграє ключову роль у сучасній розробці програмного забезпечення, принесши із собою значні зрушення у способі мислення про створення програм [18]. Запропоноване в 1960-х роках, ООП швидко стало домінуючим підходом у світі програмування.

Основна ідея ООП полягає в тому, щоб розглядати програму як сукупність об'єктів, які мають властивості та можуть взаємодіяти між собою. Кожен об'єкт є екземпляром певного класу і має власний стан та поведінку. ООП дозволяє моделювати реальний світ у програмі, що спрощує розробку та робить код більш зрозумілим і повторно використовуваним.

Основні принципи ООП - інкапсуляція, наслідування та поліморфізм - на яких ґрунтується об'єктно-орієнтований підхід. Інкапсуляція дозволяє об'єднати дані та методи, які їх обробляють, в один об'єкт, наслідування дозволяє створювати нові класи на основі існуючих і поліморфізм дозволяє об'єктам реагувати на методи відповідним чином залежно від контексту виклику. Вивчення та розуміння цих принципів дозволяє розробникам писати більш читабельний, ефективний та підтримуваний код, що є ключем до успішної розробки програмного забезпечення.

Використання мови моделювання UML є однією з ключових складових об'єктно-орієнтованого проектування та сприяє вдосконаленню процесу розробки програмного забезпечення.

Мова моделювання є потужним інструментом для візуалізації, структурування та опису системного проектування в об'єктно-орієнтованій парадигмі. Використання UML дозволяє розробникам чітко уявити архітектуру та взаємозв'язки між компонентами системи, а також детально описати її функціональність та поведінку. Завдяки різноманітним типам діаграм UML, таким як діаграми класів, послідовностей та варіантів використання, розробники можуть ефективно спілкуватися між собою та з зацікавленими сторонами, покращуючи розуміння, документацію та якість програмного забезпечення в цілому.

Приклади практичного застосування об'єктно-орієнтованого проектування широко представлені в різних галузях розробки програмного забезпечення такі як:

- Системи управління проектами

У програмах для управління проектами, таких як Jira або Trello, ООП дозволяє створювати об'єкти для представлення завдань, проектів, користувачів та команд. Кожен об'єкт може мати свої властивості та методи для управління ним.

- Інтернет-магазини

У веб-розробці для створення інтернет-магазинів використовується ООП для моделювання товарів, замовлень, користувачів та корзини покупок. Кожен товар може бути представлений як об'єкт класу «Товар», а кожен замовлення - як об'єкт класу «Замовлення».

- Графічні редактори

У графічних редакторах, таких як Adobe Photoshop або GIMP, ООП використовується для моделювання об'єктів, таких як шари, фігури та тексти. Кожен об'єкт на полотні може бути представлений як окремий об'єкт класу з відповідними методами для редагування та маніпулювання.

- Моделювання транспортних систем

У програмах для моделювання транспортних систем, таких як Simulink або AnyLogic, ООП дозволяє створювати об'єкти для представлення транспортних засобів, маршрутів та пасажирів. Кожен засіб транспорту може бути представлений як об'єкт класу «Транспорт», а кожен пасажир - як об'єкт класу «Пасажир».

Ці приклади ілюструють широкий спектр застосувань об'єктно-орієнтованого проектування у різних сферах індустрії та розробки програмного забезпечення.

2.4 Ескізний проект

Проект ескізу - це етап, коли проект програмного забезпечення починається і представляє результати зовнішнього дизайну програмного

забезпечення. Її мета - створити концепцію загального плану чи розвиненої програми. Під час цього етапу технічні або впроваджувальні аспекти програми зазвичай не детально описані. Натомість спостерігається визначення функцій, інтерфейсу користувача та основних особливостей програми.

Результати проекту ескізу включають такі елементи:

− Опис функціональності – визначення основних функцій та можливостей програми. Він включає список основних функцій, які програма повинна виконувати, та короткий опис кожної з них.

− Прототип інтерфейсу користувача – створення простого прототипу інтерфейсу користувача, який демонструє, як програма буде взаємодіяти з користувачем. Це може бути малюнок або макет екранів програми.

− Визначення основних модулів – ідентифікація основних компонентів або модулів, які будуть потрібні для реалізації функціональності програми. Це допомагає зрозуміти, як програма буде організована і які будуть основні компоненти.

Загалом, ескізні проекти дають загальні уявлення про те, як виглядає програмне забезпечення, і визначають його основні вимоги. Це важливий крок для забезпечення розвитку програми в правильному напрямку з самого початку.

Мова моделювання UML була обрана при розробці цього програмного забезпечення.

2.4.1 Контекстна діаграма

Контекстна діаграма - це початкова діаграма моделі, що характеризує зв’язки системи з навколишнім середовищем [19].

Контекстна діаграма для програмної системи представлена на рисунку 2.9.

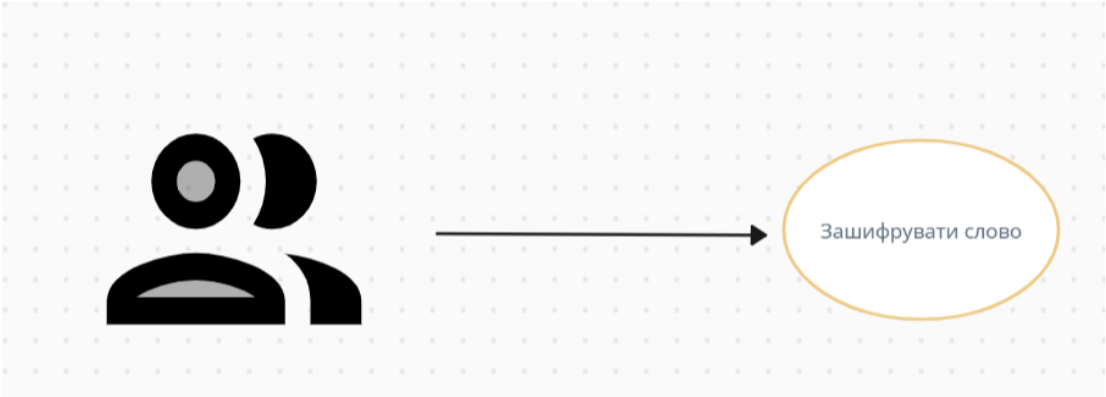

Рисунок 2.9. Контекстна діаграма

2.4.2 Діаграма варіантів використання

Діаграма варіантів використання – діаграма, на якій зображено відношення між акторами та прецедентами в системі. Діаграма варіантів використання для програмної системи представлена на рисунку 2.10**.**

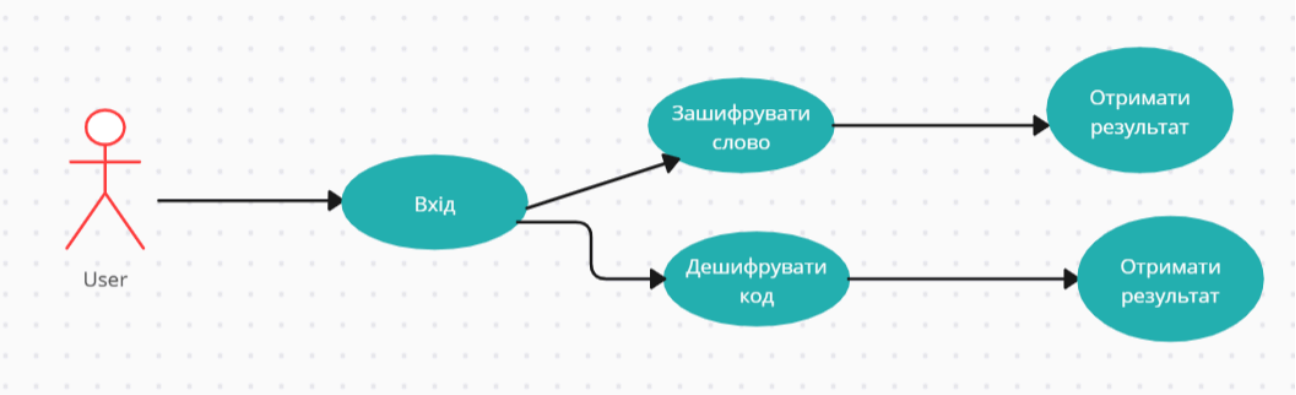

Рисунок 2.10. Діаграма варіантів використання для програмної системи

Розглянемо детально кожний варіант використання, надаючи опис потоків подій. Специфікації варіантів використання представлено в таблицях 2.2-2.6.

Таблиця 2.2

Специфікація прецеденту «Входу»

Вхід	
Складова	**Опис**
Короткий опис	Даний варіант використання дозволяє користувачу зайти в системі.
Передумови	Не визначені
Основний потік подій	Користувач ініціює відкриття інтерфейсу входу. Система відкриває інтерфейс програми. Користувач вибирає що йому необхідно зроби, а самезашифрувати слово чи дешефрувати код.
Альтернативний потік подій	Система повідомляє користувачу, про невалідність введених даних. Дійовій особі пропонується можливість повторити введення або завершити варіант використання.
Постумови	Не визначені

Таблиця 2.3

Специфікація прецеденту «Зашифрування слів»

Шифрування слів	
Складова	**Опис**
Короткий опис	Даний варіант використання дозволяє користувачу зашифрувати слово
Передумови	Виконано прецедент «Входу ».
Основний потік подій	Користувач ініціює відкриття зашифрування слова. Система запускає в дію код програмиза яким здійснюється зашифруваня слова.Системавидає кінцевий результат,а саме код заашифрованого слова. Результати операції відображається на інтерфейсі.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

Таблиця 2.4

Специфікація прецеденту «Дешифрування коду»

Дешифрування слова	
Складова	**Опис**
Короткий опис	Даний варіант використання дозволяє користувачу дешифрувати код
Передумови	Виконано прецедент «Входу ».
Основний потік подій	Користувач ініціює відкриття дешифруваннння. Система запускає в дію код програмиза яким здійснюється дешифрування коду.Системавидає кінцевий результат,а саме словояке булозаашифроване. Результати операції відображається на інтерфейсі.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

Таблиця 2.5

Специфікація прецеденту «Отримання результату»

Отримання результату після шифрування слова	
Складова	**Опис**
Короткий опис	Даний варіант використання дозволяє користувачу отримане певний код за допомогою якого можна буде отримати зашифроване слово.
Передумови	Виконано прецедент «шифрування слова».
Основний потік подій	Системавидає кінцевий результат,а саме код заашифрованого слова . Результати операції відображається на інтерфейсі.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

Таблиця 2.6

Специфікація прецеденту «Отримання результату»

Отримання результату після шифрування слова	
Складова	**Опис**
Короткий опис	Даний варіант використання дозволяє користувачу отримати зашифроване слово замість певного коду .
Передумови	Виконано прецедент «дешифрування слова ».
Основний потік подій	Системавидає кінцевий результат, а саме код зашифрованого слова. Результати операції відображається на інтерфейсі.
Альтернативний потік подій	Не визначено.
Постумови	Не визначені.

2.4.3 Діаграма станів

Діаграма станів – це діаграма, щопоказує, як об'єкт переходить з одного стану в інший [20]. Діаграма станів служить для моделювання динамічних аспектів системи, також корисна при моделюванні життєвого циклу об'єкта. Від інших діаграм діаграма станів відрізняється тим, що описує процес зміни станів об'єкта, поведінка якого характеризується його реакцією на зовнішні події.

Перехід між підсистемами здійснюватиметься через головне вікно, частина якого буде активна незалежно від обраної підсистеми і інших об’єктів конфігурації.

Діаграма станів для програмної системи представлена на рисунку 2.11.

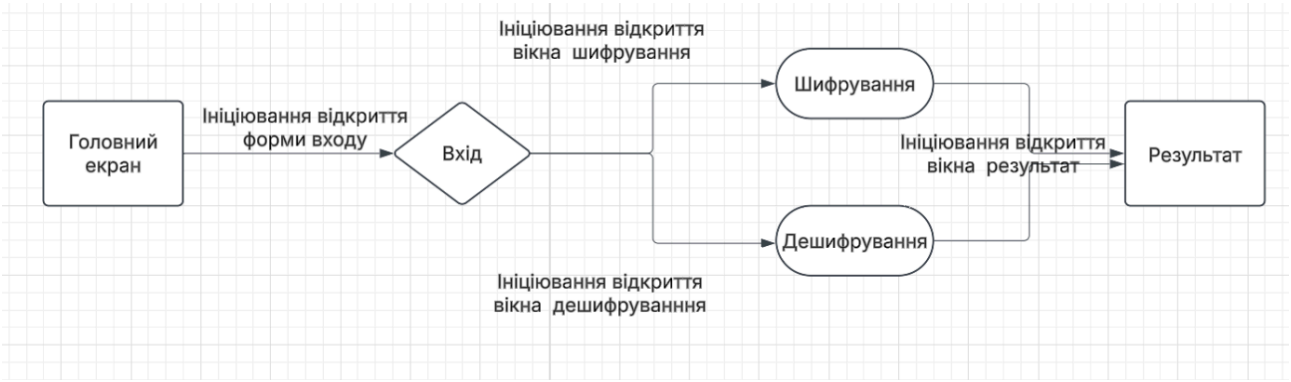

Рисунок 2.11. Діаграма станів для переходу між формами

2.5 Проектування інтерфейсу

Інтерфейс користувача – засіб зручної взаємодії користувача з інформаційною системою; сукупність засобів для обробки та відображення інформації, максимально застосованих для зручності користувача [21].

Макет головного вікна представлений на рисунку 2.12.

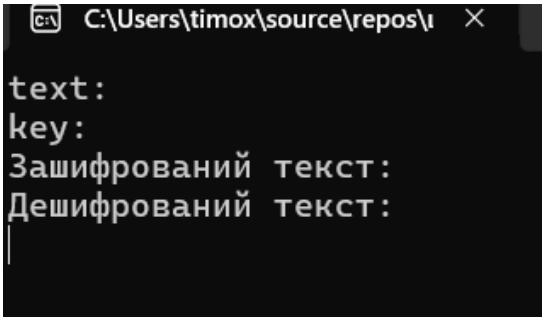

Рисунок 2.12. Макет головного вікна

Макет діалогового вікна “Key”представленийна рисунку 2.13.

Рисунок 2.13. Макет діалогово вікна “Key”

Макет діалогово вікна “Зашифрований текст” представлений на рисунку 2.14.

Рисунок 2.14. Макет діалогового вікна з зашифрованим текстом

Макет текстового документа дешифрованого тексту представлений на рисунку 2.15.

Рисунок 2.15. Макет діалогового вікна з дешифрованого тексту

2.6 Робочий проєкт

В якості засобу розробки дипломного проекту було обрано середовище Visual Studio.

Visual Studio — це потужне середовище розробки, яке широко використовується для створення програмного забезпечення різного рівня складності [22]. Воно підтримує безліч мов програмування, таких як C#, C++, Python, JavaScript тощо, що робить його універсальним інструментом для дипломних проєктів.

Visual Studio — це сучасне середовище розробки, оптимізоване для вирішення типових завдань, пов’язаних зі створенням застосунків для різних платформ, включаючи Windows, Android та iOS. Воно містить вбудовані засоби, що полегшують тестування програм на сумісність із різними версіями операційних систем, а також інструменти для проектування застосунків, які адаптуються до пристроїв із різною роздільною здатністю екрана (планшети, смартфони, ноутбуки тощо).

2.6.1 Обґрунтування вибору інструментарію

У процесі розробки програмного забезпечення важливим етапом є вибір відповідного інструментарію, що забезпечить ефективну реалізацію

поставлених завдань. Для створення застосунку було обрано середовище розробки Visual Studio, яке надає широкий спектр можливостей для розробки, тестування та налагодження програмного забезпечення.

Основними причинами вибору Visual Studio є:

- Широка підтримка мов програмування (C#, C++, Python, JavaScript тощо), що дає змогу адаптувати проєкт під різні вимоги.
- Інтегровані засоби налагодження та тестування, які дозволяють виявляти та усувати помилки на ранніх етапах розробки.
- Засоби роботи з Git та іншими системами контролю версій, що спрощує командну розробку.
- Гнучкість та можливість розширення функціоналу за допомогою плагінів і додаткових бібліотек.
- Зручне середовище для кросплатформної розробки, що дозволяє створювати застосунки для Windows, Android, iOS та інших платформ.

Використання Visual Studio сприяє підвищенню продуктивності розробки та забезпечує ефективний контроль за якістю програмного продукту.

Альтернативи та порівняння. Серед інших можливих середовищ розробки розглядалися IntelliJ IDEA, Android Studio, Eclipse та інші. Однак Visual Studio було обрано через його широку функціональність, стабільність роботи та підтримку кросплатформної розробки.

Вибір Visual Studio як основного середовища розробки зумовлений його потужним функціоналом, гнучкістю та зручністю. Це дозволяє ефективно реалізувати поставлені завдання та забезпечити високу якість кінцевого продукту.

Критерії обрання мови програмування. При виборі мови програмування для розробки програмного забезпечення необхідно враховувати низку критеріїв, які впливають на продуктивність, гнучкість і масштабованість проєкту. У мойомувипадку було обрано C#[23], виходячи з наступних факторів:

1. Продуктивність і швидкодія

C# є однією з найшвидших мов програмування завдяки ефективному використанню пам’яті та низькорівневому управлінню ресурсами.

Забезпечує високий рівень оптимізації коду, що важливо для ресурсомістких програм.

2. Гнучкість і можливість низькорівневого програмування

Дозволяє працювати як із високорівневими абстракціями, так і безпосередньо з пам’яттю та апаратним забезпеченням.

Використовується для розробки системного програмного забезпечення, драйверів, ігор, вбудованих систем тощо.

3. Об’єктно-орієнтоване програмування (ООП)

Підтримує парадигму ООП, що забезпечує модульність, повторне використання коду та спрощує підтримку проєкту.

Крім ООП, підтримує процедурний та узагальнений підходи до програмування.

4. Переносність і кросплатформність

Код на C# можна компілювати для різних операційних систем (Windows, Linux, macOS, Android, iOS тощо).

Велика кількість компіляторів (GCC, Clang, MSVC), що дозволяє адаптувати код під різні платформи.

5. Розвинена екосистема та бібліотеки

Наявність великої кількості стандартних (STL) і сторонніх бібліотек, що полегшують розробку.

Підтримка багатьох фреймворків для створення графічних інтерфейсів, обробки даних, мережевого програмування (Qt, Boost, OpenCV тощо).

З огляду на ці критерії, C# є оптимальним вибором для реалізації могопроєкту, оскільки забезпечує високу продуктивність, гнучкість у розробці та можливість створення ефективних програмних рішень.

У розділі було проведено аналіз можливих технологій, середовищ розробки та інструментів для реалізації дипломного проєкту. Було визначено, що Visual Studio є оптимальним вибором завдяки своїм широким можливостям, підтримці кросплатформної розробки, гнучкості та інтеграції із сучасними інструментами DevOps.

Обраний стек технологій та засоби розробки забезпечать ефективну реалізацію поставлених завдань Вибір відповідного інструментарію також полегшить тестування, підтримку та подальше вдосконалення застосунку.

На основі проведеного аналізу можна переходити до безпосередньої розробки програмного забезпечення, включаючи проєктування архітектури, реалізацію основних модулів та їх тестування.

Розділ 3. РЕАЛІЗАЦІЯ ПРОЄКТУ

Результатом розробки проекту є програма для шифруваня та дешифрування слів методом шифрування Віженера. Дане програмне забезпечення являє собою програму, яку розроблено для Windows. Для досягнення поставленої цілі виконано аналіз предметної області, в якому було досліджено роботу інших програм.

В ескізному проекті побудовано контекстну діаграму, діаграму варіантів використання, концептуальну модель, діаграми переходів станів, розроблено ескізи користувацького інтерфейсу.

В технічному проекті побудовано логічну модель бази даних, розроблено діаграму послідовності, пакетів та класів для варіантів використання програмного забезпечення. В робочому проекті виконано та аргументовано вибір інструментів.

Під час реалізації алгоритму шифру Віженера було використано мову програмування C#, що дозволило забезпечити ефективне виконання операцій шифрування та дешифрування.

Код для програми було написано на універсальній мові C#.В середовищі Visual Studio.

Visual Studio — це потужне середовище розробки, яке широко використовується для створення програмного забезпечення різного рівня складності [22]. Воно підтримує безліч мов програмування, таких як C#, C++, Python, JavaScript тощо, що робить його універсальним інструментом для дипломних проєктів [22].

Visual Studio — це сучасне середовище розробки, оптимізоване для вирішення типових завдань, пов’язаних зі створенням застосунків для різних платформ, включаючи Windows, Android та iOS. Воно містить вбудовані засоби, що полегшують тестування програм на сумісність із різними версіями

операційних систем, а також інструменти для проектування застосунків, які адаптуються до пристроїв із різною роздільною здатністю екрана (планшети, смартфони, ноутбуки тощо).

В результаті розробки було реалізовано функціонал входу в саму програму, та її складову.

В результаті розробки було реалізовано наступний функціонал:

- Шифрування тексту за алгоритмом Віженера;
- Дешифрування тексту при відомому ключі;

Основний функціонал реалізований у вигляді наступних модулів:

1.Шифрування тексту;

2.Дешифрування тексту;

3.Результат операції.

3.1 Інтерфейс програми

Головне вікно (рис 3.1.) – це вікно є вітальним екраном, який надає користувачам можливість увійти в саму програму та вибрати метод шифрування та мати можливістьвибрати дію, а саме шифрувати словоабо дешифрувати код та отримати певний результат.

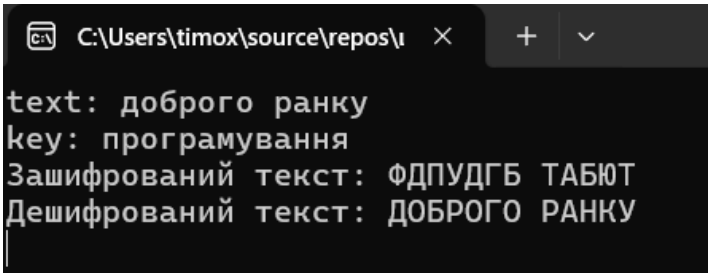

Рисунок 3.1 Головне вікно

Головне вікно є початковою точкою взаємодії користувачів з програмою і має інтуїтивно зрозумілий дизайн, що сприяє зручному входу та добре виконує

свою функцію, забезпечуючи простий та зручний доступ до основних можливостей програми.

Вікно має дві кнопки:

- Шифр Віженера – це метод шифрування за допомогою якого буде шифрувати певне слово;
- Ключ– це зашифроване слово, яке нам потрібно щоб зашифрувати або дешифрувати певний текст ;
- Зашифрувати – це функціяза допомогою якою ми можемо зашифрувати слово та отримати результат
- Дешифрувати – це функціяза допомогою якою ми можемо дешифрувати певний код та отримати результат

Кнопка “Метод шифрування” (рис 3.2) - наступний важливий елемент додатку, є функція яка частиною процесушифрування коду користувача, коли користувачі повинні вибрати певний метод шифрування,щоб користувач який зайшов у програму та міг вибрати певний метод шифрування . Дизайн форми простий і зосереджений на основних елементах, необхідних для виконання функції шифрування коду .

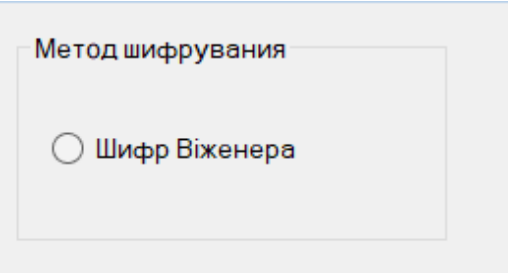

Рисунок 3.2 Метод шифрування

Ще одна дуже важлива опція, це є кнопка введення тексту (Рис.3.3).

Вона має такі функції у програмі :

1. Користувач вводить певний текст який йому потрібно зашифруватичерез спеціальне текстове поле.

2. Користувач вводить ключ через спеціальне текстове поле.

3. Програма перевіряє коректність введених символів (допустимі лише літери).

4. Після натискання кнопки Enter, введене слово використовується для шифрування або дешифрування.

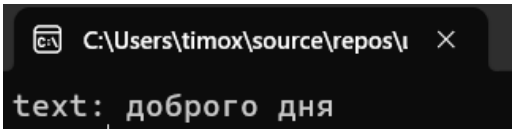

Рисунок 3.3 Діалогове вікно “ введення ключового слова”

Діалогове вікно “Ключ” (рис.3.4) - наступний важливий елемент додатку призначене для введення секретного ключа, який є частиною процесу шифрування коду користувача, коли користувачі повинні ввести певний засекречений ключ, щоб отримати результат. Дизайн форми простий і зосереджений на основних елементах, необхідних для виконання функції шифрування коду.

Рисунок 3.4 Діалогове вікно “Ключ

Діалогове вікно ”Зашифруваний текст” (рис 3.4) – наступний важливий елемент додатку, є функція яка частиною процесу шифрування коду користувача, коли користувачі повинні ввести певне слово, щоб отримати результат це певні букви. Дизайн форми простий і зосереджений на основних елементах, необхідних для виконання функції шифрування коду .

Рисунок 3.4 діалогове вікно “Зашифрований текст”

Діалогове вікно ”Дешифрований текст ” (рис 3.5) – наступний важливий елемент додатку, є функція яка є частиною процесу дешифрування певних букв користувача, коли користувачі повинні ввести певний набір букв, щоб отримати результат у вигляді загаданого слова. Дизайн форми простий і зосереджений на основних елементах, необхідних для виконання функції дешифрування коду .

Рисунок 3.5 діалогове вікно “Дешифрований текст”

Детальніше результат роботи представлений у вигляді коду в додатку Б.

Для розробки додатку було використано наступні інструменти:

- C# – об’єктно-орієнтована мова програмування.
- Visual Studio – це потужне середовище розробки, яке широко використовується для створення програмного забезпечення різного рівня складності. Воно підтримує безліч мов програмування, таких як C#, C++, Python, JavaScript тощо.

3.2 Тестування та результати розробки програмного забезпечення

Перейдемо до першої перевірки нашої програми, за допомогою якої можна буде шифрувати та дешифрувати слова за методом шифрування Віженера.

Отже, будемо шифрувати та дешифрувати простий вираз “Доброго ранку”, ключом буде слово програмування ,отже що в нас вийшло :

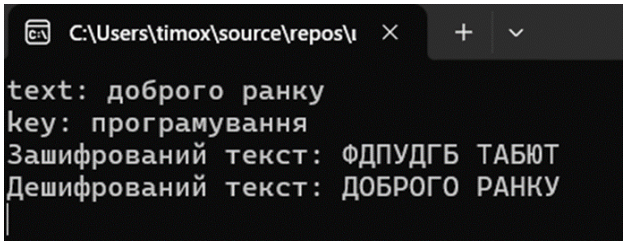

Рисунок 3.6 Діалогове вікно «Результат»

Отже, програма шифрує та дешифрує прості вирази.

Проведемо другу перевірку, де наша програма буде шифрувати та дешифрувати певний текст який складеться з 100 слів.

Отже текст :

Над широкими степами , над вузеньким струмком сонце сходить рано. Птахи щебечуть, пробуджуючи все довкола. Легкий подих бризу колише верхівки дерев. Люди крокують стежками, думаючи про нову добу.

Запускаємо програму , вводимо наш текст, ключем буде вираз: “мова – це скарб.

Ось, що вийшло з перевірки :

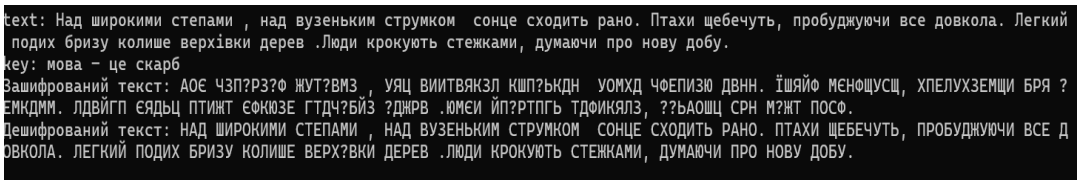

Рисунок 3.7 Діалогове вікно ”Результат”

Перейдемо до третьої перевірки програми, а саме будемо шифрувати слово “сад”, а ключем буде слово “код”. Отже, розпочнемо перевірку програми.

Перший етап перевірки програми:

Запуск програми та водимо наше задумане слово.

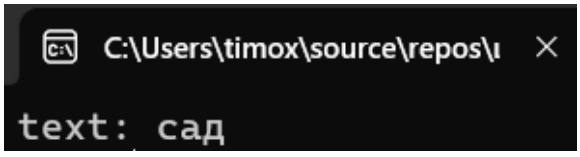

Рисунок 3.8 Діалогове вікно ”Слово”

Другий етап перевірки:

Задаємо ключ за допомогою якого можемо дізнатися зашифрований текст нашого слова.

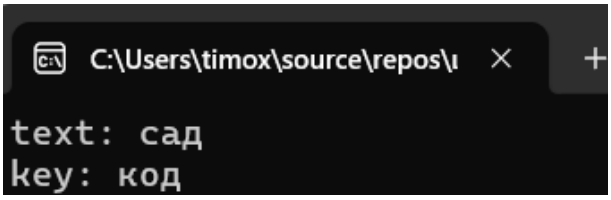

Рисунок 3.9 Діалогове вікно”Ключ ”

Третій етап дізнаємося зашифрований текст нашого слова.

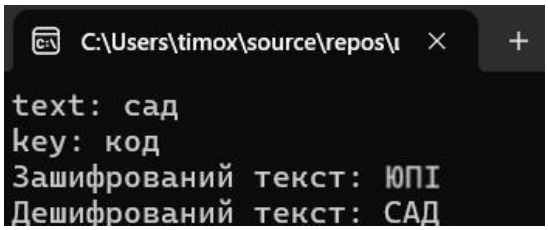

Рисунок 3.10 Діалогове вікно”Результат”

Отже, бачимо якщо наше слово “сад”, а ключ “код”, то виходить зашифрований текст “ЮПІ”.

Розпишемо процес кодування кожної букви:

Наший український алфавіт містить 33 букви (А-1,Б-2,В-3,…,Я-33).

Таблиця 3.1

” Кодування”

Буква тексту	Позиція	Буква ключа	Позиція	Сума	Нова буква
С	20	К	12	32	Ю
А	1	О	16	17	П
Д	5	Д	5	10	І

Розбір для кожної букви:

С + К → (20+12) = 32 →Ю

А + 0 → (1+16) = 17 →П

Д + Д → (5+5) = 10 →І

Висновок: Текст "САД" при шифруванні ключем "КОД" перетворюється на "ЮПІ". З цього моменту можемо сказати, що програма працює бездогано та виконує задачі, які ми поставили перед нею.

3.3 Основні функціональні можливості

1. Шифрування тексту – можливість введення тексту користувачем та його подальше перетворення за допомогою алгоритму Веженера. Користувач може задавати ключ шифрування або використовувати згенерований автоматично.
2. Дешифрування тексту – відновлення початкового тексту за умови введення правильного ключа. Декодування відбувається згідно з алгоритмом, що використовується у шифруванні.
3. Візуалізація процесу шифрування – у разі потреби користувач може переглянути поетапний процес шифрування/дешифрування, що дозволяє краще зрозуміти принцип роботи алгоритму.
4. Підтримка декількох мов – можливість вибору мови інтерфейсу, що робить програмне забезпечення доступним для ширшого кола користувачів.
5. Підтримка різних форматів введення та виведення – користувач може працювати з текстовими файлами різних форматів, що підвищує зручність використання.

Отже, програму можна вважати універсальною та швидкою, а головне користуватися цієї програмою можна навіть без інтернету.

3.4 Аналіз аналогічного програмного забезпечення

Порівняння розробленого програмного забезпечення з подібними рішеннями може висвітлити переваги та недоліки. Основними критеріями

порівняння були швидкість роботи, зручний для користувачів, рівень безпеки та можливість налаштування вимог користувачів.

Переваги розробленого програмного забезпечення:

- Висока швидкість шифрування та дешифрування - Алгоритми, оптимізовані для ефективної роботи, навіть при великій кількості даних.
- Інтуїтивно зрозумілий інтерфейс - Програма була розроблена з урахуванням базовим вміням користувачів.
- Можливості програмипрацювати без підключення до Інтернету - забезпечує самостійність використання.
- Кросплатформеність – підтримка роботи на Windows, Linux та macOS.
- Масштабованість – можливість подальшого розширення функціоналу.

3.5 Тестування та оцінка роботи програми

Проведене тестування програмного забезпечення включало такі види тестів:

- Функціональне тестування – перевірка коректності роботи основних функцій.
- Юзабіліті-тестування – оцінка зручності інтерфейсу.
- Навантажувальне тестування – перевірка стабільності роботи програми при великих обсягах даних.
- Кросплатформне тестування – перевірка роботи на різних операційних системах.

Висновок : результати тестування підтвердили коректність роботи програмного забезпечення та відповідність технічним вимогам.

3.6 Висновки до програми

Це програмне забезпечення було розроблено в рамках проекту для шифрування текстових даних за допомогою **Windows Form**. Основна мета

створення програми полягала в забезпеченні безпеки інформації за допомогою простих, але ефективних методів шифрування, які ускладнюють перехоплення та розшифровку даних третіми сторонами.

Windows Form- інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft.NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за допомогою створення обгортки для Win32 API в керованому коді**.** [24]

Розроблене програмне забезпечення успішно реалізує завдання та має зручний інтерфейс, забезпечуючи високий рівень ефективності роботи. Аналіз тесту підтвердив стабільність системи та відповідність визначеним вимогам.

У процесі розробки було:

1. Реалізовано механізм введення вихідного тексту користувачем та задання ключа шифрування.
2. Побудовано алгоритм шифрування та дешифрування тексту за принципами шифру Віженера з урахуванням українського алфавіту.
3. Забезпечено обробку випадків невідповідності довжини ключа шляхом його циклічного повторення.
4. Оформлено інтуїтивно зрозумілий графічний інтерфейс з кнопками для шифрування, розшифрування та очищення полів.
5. Проведено тестування програми на прикладах реального тексту, що підтвердило коректність роботи алгоритмів і відсутність втрат інформації при шифруванні та дешифруванні.

Результати тестів показують, що програма працює правильно з короткими словами та довгими текстовими повідомленнями. Шифрування робиться швидко та надійно, навіть якщо налаштована основна конфігурація комп'ютера.

Головними перевагами розробленого рішення є:

- Простота використання та мінімальні вимоги до навичок користувача.
- Швидкість обробки інформації.

Тому завдання розробки надійного, функціонального та комфортного методу шифрування текстових даних успішно виконано. Надалі потенціал для розробки проекту - інтеграція програми з іншим інформаційним захистом, підтримуючи підтримку алфавіту та впровадження нових методів шифрування.

На основі проведеного аналізу можна виділити кілька напрямків для подальшого вдосконалення:

1. Оптимізація алгоритму - здатність покращувати продуктивність за допомогою більш ефективних математичних моделей.
2. Розширення підтримки мобільних платформ – створення додатків для iOS та Android для підвищення доступності програми.
3. Підтримка хмарних технологій - це можливість зберігання та обміну зашифрованими файлами через захищений сервер.
4. Інтеграція в інші програмні продукти, наприклад, можливість використання програмного забезпечення у веб -браузері або інтегрувати його в систему захисту даних вашої компанії.

Отже, поточна версія програмного забезпечення є повноцінним робочим продуктом, який може бути використаний для шифрування даних, а подальше вдосконалення дозволить значно розширити його можливості та покращити користувацький досвід. Також програмне забезпечення ефективно захищає дані у хмарних сервісах від атак типу "людина посередині". Процес розробки реалізував основні механізми шифрування, аутентифікації та визнання аномальної діяльності в мережевому трафіку. Програма характеризується своїм комфортним інтерфейсом, швидкісною обробкою та надійністю захисту інформації. Результати тестів підтверджують ефективність розвиненого рішення.

Розділ 4. ЕРГОНОМІКА ІТ ТА ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ

4.1 Вимоги до програмного забезпечення та основні підходи до його проектування з погляду користувача.

Шифрування даних є одним із ключових компонентів безпеки в сучасних інформаційних системах. Це передбачає перетворення даних таким чином, щоб сторонні люди не могли читати без належного ключа дешифрування. Тому проект програмного забезпечення для шифрування повинно враховувати багато вимог та принципів для задоволення потреб користувачів та в той же час для забезпечення ефективності, надійності та зручності.

Вимоги до програмного забезпечення для шифрування:

1. **Безпека:** головна вимога програмного забезпечення для шифрування. Криптографічні алгоритми повинні бути стійкими до нападів криптоаналітичних атак та різних атак на напади, особливо через чутливість до алгоритмів. Ключі шифрування повинні бути забезпечені проти несанкціонованого доступу.
2. **Сумісність:** Програмне забезпечення для шифруванняповинно бути сумісним з різноманітними операційними системами та платформами. Таким чином, користувачі можуть обмінюватися зашифрованими даними між різними пристроями, такими як комп’ютери, смартфони та сервери.
3. **Простота у використанні:** шифрування - це технічний процес, тому важливо, щоб користувачі могли легко налаштувати та використовувати програмне забезпечення. Інтерфейс повинен бути інтуїтивним, а процес шифрування повинен бути простим і швидким.
4. **Відповідність стандартам**: Програмне забезпечення для шифрування повинно відповідати міжнародним стандартам безпеки, таких як

AES (розширене стандарт шифрування) для симетричного шифрування та RSA для асиметричного шифрування. Це підтвердить обраний метод і забезпечить впевненість у тому, що ваші дані належним чином захищені.

5. **Масштабованість:** Програмне забезпечення повинно мати можливість обробляти великі кількості даних без зниження швидкості роботи.

Основні підходи до проектування програмного забезпечення з погляду користувача :

- **Модульність і гнучкість:** Програмне забезпеченняповинно бути розділене на незалежні модулі, щоб компоненти можна було регулювати або змінювати, не впливаючи на інші частини системи.
- **Автоматизація процесу:** Автоматизація процесу шифрування та дешифрування є важливим аспектом, оскільки багато користувачів не хочуть налаштувань ручного шифрування. Сюди входить автоматичне шифрування всіх даних, коли ви зберігаєте або обмінюєте зашифровані повідомлення на дотику кнопки.
- **Висока продуктивність:** Оскільки шифрування є інтенсивним ресурсом, програмне забезпечення повинно бути оптимізоване для швидких процесів з великою кількістю даних, одночасно забезпечуючи низьку затримку та мінімальні апаратні вимоги.
- **Кросплатформенність**: Щоб забезпечити максимальну зручність для користувачів, програмне забезпечення повинно працювати над різними операційними системами (Windows, MacOS, Linux, Android, iOS) та на різних пристроях. Це дозволяє забезпечити безпечний доступ до зашифрованих даних у будь -якому середовищі.

Отже, можемо зробити висновок що розробка програмного забезпечення для шифрування даних - це складний процес, який вимагає як технічних аспектів безпеки, так і зручності для кінцевих користувачів. Враховуючи вимоги безпеки, доброзичливість користувача, сумісність та інтеграція з іншими

системами є ключовими елементами ефективного та надійного обладнання для шифрування. В результаті такого підходу користувачі отримують не лише надійні інструменти захисту даних, але й зручне та інтуїтивне програмне забезпечення.

4.2 Ергономічні цілі і показники якості програмного продукту

Ергономічні особливості програмного продукту важливі для користувачів використовувати програмне забезпечення зручним, ефективним та безпечним. У сучасному світі, оскільки інтерфейси стають складнішими, а кількість інформації зростає, важливо не лише забезпечити функціональність програмного продукту, а й для того, щоб зробити його доступним, зрозумілим та зручним для користувачів. Це основа програмного забезпечення ергономіки.

Ергономічні цілі та показники якості програмного продукту є ключовими факторами для створення програмного забезпечення, яке буде не лише функціональним, а й зручним для користувача. Вони охоплюють всі аспекти взаємодії користувача з продуктом — від зручності та ефективності до доступності та комфорту. Врахування цих вимог дозволяє мінімізувати помилки, підвищити задоволеність користувачів і забезпечити високий рівень продуктивності. Тому проектування програмного забезпечення, яке відповідає ергономічним стандартам, значно підвищує його конкурентоспроможність на ринку та забезпечує позитивний досвід користувачів.

4.3 Основні характеристики, що враховуються при розробці інтерфейсу користувача

Розробляючи інтерфейс користувача для програмного забезпечення, який забезпечує шифрування даних, важливо отримати основні функції, такі як послідовність, інтуїція, естетичний вигляд, ефективність та доступність проектів усіх користувачів. Вимога щодо зручності та комфорту полягає у

створенні інтерфейсу, який можна легко сприймати та використовувати з мінімальним когнітивним стресом та позитивними емоційними враженнями, оскільки користувачам потрібно проводити складні процеси шифрування, які потребують високого рівня фокусування та уваги.

Прототип інтерфейсу повинен враховувати такі проблеми, як неправильне розміщення елементів, несправні функції та недостатня візуалізація стану шифрування. Це особливо важливо в контексті безпеки даних, коли кожен процес повинен бути максимально зрозумілим для користувача. Тому прототипи потребують ретельного тестування користувачів для виявлення та виправлення недоліків.

Принципи впровадження інтерфейсуповинні бути отримані шляхом надання чіткого зворотного зв'язку статусу процесу шифрування (наприклад, метрики прогресу), використовуючи чітку мову, зручні елементи керування, такі як кнопки для ініціювання шифрування та дешифрування. Важливим аспектом є зменшення стресу та непорозуміння, надаючи користувачам інформацію про можливі помилки або успішне завершення процесів.

Процес інтерфейсу та вимоги до проектування компонентів включають детальну розробку макета, реальне тестування користувачів та вдосконалення на основі отриманих оглядів. У цьому контексті важливо не тільки перевірити функціональність, але й перевірити прозорість кожного користувача, особливо при взаємодії з програмою, коли ви намагаєтесь керувати клавішами шифрування та перевіряти цілісність зашифрованих даних.

Компоненти шифрування інтерфейсувключають вибір методу шифрування для розробки та впровадження компонентів у програмне забезпечення, створюючи інтуїтивно видалені елементи, такі як кнопки, форми та вибір. Усі ці компоненти повинні бути інтегровані в загальний дизайн, враховуючи потреби користувачів. Це знижує ризик помилок і підвищує ефективність роботи

програми. Важливо полегшити користувачам проводити всі необхідні процеси, які їм потрібні, з мінімальними зусиллями та високим рівнем безпеки даних.

4.4 Техніко-економічне обґрунтування розробки

Успішне впровадження проекту шифрування проекту вимагає чіткого бачення розробки вибраних ідей та детального планування для кожного етапу. Цей пристрій містить детальний опис проекту. Оцінимо технічний потенціал програми, зрозумійте конкурентні переваги в галузі безпеки даних, розробити ефективні стратегії доставки користувачів та визначити найважливіші аспекти, які допоможуть забезпечити фінансову стабільність проекту.

Вибір методу шифрування повинен враховувати його ефективність та безпеку та забезпечити зручний та інтуїтивний інтерфейс. Оцінка ризику передбачає виявлення можливої чутливості до безпеки та розробки стратегій для мінімізації періодичних оновлень програмного забезпечення та вдосконалення алгоритмів шифрування. Страхування проекту включає створення механізму резервного копіювання даних та забезпечення конфіденційності на всіх етапах обробки.

4.4.1 Резюме проекту

Назва: Шифрування методом Веженера .

Місце розташування: програмадоступна для комп’ютерів та ноутбуків.

Мета: забезпечити високий рівень безпеки та конфіденційності даних користувачів за допомогою надійного шифрування методом Веженерата захисту персональних даних. Це досягається шляхом введення ефективного та простого інструменту, повідомлень та інших даних на ноутбуці. Ця програма надає користувачам можливість забезпечити повну конфіденційність при збереженні та передачі, зменшуючи ризик персональних даних та розширення довіри до цифрових технологій.

Суть проекту: Проект передбачає розробку програмного забезпечення для шифрування даних, яке автоматизує процес захисту особистої інформації

користувачів, забезпечуючи надійне та зручне шифрування повідомлень та інших даних. Програмний продукт пропонує широкий спектр функціональних можливостей для забезпечення безпеки та конфіденційності:

1. **Забезпечення безпеки даних**: Шифрування важливої інформації для запобігання несанкціонованому доступу.
2. **Простота використання:** Створення інтуїтивно зрозумілого інтерфейсу для користувачів з будь-яким рівнем технічної підготовки.
3. **Доступність на різних платформах:** Підтримка основних операційних систем для ноутбуків (Windows, macOS, Linux).
4. **Швидкість і ефективність процесів шифрування/дешифрування:** Оптимізація процесів для мінімізації часу, необхідного для обробки великих обсягів даних.
5. **Захист конфіденційності**: Гарантування того, що дані не будуть збережені на серверах і не будуть доступні стороннім особам.

Отже, завдяки цим функціях програмабуденадійним інструментом для користувачів, які хочуть захистити свої дані від потенційних загроз.

Переваги програми :

1. Шифрування файлів та повідомлень зменшує ймовірність порушення конфіденційних даних, що забезпечує більшу конфіденційність.
2. Обмін програмним забезпеченням для шифрування Кілька користувачів можуть зменшити витрати на інші інструменти захисту даних та підвищити ефективність управління безпекою.
3. Створення інтуїтивно зрозумілого інтерфейсу для користувачів з будь-яким рівнем технічної підготовки.
4. Оптимізація процесів для мінімізації часу, необхідного для обробки великих обсягів даних.

Цільова аудиторія:

Цільова аудиторія – користувачі, які прагнуть забезпечити захист своїх особистих та конфіденційних даних, зокрема, бізнесмени, фрілансери, студенти, а також компанії, що працюють з чутливою інформацією. Додаток розрахований для людей віком від 12 років. До того ж, додаток орієнтований на тих, хто активно використовує цифрові технології для роботи, комунікації та обміну інформацією, і потребує надійних інструментів для захисту своїх даних від несанкціонованого доступу та кібератак.

Очікувані результати:

- **Забезпечення конфіденційності даних:** підвищує рівень інформаційної безпеки для приватних осіб та компаній.
- **Покращення рівня довіри:** збільшення довіри між користувачами за допомогою довірених методів шифрування.
- **Зниження ризику кібератак:** зменшити витік даних з повідомленнями шифрування.
- **Покращення безпеки цифрового середовища:** надійні механізми захисту для підвищення рівня цифрової безпеки користувачів цифрової безпеки користувачів.

Підведемо підсумок цього підрозділу, що цей проект має суттєвий позитивний вплив на безпеку даних, може підвищити конфіденційність користувачів, покращити цифрову безпеку користувачів та сприяти розробці довіри до технології перегляду інформації.

4.4.2 Опис проектованого продукту

Проектований продукт: Програма для шифрування.

Додаток для шифрування - це інноваційне програмне забезпечення для захисту даних, яке забезпечує надійне шифрування та захист особистої інформації від користувачів. Основна мета - надати користувачам зручні інструменти для шифрування, повідомлень та інших важливих даних, забезпечення конфіденційності та безпеки. Додаток зосереджується на

користувачах, які захищають особисту та ділову інформацію, зменшують ризик витоків даних та збільшують цифрову безпеку.

Основні функції та можливості програми :

1. **Шифрування тексту**

- Введення відкритого (нешифрованого) тексту.
- Введення ключа (ключового слова).
- Генерація шифрованого тексту за допомогою алгоритму Веженера.

2. **Підтримка різних алфавітів –**українська та англійська мова .

3. **Автоматичне повторення ключа** – програма автоматично повторює ключ, щоб він збігався за довжиною з відкритим текстом.

4. **Виведення результату** .

Користувацький інтерфейс

Головний екранцевідправна точка для взаємодії з застосуванням користувача. Він має простий і зрозумілий дизайн, який дозволяє користувачам вирішувати шифрувати, розшифрувати чи налаштувати. Також можна включити короткі інструкції щодо використання.

Екран шифрування тексту який містить інтуїтивну форму для введення відкритого тексту та ключа. Після натискання кнопки "Зашифрувати" виводиться результат – шифрований текст. Передбачено копіювання або збереження результату.

Отже, моя програма забезпечує всебічне рішення для безпечного обміну текстовою інформацією. Це не тільки спрощує дані про процеси шифрування та дешифрування, але й збільшує міру цифрових можливостей та захисту особистої інформації та заохочує формування безпечного цифрового середовища.

4.4.3 Оцінка ринку

Оцінка ринку - важливий крок у розробці та впровадженні додатків для шифрування тексту методом Веженера як і моя програма. Таким чином ми

можемо зрозуміти проблеми, пов'язані з безпекою особистої інформації в потенційному попиту, конкурентних середовищах, цільових групах та сучасних цифрових просторах.

Аналіз цільової аудиторії

Аналіз цільової аудиторії приведено в таблиці 4.1.

Таблиця 4.1

Аналіз цільової аудиторії

Вік користувачів	Від 14років. Молодь, студенти, IT-фахівці, журналісти, активні користувачі мережі.
Застосування	Великі та середні міста, де цифрова безпека стає все більш актуальною.
Професійний статус користувачів	Студенти, освітяни, програмісти, журналісти, блогери, активісти, підприємці.
Мотиви	Захист особистих даних, навчальні цілі, інтерес до криптографії, цифрова грамотність.

Застосування програми :

Навчальні заклади

Університети, школи, технікуми — як частина навчальних програм із криптографії або інформатики.

Фахівці в галузі безпеки

Люди, що працюють у сфері кібербезпеки, або ті, хто шукає прості інструменти шифрування для особистого користування.

Інтернаціоналізація

Підтримка кількох мов та алфавітів (латиниця, кирилиця, інші) дозволить залучити користувачів з різних країн.

Аналіз попиту:

У сучасних цифрових умовах, де проблеми конфіденційності та безпеки особистої інформації стають все більш актуальними, зростає попит на прості, доступні пристрої шифрування. Люди все частіше шукають можливості захистити особисті комунікації, нотатки чи конфіденційні дані.

Метод Веженера один з класичних прикладів симетричного шифрування ,якийє значним потенціалом для практичного використання. Простота алгоритмів, інтуїції та розуміння покращують привабливість користувачів, які знайомлять один одного в основах шифрування, або серед користувачів, які мають комфортний інструмент для шифрування текстів у повсякденному житті. Тим часомтакож є важливим економічним фактором. Багато існуючих продуктів захисту даних важко використовувати або мати оплачувану основу. Моя програма - це приваблива альтернатива.

Програми для шифрування даних були важливим способом вирішення щоденних проблем тривалий час, як і моя програма - це спосіб зробити шифрування доступним для будь -кого. Завдяки інтуїтивно зрозумілою поверхнею, швидкій роботі та здатності шифрувати/дешифрувати, застосування популярний у широкому діапазоні аудиторій.

Отже, підведемо висновки з нашого підрозділу. Оцінки ринку показують, що моя програма має хороший потенціал для успішного запуску та розвитку. Збільшення інтересу до конфіденційності, навчання шифрування та поширеність комп'ютерних пристроїв створюють вигідні умови для фінансування продукції. Своєчасна локалізація, гнучкість конфігурації та вміст навчання можуть виявити стабільну нішу на ринку інструментів шифрування.

4.4.3 Стратегія маркетингу

Маркетингова стратегія є ключовим елементом успішного запуску та розвитку будь-якого програмного продукту, включаючи комп’ютерний застосунок для шифрування тексту методом Веженера. В умовах зростаючого попиту на безпечне зберігання та передачу інформації, ефективна маркетингова

стратегія дозволяє не лише привернути увагу до продукту, але й утримувати користувачів, формуючи довіру та інтерес до теми криптографії.

Цей підрозділ присвячено аналізу методів просування застосунку на ринку. Розглядаються ключові канали комунікації, які сприятимуть підвищенню впізнаваності продукту, залученню цільової аудиторії та стимулюванню лояльності користувачів.

Методи просування:

Соціальні мережі (Facebook, Instagram, YouTube).

Цілеспрямована реклама

Реклама у Facebook, Instagram, Reddit, X (Twitter) або YouTube, орієнтована на ІТ-спільноти, студентів технічних спеціальностей, викладачів, журналістів, письменників, дослідників, які працюють із конфіденційною інформацією.

Контент-маркетинг:

Створення навчального контенту:

1.Дописи у блогах та на тематичних форумах (Stack Overflow, Quora);

2.Відео на YouTube із поясненням принципів роботи шифру Віженера;

3.Демонстрації функціоналу застосунку;

4.Порівняння з іншими шифрами та приклади використання в історичному контексті.

Такий контент підвищує обізнаність та стимулює завантаження програми.

Партнерства з навчальними закладами та освітніми платформами:

- **Університети, технікуми, ліцеї**

Запровадження програмного продукту як частини навчального процесу у курсах з криптографії, інформаційної безпеки, історії інформатики або програмування. Можливе створення версії для навчальних закладів із доступом до демонстраційних сценаріїв.

- **Освітні платформи та конкурси**

Співпраця з онлайн-курсами або олімпіадами з ІТ, математичних гуртків – просування застосунку як інструменту для вивчення основ симетричного шифрування.

Висновок: Ефективна маркетингова стратегія для моєї програми, як комп'ютерного застосунку повинна базуватися на освітній та просвітницькій цінності продукту. Цільовою аудиторією є студенти, викладачі, ІТ-фахівці, а також користувачі, що цікавляться інформаційною безпекою. Комбінація соціального маркетингу, партнерств з навчальними закладами та створення якісного навчального контенту дозволить успішно позиціонувати програму на ринку та забезпечити її стабільний розвиток. Маркетингова стратегія орієнтована на залучення користувачів, які цікавляться інформаційною безпекою, цифровою грамотністю та криптографією. Використання соціальних мереж, контент-маркетингу, співпраця з навчальними закладами та ІТ-спільнотами, а також участь у профільних освітніх ініціативах дозволять ефективно просувати застосунок серед цільової аудиторії. Ці методи сприятимуть підвищенню обізнаності про важливість шифрування, популяризації класичних криптографічних методів та забезпечать довгостроковий розвиток і впізнаваність продукту на ринку освітнього та прикладного програмного забезпечення.

4.4.4 План виробництва додатку

Планування виробництва

Рівень 1:

Аналіз вимог та проектування архітектури додатку

Початковий етап включає детальний аналіз потреб користувачів та визначення основної функціональності програми. Після запису вимог визначаються архітектурний дизайн, вибір технологій та структуру бази даних. Ця фаза закладає основу для подальшого розвитку та забезпечує чіткість та послідовність у кожному наступному етапі.

Рівень 2:

Основна розробка функціонального модуля

Під час цієї фази основні функції в Додатку реалізуються відповідно до зазначених вимог. Сюди входять розробка користувачів, логіка шифрування та інтеграція в інші системи. Кожен модуль тестується на надійність функціональності та сумісності з іншими частинами системи.

Рівень 3:

Додаток

Тестування та захист на цьому етапі цього етапу виконує різні тестові типиодиниць, інтеграційних тестів та систем. Основна мета - визначити та виправити помилки в програмі для забезпечення надійності та стабільності перед затвердженням.

Рівень 4:

BETA запуск та колекція рейтингів користувачів

Після успішних тестів,бета - версія програми доступна обмеженій кількості користувачів. Відгуки користувачів про загальні враження від застосування функцій, інтерфейсів та додатків збираються.

Рівень 5:

Стабільна версія

Версія та постійні оновленнядляпісля успішного завершення всіх етапів та отриманих зворотного зв'язку буде створена стабільна версія програми. Після публікації триватимуть періодичні оновлення, включаючи підтримку та підтримку нових функцій, помилок та сумісності компонентів платформи.

Отже, цей виробничий план дозволяє систематично та ефективно розробляти свої програми, щоб забезпечити високу якість, відповідність та успіх на ринку для ваших користувачів.

4.4.5 Юридичний план

Юридичний план охоплює ключові заходи, необхідні для забезпечення правової відповідності та захисту даних користувачів при розробці та впровадженні комп’ютерного застосунку для шифрування текстів:

1. Реєстрація компанії та отримання дозволів

Першим кроком є офіційна реєстрація юридичної особи, яка буде здійснювати розробку та поширення програмного забезпечення. Також необхідно отримати дозволи на ведення господарської діяльності у сфері IT, за потреби — ліцензії (наприклад, якщо передбачена обробка платіжних даних користувачів).

2.Дотримання законодавства про захист персональних даних

Комп’ютерний застосунок опрацьовуватиме персональні дані користувачів (ім’я, контактна інформація, геолокаційні дані маршрутів тощо), тому він має відповідати вимогам українського законодавства про захист персональних даних, а також міжнародних норм (наприклад, GDPR у разі виходу на європейський ринок).

Це включає:

- впровадження політики захисту конфіденційності;
- шифрування даних;
- обмеження доступу до чутливої інформації.

3. Розробка користувацької угоди та політики конфіденційності

Ці документи мають бути доступні всім користувачам перед початком користування застосунком. Вони регламентують:

- права та обов’язки користувача;
- правила використання комп’ютерного застосунку;
- межі відповідальності розробника;
- порядок обробки персональних даних.

Висновок:

Юридичний супровід проєкту — критично важлива складова для легального функціонування комп'ютерного застосунку для карпулінгу. Завдяки відповідній реєстрації, дотриманню норм захисту даних та прозорій взаємодії з користувачами, проєкт забезпечує правову безпеку як для розробника, так і для кінцевих користувачів, що підвищує рівень довіри та лояльності до сервісу.

4.4.6 Оцінка ризиків

Оцінка ризиків є важливою складовою в управлінні проектом, що дозволяє виявити потенційні загрози та проблеми, які можуть виникнути під час розробки та експлуатації застосунку. Це можуть бути технічні труднощі, зміни в вимогах користувачів або зміни в законодавстві, що впливають на бізнес-процеси. Завдяки оцінці ризиків можна заздалегідь розробити стратегії для їхнього управління, що допомагає мінімізувати несподівані ситуації та зробити проект більш стабільним. Визначення ризиків дозволяє команді проекту адаптуватися до можливих змін і ефективно реагувати на них.

Види ризиків:

1. **Технічні ризики:**

Проблеми з сумісністю:

- **Проблеми з сумісністю:** Застосунок може стикатися з проблемами сумісності на різних платформах (Windows, macOS, мобільні пристрої), що ускладнить його коректну роботу.
- **Безпека даних:** Витоки персональних даних користувачів через несанкціонований доступ або кібератаки є великим ризиком. Це вимагає належного захисту даних, використання шифрування та виконання регулярних аудитів безпеки.

2. **Бізнес-ризики:**

- **Конкуренція:** Зростаюча кількість інших додатків або сервісів може ускладнити залучення користувачів. Це потребує продуманої маркетингової стратегії та диференціації продукту.

3. Організаційні ризики:

- **Кадрові проблеми:** Невідповідність складу команди або проблеми у взаємодії між учасниками можуть призвести до затримок у розробці або погіршення якості продукту.
- **Маркетинг та просування:** Невірна оцінка бюджету для маркетингових кампаній або неефективна рекламна стратегія можуть призвести до слабкого залучення користувачів і зниження впізнаваності програми.

Страхування

Для будь-якого технологічного проекту важливо передбачити страхування для зниження фінансових ризиків у разі негативних подій:

1.Страхування від юридичних ризиків:

Високотехнологічні проекти можуть підпадати під судові позови з приводу порушення авторських прав або конфіденційності. Страхування може покривати витрати на юридичні послуги та відшкодування збитків.

2.Страхування від кіберзагроз:

Оскільки застосунок обробляє персональні дані користувачів, ризики витоку даних або атак з боку хакерів можуть мати серйозні фінансові наслідки. Страхування від кіберзагроз забезпечує фінансову підтримку у разі витоків даних, відновлення репутації та витрат на відновлення інформації.

3.Страхування інфраструктури:

Охорона серверного обладнання, хостингових послуг і інших критичних активів проекту є важливим аспектом. Страхування покриває витрати на відновлення інфраструктури у разі природних катастроф, крадіжок або фізичних пошкоджень.

Зробимо висновок з цього підрозділу , щоефективне управління ризиками та належне страхування можуть значно знизити вплив негативних подій на розвиток проекту. Це дозволяє не лише знизити фінансові витрати на відшкодування збитків, але й підвищити довіру користувачів та інвесторів до

проєкту. Враховуючи постійну зміну технологічних умов та нормативного середовища, такі стратегії стають критичними для успіху проекту на конкурентному ринку.

4.4.7 Розрахунок витрат на створення програми

У межах дипломного проекту був розроблений комп'ютерний додаток. Для обґрунтування доцільності його створення проведено розрахунок вартості розробки та економічної ефективності, що дозволяє визначити рівень інвестицій, собівартість програмного продукту та термін його окупності.

Середня заробітна плата розробників (Junior) складає 32000 грн/міс. Додаткові витрати за сплату інтернету 350 грн/місяць.

Вартість програми розраховується по формулі наведені в таблиці 4.2 :

Таблиця 4.2

Витрати на розробку програмного забезпечення

Найменування витрат	Одиниця вимірювання	Кількість
1	2	3
Термін розробки	Місяць	2
Основна та допоміжна заробітня плата	грн	32,000
Витрати на допоміжні матеріали	грн	3,000
Витрати на електроенергію	грн	600
Витратина інтернет	грн	700

$C_{пр} = (32,000 + 3,000 + 600 + 700) * 2 = 72,600,$ це не включаючи рекламу.

4.4.8 Розрахунок економічної ефективності

Розрахунок економічної ефективності комп'ютерного додатку передбачає оцінку витрат на його розробку, технічну підтримку, оновлення та маркетингове просування у порівнянні з потенційними доходами від продажу або інших джерел монетизації (наприклад, ліцензії, підписки, донати, реклама).

Ключовим чинником економічної доцільності є співвідношення понесених витрат до прогнозованого прибутку. Ретельне планування дозволяє приймати обґрунтовані рішення щодо доцільності інвестування у проект, оцінити термін його окупності, а також прорахувати стратегію масштабування та прибутковості.

У нашому випадку передбачається, що комп'ютерний додаток буде розповсюджуватись на платній основі (разовий продаж або ліцензія). Припустимо, що додаток коштує 500 грн, і в перші місяці після запуску мінімум 1 користувач на місяць здійснює покупку.

Таблиця для розрахунку терміну окупності:

Таблиця 4.2

Розрахунок терміну окупності

$T = \frac{C_{\text{пр}}}{l}$, де

С(пр)- витрати на створеня програми;

L - місячний дохід від продаж;

Тобто, максимальний термін окупності програмного забезпечення:

Т=72,600/500=145 місяців = 12 років.

Також можуть бути додаткові джерела доходу, які зменшать термін окупності:

1.Розміщення реклами в додатку;

2.Платні оновлення чи розширені версії;

3.Спонсорські угоди або партнерства.

Якщо кількість клієнтів в місяць збільшиться до 2 користувачів, то термін окупності скоротиться вдвічі.

Висновок:

Економічна ефективність проекту визначається не лише початковими витратами на розробку, а й здатністю генерувати стабільний дохід у довгостроковій перспективі. У випадку з комп'ютерним додатком для

шифрування, при правильно побудованій маркетинговій стратегії та належному просуванні, існує потенціал для скорочення терміну окупності та досягнення прибутковості. Оцінка ринку, конкурентного середовища та потреб користувачів є вирішальними факторами для фінансової стабільності продукту.

ВИСНОВКИ

Завдання дослідження:

1) Теоретичне обґрунтування сутності атаки типу "людина посередині" (MITM), її механізми та особливості у контексті хмарних сервісів.

2) Аналіз ризиків і вразливостей.

3) Огляд методів захисту та шифрування даних

4) Розробка пропозицій оптимальних методів та інструментів для захисту хмарних сервісів від MITM-атак.

5) Розробити алгоритм або методику, що забезпечує підвищення рівня безпеки передавання даних.

У ході дослідження було детально проаналізованосутністьатаки типу "людина посередині" (MITM), що є однією з найнебезпечніших загроз в умовах інтенсивного розвитку хмарних технологій. Теоретичне обґрунтуваннязасвідчило,що MITM-атаки можутьвиконуватисяшляхом перехоплення, підміни чи маніпуляцій інформацією,яка передається між користувачем і хмарним сервером. Особливої увагипотребуєвикористання ненадійних мереж, відкритих Wi-Fi, а також слабке шифрування трафіку.

Аналіз ризиків і вразливостей показав,що до головних загроз відносяться:

1.Відсутність наскрізного шифрування,

2.Застарілі протоколи передачі даних,

3.Відсутність автентифікації серверів і клієнтів,

4.недоліки в конфігурації TLS/SSL.

Також хмарні середовища часто є мішенню для атак через свою публічну доступність і складність у централізованому адмініструванні безпекою.

Крім того, було розроблено методологію дляпокращення захисту передачі даних. Вона передбачає поетапне впровадження багаторівневої аутентифікації.Вона включає шифрування на кожному етапі зв'язку ,разом з контролем доступу,що враховує контекст - такий як геолокація та пристрій.

У результаті виконання дипломного проекту було розроблено комп’ютерний додаток для шифрування та дешифрування текстової інформації методом Віженера. Цей підхід проілюстрував основні засади симетричного шифрування та наглядно показав, як навіть найпростіші засоби можуть істотно ускладнити роботу зловмисникові, коли ключ відсутній. Додаток дозволяє користувачам вводити текст, задавати ключове слово та отримувати зашифрований або розшифрований результат у зручному графічному інтерфейсі. Рішення орієнтоване на забезпечення базового рівня конфіденційності персональних повідомлень або документів.

Під час роботи над дипломним проєктом було реалізовано такі етапи:

1. Детальне дослідження предметної області, аналіз готових рішень та постановка задачі.

Було проаналізовано основні методи симетричного шифрування, зокрема алгоритм Віженера, його історію, переваги та обмеження. Також досліджено сучасні аналоги програм, що реалізують просте текстове шифрування, визначено їхні недоліки та користувацькі проблеми, які не вирішуються. На основі цього сформульовано завдання на розробку зручного, легкого у використанні додатку з чітким акцентом на захист текстової інформації.

2. Проєктування

Проєктування додатку охоплювало створення логічної структури програми, розробку графічного інтерфейсу, вибір відповідних засобів розробки, мовОЮ програмування C++за допомогою Windows Form. Було побудовано діаграми взаємодії компонентів системи, розроблено макети вікон додатку.

3. Розробка і тестування функціональних модулів комп’ютерного додатку

Були реалізовані основні функціональні модулі: введення вихідного тексту, введення ключа, шифрування та дешифрування, збереження та копіювання результату. Проведено тестування додатку з різними типами даних

(короткі, довгі тексти, кирилиця, латиниця), щоб перевірити його стійкість до помилок, правильність роботи алгоритму та стабільність функціонування.

4. Опрацювання техніко-економічної частини та підрахунок собівартості програмного забезпечення

Проведено аналіз витрат на розробку, включаючи оплату праці розробника, вартість обладнання, електроенергії, програмного забезпечення та допоміжних матеріалів. Здійснено розрахунок економічної ефективності, терміну окупності проєкту та потенційної прибутковості за умов монетизації додатку. Підготовлено повну документацію до програми, включно з інструкцією користувача, описом функцій та кодовою базою.

У процесі розробки були опрацьовані ключові елементи розробки робота з текстовими даними, реалізація алгоритму Віженера. У результаті було створено функціональний, інтуїтивно зрозумілий комп’ютерний додаток, який дозволяє ефективно шифрувати та дешифрувати текст, підвищуючи безпеку цифрового листування користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Безпека хмарних технологій. KPMG. URL: https://kpmg.com/ua/uk/home/insights/2024/03/securing-the-cloud.html

2. Кібербезпека та кібервійна: Виклики та заходи для захисту інтернет-простору. Дніпропетровська обласна універсальна наукова бібліотека ім. Первоучителів слов'янських Кирила і Мефодія. URL: https://www.libr.dp.ua/student_notes_it_referat6.html

3. SPEKA, Дані під наглядом: як захистити себе та своїх клієнтів. Чекліст від співзасновника Finmap Івана Каунова. URL: https://speka.media/kiberbezpeka/dani-pid-naglyadom-yak-zahistiti-sebe-ta-svoyih-kliyentiv-kvr63v

4. HOSTPRO, SSL-сертифікат: що це таке, навіщо він потрібен і як його отримати. URL: https://hostpro.ua/blog/ua/what-is-ssl-certificate/

5. NETREAK JOURNAL, Що таке VPN-з'єднання і як ним користуватися. URL: https://netpeak.net/uk/blog/shcho-take-vpn-z-ednannya-i-yak-nim-koristuvatisya/

6. HostZealot, Що таке DNSSEC і чим важлива ця технологія. URL: https://www.hostzealot.com.ua/blog/about-web-hosting/scho-take-dnssec-i-chim-vazhliva-tsya-tehnologiya

7. CyberSet, IDS (Intrusion Detection System): Захист. URL: https://cyberset.com.ua/network/what-is-ids-intrusion-detection-system-zakhyst/

8. IITD, Аналіз мережевого трафіку (NTA – Network Traffic Analysis). URL: https://iitd.ua/analiz-merezhevogo-trafiku-nta/

9. HackYourMom, Посібники про Honeypot. URL: https://hackyourmom.com/kibervijna/posibnyky-pro-honeypot-vid-a-do-ya/

10. Wikipedia, ARP spoofing. URL: https://uk.wikipedia.org/wiki/ARP_spoofing

11. Smart, Що таке багатофакторна автентифікація (MFA)?. URL:

https://cloud.smart-it.com/news-post/what-is-mfa/

12. KingstonTehnology, Що таке шифрування та як воно працює?. URL:

https://www.kingston.com/ua/blog/data-security/what-is-encryption

13. StudFiles, Криптографічні протоколи. URL:

https://studfile.net/preview/6012701/page:45/

14. Bilous, Шифрування методом: Цезаря. Вермана. Віженера. URL:

https://bilousmi.wordpress.com/2014/10/29/%D1%88%D0%B8%D1%84%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BC-%D1%86%D0%B5%D0%B7%D0%B0%D1%80%D1%8F-%D0%B2%D0%B5%D1%80%D0%BC%D0%B0%D0%BD%D0%B0-%D0%B2%D1%96/

15. Huzva12, Лабораторна 2 (Шифр Віженера). URL:

https://huzva12.wordpress.com/2014/12/18/%D0%BB%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0-2/

16. Криптографія UA, Шифр Віженера. URL:

https://crypthography.blogspot.com/2016/04/blog-post_71.html

17. Hostkoss blog, Шифрування: Типи та алгоритми. Що це і який тип шифрування кращий?. URL:

https://hostkoss.com/b/uk/encryption-types-algorithms/#Symmetric-encryption

18. StudFile, Об'єктно-орієнтоване проектування. URL:

https://studfile.net/preview/9404039/page:7/

19. MindOnMap, Контекстна діаграма. URL:

https://www.mindonmap.com/uk/blog/context-diagram/#part1

20.StudFiles, Діаграма станів. URL:

https://studfile.net/preview/5010027/page:5/

21. Wikipedia, Інтерфейс користувача. URL: https://uk.wikipedia.org/wiki/

22. FoxmindEd, Visual Studio. URL: https://foxminded.ua/cs-seredovyshche-rozrobky/

23.Wikipedia,C#. URL: https://uk.wikipedia.org/wiki/C_Sharp

24.Wikipedia, Windows Forms. URL: https://uk.wikipedia.org/wiki/Windows_Forms

25.Blog.imena.ua. Атаки типу Man-In-The-Middle. URL:

https://www.imena.ua/blog/man-in-the-middle/

26.GRIDINSOFT, Визначення атаки посередника (MITM). URL: https://gridinsoft.ua/mitm

27.Ranktracker, Кібербезпека, URL: https://www.ranktracker.com/uk/blog/cyber-security-why-its-imperative-for-businesses-to-prioritize/

28.NordVPN, Різні типи кібератак,URL:https://nordvpn.com/uk/blog/shcho-take-kiberataka/

29.Cybercalm, Що таке атака “Людина посередині” (MITM), URL:

https://cybercalm.org/novyny/shho-take-ataka-man-in-the-middle-ta-yak-sebe-zahystyty/

30.ProxyElite, MITM (Людина посередині), URL:

https://proxyelite.info/uk/glossary/mitm-man-in-the-middle/

31.Wikipedia, Асиметричні алгоритми шифрування. URL:

https://uk.wikipedia.org/wiki/%D

32. Studcon, Шифрування даних. URL:https://studcon.org/shyfruvannya-danyh

33.JavaScript, Основи криптології. URL:https://javascript.org.ua/osnovi-kriptologi%D1%97-klyuch-do-czifrovo%D1%97-bezpeki/

34.Ukrfintech, Шифрування даних у сучасному світі. URL:

https://ukrfintech.com.ua/kriptografiya-v-suchasnomu-sviti-finansiv/

35. Exbace, Порівняння симетричного і асиметричного шифрування. URL:

https://exbase.io/uk/wiki/simetrichne-i-asimetrichne-shifruvannya

ДОДАТОК А. Код розробки:

```
using System;

public class VigenereCipher
{
    const string defaultAlphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    readonly string letters;

    public VigenereCipher(string alphabet = null)
    {
        letters = string.IsNullOrEmpty(alphabet) ? defaultAlphabet
: alphabet;
    }

    //генерування повторюваного пароля
    private string GetRepeatKey(string s, int n)
    {
        var p = s;
        while (p.Length < n)
        {
            p += p;
        }

        return p.Substring(0, n);
    }

    private string Vigenere(string text, string password, bool
encrypting = true)
    {
        var gamma = GetRepeatKey(password, text.Length);
        var retValue = "";
        var q = letters.Length;

        for (int i = 0; i < text.Length; i++)
        {
            var letterIndex = letters.IndexOf(text[i]);
            var codeIndex = letters.IndexOf(gamma[i]);
            if (letterIndex < 0)
            {
                //якщо літера не знайдена, додаємо її в незмінному
вигляді
                retValue += text[i].ToString();
            }
            else
            {
                retValue += letters[(q + letterIndex + ((encrypting
? 1 : -1) * codeIndex)) % q].ToString();
            }
```

```
        }

        return retValue;
    }

    //шифрування тексту
    public string Encrypt(string plainMessage, string password)
        => Vigenere(plainMessage, password);

    //дешифрування тексту
    public string Decrypt(string encryptedMessage, string password)
        => Vigenere(encryptedMessage, password, false);
}

class Program
{
    static void Main(string[] args)
    {
        //передаємо в конструктор класу літери українського
алфавіту
        var cipher = new
VigenereCipher("АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ");
        Console.Write("text: ");
        var inputText = Console.ReadLine().ToUpper();
        Console.Write("key: ");
        var password = Console.ReadLine().ToUpper();
        var encryptedText = cipher.Encrypt(inputText, password);
        Console.WriteLine("Зашифрований текст: {0}",
encryptedText);
        Console.WriteLine("Дешифрований текст: {0}",
cipher.Decrypt(encryptedText, password));
        Console.ReadLine();
    }
}
```