Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2022 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-21М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Сушков В.В.
«____» _____ 2022 р.

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2022 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Сушкову Вадиму Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 19-13 від 17.08.2022 року

3. Строк подання студентом роботи до захисту *10.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2022	14.11.2022
Охорона праці	Оришака О.В.	06.10.2022	16.11.2022

7. Дата видачі завдання « 6 » вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2022 р.	
3.	Розробка моделі компонента	20.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2022 р.	
6.	Програмування алгоритмів	10.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2022 р.	
9.	Оформлення ПЗ	17.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р.	

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Смірнов О.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Сушков В.В.
(прізвище та ініціали)

АНОТАЦІЯ

Сушков В.В. Дослідження та програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи хмарного сервісу з використанням алгоритму TDEA.

Метою розробки є дослідження та програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA.

Об'єктом дослідження є процес хмарного сервісу з використанням алгоритму TDEA.

Предметом дослідження є методи хмарного сервісу з використанням алгоритму TDEA.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Visual C#.

Ключові слова: комп'ютерна інженерія, хмарний сервіс, TDEA

ABSTRACT

Sushkov V.V. Research and software implementation of the cloud service system using the TDEA algorithm. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the cloud service system using the TDEA algorithm.

The purpose of the development is the research and software implementation of the cloud service system using the TDEA algorithm.

The object of research is the cloud service process using the TDEA algorithm.

The subject of the study is cloud service methods using the TDEA algorithm.

Research methods are based on information protection methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the cloud service system using the TDEA algorithm.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Visual C# environment.

Keywords: computer engineering, cloud service, TDEA

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 7
1.1 Призначення системи..... 7
1.2 Область застосування..... 9
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 14
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 14
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 23
2.3 Розгорнута постановка завдання 26
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 28
3.1 Опис функціонування системи 28
3.2 Розробка структурної схеми..... 37
3.3 Розробка функціональної схеми 41
3.4 Розробка діаграми процесів..... 45
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 48
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 48
4.2 Захист розробленого програмного забезпечення..... 58
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 61
6 НАУКОВА НОВИЗНА 64

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ..... 65
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 65
7.2 Розрахунок трудомісткості розробки програмної продукції..... 67
7.3 Визначення чисельності виконавців і планового фонду зарплати..... 69
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника..... 74
7.5 Визначення собівартості розробки та ціни програмної продукції..... 78
7.6 Визначення об’єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції..... 81
7.7 Визначення експлуатаційних витрат..... 81
7.8 Визначення економічної ефективності програмної продукції..... 83
7.9 Висновок..... 85
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ..... 86
8.1 Вступ..... 86
8.2 Пожежна безпека..... 87
8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста..... 89
8.4 Розробка заходів з умов поліпшення охорони праці..... 92
8.5 Розрахункова частина..... 94
8.6 Висновки до розділу..... 95
9 ОСНОВНІ ВИСНОВКИ..... 96
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 98

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

КСЗ	–	комплексна система захисту
ПЕОМ	–	персональна електронно-обчислювальна машина
СУБД	–	система управління базами даних
PGP	–	Pretty Good Privacy

ВСТУП

Актуальність теми. XXI століття – вік інформатики й інформатизації. Технологія дає можливість передавати й зберігати все більші обсяги інформації. Це благо має й зворотний бік. Інформація стає усе більше вразливою з різних причин:

– зростаючі обсяги збережених і переданих даних;

– розширення кола користувачів, що мають доступ до ресурсів ЕОМ, програмам і даним;

– ускладнення режимів експлуатації обчислювальних систем.

Тому все більшу важливість здобуває проблема захисту інформації від несанкціонованого доступу (НСД) при передачі й зберіганні. Сутність цієї проблеми – постійна боротьба фахівців із захисту інформації зі своїми "опонентами". Захист інформації – сукупність заходів, методів і засобів, що забезпечують:

– виключення НСД до ресурсів ЕОМ, програмам і даним;

– перевірку цілісності інформації;

– виключення несанкціонованого використання програм (захист програм від копіювання).

Очевидна тенденція до переходу на цифрові методи передачі й зберігання інформації дозволяє застосовувати уніфіковані методи й алгоритми для захисту дискретної (текст, факс, телекс) і безперервної (мова) інформації.

Випробуваний метод захисту інформації від НСД – шифрування (криптографія). Шифруванням (encryption) називають процес перетворення відкритих даних (plaintext) у зашифровані (шифртекст, ciphertext) або зашифрованих даних у відкриті за визначеними правилами із застосуванням

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем хмарного сервісу з використанням алгоритму TDEA.

– Дослідження системи хмарного сервісу з використанням алгоритму TDEA.

– Програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA.

Об'єктом дослідження є процес хмарного сервісу з використанням алгоритму TDEA.

Предметом дослідження є методи хмарного сервісу з використанням алгоритму TDEA.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод хмарного сервісу з використанням алгоритму TDEA.

– Розроблено вітчизняний продукт хмарного сервісу з використанням алгоритму TDEA, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі хмарного сервісу з використанням алгоритму TDEA.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVI Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2022, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №13.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Призначенням системи є реалізація системи хмарного сервісу з використанням алгоритму TDEA, за рахунок програмної реалізації алгоритму захисту інформації 3DES. Цим алгоритмом буде шифруватися інформація кожного з легітимних користувачів системи, які будуть заходити у систему під своїм профілем.

Алгоритми шифрування реалізуються програмними або апаратними засобами. Є множина чисто програмних реалізацій різних алгоритмів. Через свою дешевизну (деякі й зовсім безкоштовні), а також все більшої швидкості процесорів ПЕОМ, простоти роботи й безвідмовності вони досить конкурентноздатні. Широко відома програма Diskreet з пакета Norton Utilities, що реалізує DES.

Не можна не згадати пакет PGP (Pretty Good Privacy, версія 2.1, автор Philip Zimmermann), у якому комплексно вирішені практично всі проблеми захисту переданої інформації. Застосоване стискання даних перед шифруванням, потужне керування ключами, симетричний (IDEA) і асиметричний (RSA) алгоритми шифрування, обчислення контрольної функції для цифрового підпису, надійна генерація ключів.

Апаратна реалізація алгоритмів можлива за допомогою спеціалізованих мікросхем (виробляються кристали для алгоритмів DH, RSA, DES, Skipjack, Дердстандарт 28147-89) або з використанням компонентів широкого призначення (через дешевизну й високу швидкість перспективні цифрові сигнальні процесори – ЦСП, Digital Signal Processor, DSP).

Для більшої надійності шифрування одночасно працюють два криптопроцесора, і блок даних в 64 бітів вважається правильно зашифрованим,

тільки якщо збігається інформація на виході обох блюмінгів. Швидкість шифрування – 250 КБ/с.

До алгоритмів шифрування пред'являються визначені вимоги:

– високий рівень захисту даних проти дешифрування й можливої модифікації;

– захищеність інформації повинна ґрунтуватися тільки на знанні ключа й не залежати від того, відомий алгоритм чи ні (правило Кіркхоффа);

– мала зміна вихідного тексту або ключа повинна призводити до значної зміни шифрованого тексту (ефект "обвалу");

– область значень ключа повинна виключати можливість дешифрування даних шляхом перебору значень ключа;

– економічність реалізації алгоритму при достатній швидкості;

– вартість дешифрування даних без знання ключа повинна перевищувати вартість даних.

За допомогою криптографічних методів можливим є:

– шифрування інформації;

– реалізація електронного підпису;

– розподіл ключів шифрування;

– захист від випадкової або навмисної зміни інформації.

Симетричні блокові шифри представлені в .NET класами DESCryptoServiceProvider, TripleDESCryptoServiceProvider, RijndaelManaged. Всі ці класи є нащадками абстрактного класу SymmetricAlgorithm, що описує все сімейство блокових алгоритмів із симетричними ключами. Клас описує властивості, що дозволяють маніпулювати основними параметрами алгоритму: розміром блоку, режимом роботи, ініціалізаційним вектором, ключем й іншими. І методи CreateEncryptor й CreateDecryptor, що повертають контексти (інтерфейс ICryptoTransform) для криптографічних трансформацій даних. Також є методи GenerateKey й GenerateIV для генерації ключів й ініціалізаційних векторів. Конкретні реалізації успадковуються від цього класу.

1.2 Область застосування

Областю застосування програмного забезпечення, яке буде розроблено у результаті виконання магістерського проектування є доступ до зашифрованих даних профілю у системах розмежування доступу.

Розробка системи захисту інформації повинна реалізовувати яку-небудь політику безпеки (набір правил, що визначають множина припустимих дій у системі), при цьому повинна бути реалізована повна й коректна перевірка її умов. Існують спеціальні моделі безпеки – системи, що функціонують у відповідності зі строго визначеним набором формалізованих правил, і реализуючі яку-небудь політику безпеки.

Зупинимося на трьох ключових математичних моделях безпеки комп'ютерних систем, як на найбільш ефективні й використовувані в цей час. Це моделі систем дискреційного, мандатного й рольового розмежувань доступу.

Модель систем дискреційного розмежування доступу

Дана модель характеризується розмежуванням доступу між пойменованими суб'єктами й об'єктами. Суб'єкт із визначеним правом доступу може передати це право будь-якому іншому суб'єктові. Для кожної пари (суб'єкт-об'єкт) повинне бути задане явне й недвозначне перерахування припустимих типів доступу (читати, писати й т.д.), які є санкціонованими для даного суб'єкта (індивіда або групи індивідів) до даного ресурсу (об'єкту). Можливі, щонайменше, два підходи до побудови дискреційного керування доступом:

– Кожний об'єкт системи має прив'язаного до нього суб'єкта, називаного власником. Саме власник установлює права доступу до об'єкта.

– Система має одного виділеного суб'єкта – суперкористувача, що має право встановлювати права володіння для всіх інших суб'єктів системи.

Можливі й змішані варіанти побудови, коли одночасно в системі присутні як власники, що встановлюють права доступу до своїх об'єктів, так і суперкористувач, що має можливість зміни прав для будь-якого об'єкта й/або

зміни його власника. Саме такий змішаний варіант реалізований у більшості операційних систем (UNIX або Windows сімейства NT).

Дискреційне керування доступом є основною реалізацією розмежувальної політики доступу до ресурсів при обробці конфіденційних відомостей відповідно до вимог до системи захисту інформації.

Мандатне керування доступом

Для реалізації цього принципу кожному суб'єктові й об'єкту повинні зіставлятися класифікаційні мітки, що відбивають місце даного суб'єкта (об'єкта) у відповідній ієрархії. За допомогою цих міток суб'єктам і об'єктам повинні призначатися класифікаційні рівні (рівні уразливості, категорії таємності й т.п.), що є комбінаціями ієрархічних і неієрархічних категорій. Дані мітки повинні бути основою мандатного принципу розмежування доступу. КСЗ при уведенні нових даних у систему повинен запитувати й одержувати від санкціонованого користувача класифікаційні мітки цих даних. При санкціонованому занесенні в список користувачів нового суб'єкта повинне здійснюватися зіставлення йому класифікаційних міток. Зовнішні класифікаційні мітки (суб'єктів, об'єктів) повинні точно відповідати внутрішнім міткам (усередині КСЗ).

КСЗ повинен реалізовувати мандатний принцип контролю доступу стосовно до всіх об'єктів при явному й схованому доступі з боку кожного із суб'єктів:

– суб'єкт може читати об'єкт, тільки якщо ієрархічна класифікація суб'єкта не менше, ніж ієрархічна класифікація об'єкта, і неієрархічні категорії суб'єкта містять у собі всі ієрархічні категорії об'єкта;

– суб'єкт здійснює запис в об'єкт, тільки якщо класифікаційний рівень суб'єкта не більше, ніж класифікаційний рівень об'єкта, і всі ієрархічні категорії суб'єкта включаються в неієрархічні категорії об'єкта.

Реалізація мандатних правил розмежування доступу повинна передбачати можливості супроводу зміни класифікаційних рівнів суб'єктів і об'єктів

спеціально виділеними суб'єктами. Повинен бути реалізований диспетчер доступу, тобто засіб, що здійснює перехоплення всіх обігів суб'єктів до об'єктів, а також розмежування доступу відповідно до заданого принципу розмежування доступу. При цьому рішення про санкціонованість запиту на доступ повинне прийматися тільки при одночасному його дозволі й дискреційними, і мандатними правилами розмежування доступу. Таким чином, повинен контролюватися не тільки одиничний акт доступу, але й потоки інформації.

Рольове розмежування

Основною ідеєю керування доступом на основі ролей є ідея про зв'язування дозволів доступу з ролями, призначуваним кожному користувачеві. Ця ідея виникла одночасно з появою багатокористувальницьких систем. Однак донедавна дослідники мало звертали увагу на цей принцип.

Рольове розмежування доступу являє собою розвиток політики дискреційного розмежування доступу, при цьому права доступу суб'єктів системи на об'єкти групуються з урахуванням їх специфіки їхнього застосування, утворюючі ролі.

Таке розмежування доступу є складовою багатьох сучасних комп'ютерних систем. Як правило, даний підхід застосовується в системах захисту СУБД, а окремі елементи реалізуються в мережних операційних системах.

Завдання ролей дозволяє визначити більш чіткі й зрозумілі для користувачів комп'ютерної системи правила розмежування доступу. При цьому такий підхід часто використовується в системах, для користувачів яких чітко визначений коло їхніх посадових повноважень і обов'язків.

Роль є сукупністю прав доступу на об'єкти комп'ютерної системи, однак рольове розмежування аж ніяк не є часткою случаємо дискреційного розмежування, так як її правила визначають порядок надання прав доступу суб'єктам комп'ютерної системи залежно від сесії його роботи й від наявних (або відсутніх) у нього ролей у кожний момент часу, що є характерним для систем мандатного розмежування доступу. З іншого боку, правила рольового

розмежування доступу є більше гнучкими, ніж при мандатному підході до розмежування.

Якщо підбити підсумок, то в кожної з перерахованих нами систем є свої переваги, однак ключовим є те, що жодна з описаних моделей не стоїть на місці, а динамічно розвивається. Прихильники є в кожної з них, однак, об'єктивно подивившись на речі, важко віддати перевагу якійсь одній системі. Вони просто різні й служать для різних цілей.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Розглянемо програми, які використовують, алгоритм 3DES.

До них відносяться наступні:

– PGP.

– MASPware SecuBox.

– Sealed Notes 1.0.

– 3DES Encryption 1.0.

– Система шифрування EFS в Windows Vista.

– Password Manager XP.

PGP

У процесах шифрування PGP використовує три основних типи алгоритмів: алгоритми криптосистем з відкритим ключем (RSA, DSA, Эльгамаль), алгоритми однобічних хеш-функцій (SHA1, MD5) і ітеративні блокові шифри (AES, CAST, 3DES, IDEA, Blowfish, Twofish). Основні параметри ітеративних блокових шифрів у реалізації PGP наведені у таблиці 2.1.

MASPware SecuBox

MASPware SecuBox – програма для зберігання особистих відомостей у зашифрованому виді. Програма, що забезпечує безпечне зберігання конфіденційних даних на пристрої, використовується кодування Triple-DES (використовується в банківських системах).

Усе впаковується й зберігається в одному файлі даних, що може бути збережений на карті пам'яті.

Таблиця 2.1 – Порівняння основних параметрів ітеративних блокових шифрів у реалізації PGP

Алгоритм	Ключ	Блок	Примітки
Triple-DES	168 біт	64 біт	Мережа Файстеля; має простір слабких і напівслабких ключів; швидкий; стійкий до криптоаналізу; порівняно низька стійкість ключа до лобової атаки (112 біт); перевірена 20 роками надійності; у розробці брало участь АНБ.
AES (Rijndael)	256 біт	128 біт	Унікальний, але простий дизайн (операції з таблицями масивів даних), що полегшує аналіз на наявність проломів; прийнятий у якості державного. стандарту США після відкритого конкурсу; у порівнянні з високою стійкістю дуже швидкий; відносно новий.
CAST	128 біт	64 біт	Мережа Файстеля (DES-подібний дизайн); не має слабких ключів; швидкий; стійкий до криптоаналізу; існує вже 10 років.
IDEA	128 біт	64 біт	Заснований на унікальній концепції (змішання операцій різних алгебраїчних груп); має простір слабких ключів; послужний список в 13 років; не всі роботи з криптоаналізу були опубліковані.
Twofish	256 біт	128 біт	Мережа Файстеля; один з фіналістів конкурсу AES; швидке шифрування, повільна установка ключа; складний дизайн, що утрудняє формальний аналіз; має великий запас міцності.
Blowfish	max 448 біт	64 біт	Мережа Файстеля; простий дизайн; швидке шифрування, повільна установка ключа; має невеликий простір слабких ключів; має високий запас міцності.

Для того щоб установити додаток MASPware SecuBox необхідно завантажити дистрибутив із сайту розроблювача. Якщо це cab файл, то перенести до пам'яті пристрою, якщо це exe файл, то приєднати пристрій до ПК за допомогою дата кабелю й запустити настановний файл при цьому додержуватися інструкцій на екрані. Крім самої програми буде запропоновано встановити MASPware HandNotes.

Для запуску програми необхідно відшукати відповідний ярлик у розділі "Пуск – Програми". Крім цього при установці, за замовчуванням – установлюється й плагін для "Сьогодні", що відображає елементи керування програми й забезпечує швидкий доступ як до самої програми, так і до її функцій. Після чого з'явиться вікно, у якому потрібно буде вибрати мову інтерфейсу (англійський або німецький).

Крім самої програми буде запропоновано встановити MASPware HandNotes. Робочий простір додатка організовано таким чином, що всі керуючі елементи перебувають у низі, а інший робочий простір зарезервований під записи. Для того щоб відредагувати запис, потрібно нажати ліву нижню кнопку, після чого відкривається вікно де будуть відображені елементи для редагування. Можна вибрати один із чотирьох кольору, товщину пера. Можна скасувати останній штрих пером або просто стерти ластиком. Через "Menu" можна експортувати або імпортувати створені замітки. Також вибираються типи сторінок: чисті, у клітинку або лінійку. Можна відправити сторінки по E-mail, можливість установки нагадувань на замітки, обмін сторінками через інфрачервоний порт.

MASPware SecuBox – дозволяє зберігати будь-який текст або двійковий код. При запуску потрібно буде створити нове сховище або відкрити вже створене. Після чого виставляється код для входу до потрібних документів, можна зберігати текстові документи й при бажання зберігати фотографії або документи. Для кодування використовується Triple-DES код (використовується в

банківських системах). Усе впаковується й зберігається в одному файлі даних, що може бути збережений на карті пам'яті.

Рисунок 2.1 – Інтерфейс користувача програми MASPware SecuBox

About – за допомогою цієї функції ми можемо переглянути інформацію про дану версію програми і її розроблювачі, у цьому вікні буде відображений сайт розроблювача, також можна зареєструвати програму для подальшого використання.

Достоїнства: Простий і лаконічний інтерфейс додатка, при установки встановлюється програма MASPware HandNotes. Програма, що забезпечує безпечне зберігання конфіденційних даних на пристрої, використовується кодування Triple-DES.

Недоліки: Програма поширюється платним шляхом EUR 9,95.

Sealed Notes 1.0

SealedNotes – програма для створення й зберігання заміток користувача в зашифрованому виді на КПК або смартфоні.

SealedNotes дозволяє вибрати один із трьох алгоритмів шифрування: Rijndael (AES), 3DES, RC2. За замовчуванням, програма використовує 3DES. Як відомо, DES був стандартом шифрування США, 3DES – його посилена версія.

Залежно від обраного алгоритму шифрування використовуються ключі шифрування різної довжини: для AES ключ шифрування має довжину 256 біт, для 3DES – 192 біт, RC2 – 128 біт. Ключі й пароль користувач установлює сам. Програма вміє автоматично генерувати ключ для шифрування.

Для кожної замітки зберігається інформація про дати створення, останнього доступу й зміни. Є швидкий пошук по замітках і експорт замітки у вигляді незашифрованого тексту в окремий обраний файл.

Рисунок 2.2 – Інтерфейс користувача програми SealedNotes

3DES Encryption 1.0

3DES Encryption 1.0. Шифрування файлів по 3DES-алгоритму. Підтримуються будь-які типи файлів.

Система шифрування EFS в Windows Vista

Як і раніше, в операційній системі Windows Vista є присутньою додаткова файлова система EFS, призначена для шифрування файлів і каталогів, що перебувають на розділах NTFS жорсткого диска. Ви можете виконати шифрування як за допомогою діалогу PROPERTIES файлу або каталогу, так і за допомогою програми командного рядка cipher.exe.

Етапи шифрування EFS.

Процес шифрування файлу за допомогою EFS можна розділити на наступні етапи.

1. Завантаження профілю. Якщо профіль користувача ще не завантажений (наприклад, таке буває, коли шифрування або дешифрування файлу відбувається за допомогою програми командного рядка runas.exe), тоді профіль користувача завантажується.

2. Створення файлу журналу. Починається процес шифрування. На цьому етапі створюється файл журналу шифрування з ім'ям формату efs.log, де X визначає номер файлу, що був зашифрований у поточний сеанс шифрування процесом lsasrv.exe. Даний файл створюється в каталозі System Volume Information.

3. Генерація FEK. Виконується генерація випадкового 128-бітного числа, використовуваного в якості FEK. FEK використовується операційною системою для шифрування вмісту файлу по визначеному алгоритму, після чого значення FEK додається до самого зашифрованого файлу. Однак перед тим як додати FEK до файлу, операційна система виконує шифрування FEK за допомогою алгоритму RSA, використовуючи при цьому відкритий користувальницький ключ. Після виконання всіх цих операцій файл вважається зашифрованим – щоб переглянути файл, потрібно розшифрувати його за допомогою FEK, а FEK можна розшифрувати тільки за допомогою ключів, які можна одержати тільки після завантаження облікового запису того користувача, що зашифрував файл (або тих користувачів, яким дозволений доступ до зашифрованого файлу). Як було

сказано вище, FEK використовується для шифрування вмісту файлу по визначеному алгоритму. Якщо питання захисту інформації для вас не є дозвільним, і ви постійно виконуєте шифрування за допомогою EFS, тоді операційна система Windows дозволяє вам змінити алгоритм, використовуваний при шифруванні файлу (на 3DES). Для цього потрібно скористатися одним із двох наведених нижче способів.

Щоб установити шифрування на основі 3DES для всіх криптографічних сервісів операційної системи Windows Vista, потрібно запустити оснащення gpedit.msc і перейти до підрозділу Computer Configuration/Windows Settings/Security Settings/Local Policies/Security Options. У даному підрозділі є елемент SYSTEM CRYPTOGRAPHY: USE FIPS COMPLIANT ALGORITHMS FOR ENCRYPTION, HASHING, AND SIGNING. Значення даного елемента потрібно встановити в ENABLED. Не забувайте, що використання алгоритму 3DES може сповільнити процес шифрування. * Також можна встановити використання алгоритму 3DES тільки службою EFS. Для цього потрібно параметру REG_DWORD типу AlgorithmID, розташованому в галузі реєстру HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\EFS, привласнити значення 0x6603. Однак це все відбудеться на наступних етапах шифрування файлу.

4. Одержання відкритого й закритого користувальницького ключа. Якщо раніше користувач ніколи не виконував шифрування файлів, тоді генерується пара користувальницьких ключів (закриті й відкритий користувальницькі ключі). У противному випадку відкритий користувальницький ключ береться з параметра REG_BINARY типу CertificateHash, розташованого в галузі реєстру HKCU\Software\Microsoft\Windows NT\CurrentVersion\EFS\CurrentKeys. На основі даних цієї галузі реєстру створюється сигнатура відкритого ключа користувача. Закриті ж ключі перебувають у каталозі %userprofile%\AppData\Roaming\Microsoft\Crypto\RSA. Вміст цього каталогу зашифровано на основі симетричного ключа, що називається майстер-ключем

користувача. Майстер ключ утримується в каталозі %userprofile%\AppData\Roaming\Microsoft\Protect. Він також зашифрований – за допомогою алгоритму 3DES і паролю користувача.

5. Створення DDF. Для шифруємого файлу створюється зв'язування ключів DDF (сукупність декількох DDF, що визначають користувачів, які можуть розшифрувати даний файл). DDF містить інформацію про користувача, що може відкрити даний файл, а також про його права доступу до файлу. У цю інформацію входить SID-користувача, ім'я контейнера, ім'я провайдеру, що зашифрував файл, хеш сертифіката EFS, використовуваний при розшифровці, а також зашифрований FEK.

6. Створення DRF. Для шифруємого файлу створюється зв'язування ключів DRF. Зв'язування DRF містить інформацію про всіх агентів відновлення, які можуть відкрити даний зашифрований файл. Інформація, що поміщається в DRF, аналогічна інформації, що поміщається в DDF. Агенти відновлення являють собою облікові записи користувачів, які можуть відкрити зашифрований файл, навіть якщо вони не входять у ту групу користувачів, яким дозволений доступ до файлу. Як правило, агентом відновлення є адміністратор. Це зроблено для того, щоб можна було відкрити зашифровані файли при втраті пароля від основного профілю користувача, що зашифрував файл, або при його ушкодженні.

7. Шифрування. Створюється резервна копія шифруємого файлу з ім'ям efs0.tmp. Надалі шифрування буде застосовуватися саме до резервного файлу, після чого його вміст буде скопійовано у вихідний файл. І в самому кінці відбудеться видалення резервного файлу, а також файлу журналу шифрування.

Password Manager XP

Ні для кого не є секретом той факт, що парольний захист є на сьогоднішній день найпоширенішим способом ідентифікації людини в різних сервісах. І це незважаючи на те, що він має багато недоліків, самий серйозний з яких – досить низька надійність. І дійсно, зловмисники можуть підібрати, підглянути або просто вгадати пароль користувача. І спостерігати це можна

суцільно й поруч. Однак у багатьох випадках заміна парольного захисту іншими видами ідентифікації сильно утруднена. Це пов'язане з необхідністю додаткових витрат, технічними й організаційними проблемами. Тобто ми виявляємося перед справжньою дилемою. З одного боку, парольний захист не забезпечує достатнього рівня безпеки, а з іншого боку – його не можна замінити. Що ж робити? Прекрасним виходом з такого положення є придбання спеціальної програми, так званого менеджера паролів. Це дуже вдале рішення, оскільки, з одного боку, воно дозволяє істотно підвищити ступінь надійності парольного захисту, а з іншого боку – вимагає лише мінімальних грошових вкладень і дозволяє обійти практично всі технічні й організаційні проблеми.

Принцип роботи менеджерів паролів дуже простий. Вся конфіденційна інформація записується в єдину базу даних (фактично в один файл), що зашифровується. Для того щоб одержати доступ до неї, необхідно знати спеціальне ключове слово. Перевага такого підходу полягає в тому, що користувачеві потрібно пам'ятати всього лише один пароль, а не десятки різних ключових слів. Це дуже добре. Адже в чому полягає найбільший недолік парольного захисту? Так в тому, що потрібно пам'ятати множина ключових слів. І користувачі прибігають до всіляких хитрувань, для того щоб полегшити свою долю: скрізь використовують той самий пароль, роблять паролі максимально простими, записують їх на папірцях або в текстових файлах на комп'ютері й т.п. Таким чином, менеджер ключових слів дозволяє без особливої праці істотно збільшити надійність парольного захисту. Яскравим прикладом такого продукту є програма Password Manager XP, розроблена фахівцями компанії CP Lab.

Розглянута утиліта може працювати з будь-якою кількістю баз даних, що відкриваються, по черзі. Фактично це забезпечує зручну й конфіденційну роботу з паролями кількох людей на одному комп'ютері. Кожна база складається з папок, які користувачі може додавати, видаляти й редагувати по своєму бажанню. Розділи можуть вкладатися друг у друга, створюючи як завгодно складну деревоподібну структуру. Саме цікаве полягає в тому, що для кожного з

розділів можна самостійно встановлювати набір полів, які будуть містити дані, що зберігаються в ньому. Причому уведені в кореневій папці поля можуть бути успадковані всіма підпапками. Це істотно полегшує й прискорює настроювання бази даних.

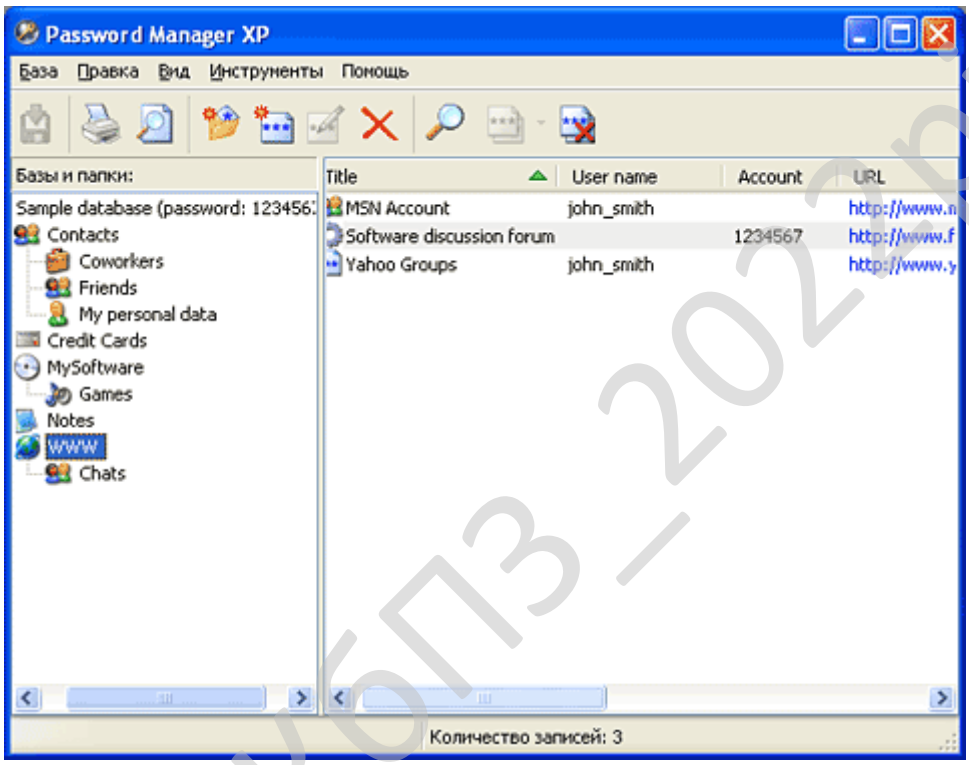

Рисунок 2.3 – Головне вікно програми Password Manager XP

Більшість людей вибирають програмне забезпечення по зручності використань і широті функціональних можливостей. Проте про безпеку забувати не можна. У противному випадку в один "прекрасний" момент можна стати жертвою зловмисника. У програмі Password Manager XP реалізований цілі ряди алгоритмів шифрування (3DES, Rijndael, Tea, Cast128, RC4, Serpent, Twofish). Всі вони добре відомі й відносяться до надійних криптографічних технологій. Крім того, у розглянутій утиліті реалізовано кілька додаткових можливостей для забезпечення безпеки: перевірка на наявність програм, що здійснюють моніторинг буфера обміну, автоматичне закриття бази у випадку неактиності

користувача, установка мінімально можливої довжини пароля, обмеження комп'ютерів, які мають доступ до бази по мережі, запис тимчасової інформації на диск тільки в зашифрованому виді й т.д. Все це свідчить про те, що інформація, що зберігається в базах програми Password Manager XP, дійсно надійно захищена.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Програмне забезпечення написане мовою Visual C#. Ця мова обрана виходячи з наступних міркувань. Visual C# – строго типізована об'єктно-орієнтована мова, призначена для розробки різноманітних безпечних і потужних застосунків, виконуваних у середовищі .NET Framework. Мовою Visual C# можна розробляти звичайні клієнтські застосунки Windows, веб-служби XML, розподілені компоненти, застосунки типу " сервер-клієнт", застосунки баз даних і багато яких інших. В Visual C# є розширений редактор коду, конструктори зі зручним користувальницьким інтерфейсом, вбудований відладник і багато інших засобів, покликані спростити розробку застосунків мовою Visual C# версії 5.0 і .NET Framework версії 4.5.

Синтаксис Visual C# дуже виразний, але простий у вивченні. Усі, хто знаком з мовами C, C++ або Java з легкістю взнають синтаксис із фігурними дужками, характерний для мови Visual C#. Розроблювачі, що знають кожну із цих мов, як правило, зможуть домогтися ефективної роботи з мовою Visual C# за дуже короткий час. Синтаксис Visual C# робить простіше те, що було складно в C++, і забезпечує потужні можливості, такі як типи значень Nullable, перерахування, делегати, лямбда-вираження й прямий доступ до пам'яті, чого немає в Java. Visual C# підтримує універсальні методи й типи, забезпечуючи більше високий рівень безпеки й продуктивності, а також ітератори, що дозволяють при реалізації колекцій класів визначати власне поводження ітерації, що може легко використовуватися в клієнтському коді. В Visual C# 5.0

вираження LINQ (Language-Integrated Query) роблять строго-типізований запит першокласною конструкцією мови.

Як об’єктно-орієнтована мова, Visual C# підтримує поняття інкапсуляції, спадкування й поліморфізму. Всі змінні й методи, включаючи метод `Main` – крапку входу застосунка – інкапсулюється у визначення класів. Клас може успадковувати безпосередньо з одного родового класу, але може реалізовувати будь-яке число інтерфейсів. Для методів, які перевизначають віртуальні методи в батьківському класі, необхідно ключове слово override, щоб виключити випадкове повторне визначення. У мові Visual C# структура схожа на полегшений клас: це тип, що розподіляється по стопках, що реалізує інтерфейси, але не підтримуюче спадкування.

На додаток до основних описаних об’єктно-орієнтованих принципів, мова Visual C# спрощує розробку компонентів програмного забезпечення завдяки декільком інноваційним конструкціям мови, у число яких входять наступні:

– Інкапсульовані підписи методів, називані делегатами, які підтримують строго-типізовані повідомлення про події.

– Властивості, що виступають у ролі методів доступу для закритих змінних-членів.

– Атрибути з декларативними метаданими про типи під час виконання.

– Вбудовані коментарі XML-документації.

– LINQ (Language-Integrated Query), що пропонує вбудовані можливості запитів у різних джерелах даних.

Якщо буде потрібно забезпечити взаємодію з іншим програмним забезпеченням Windows, таким як об'єкти COM або власні бібліотеки DLL Win32, у мові Visual C# можна використовувати процес, що називається "Interop". Процес Interop дозволяє програмам на Visual C# виконувати практично будь-які дії, які може виконувати вихідний додаток на C++. Мова Visual C# підтримує навіть покажчики й поняття "небезпечного" коду для тих випадків, коли прямий доступ до пам'яті має вкрай важливе значення.

Процес побудови Visual C# у порівнянні з C і C++ простий і є більше гнучким, чим в Java. Немає окремих файлів заголовка, а методи й типи не потрібно повідомляти в певному порядку. У вихідному файлі Visual C# може бути визначене будь-яке число класів, структур, інтерфейсів і подій.

Архітектура платформи .NET Framework

Програма мовою Visual C# виконується в середовищі .NET Framework – інтегрованому компоненті Windows, що містить віртуальну систему виконання (середовище CLR) і уніфікований набір бібліотек класів. Середовище CLR являє собою комерційну реалізацію корпорацією Майкрософт інфраструктури CLI, що є міжнародним стандартом, який лежить в основі створення середовищ виконання й розробки, у яких забезпечується тісна взаємодія між мовами й бібліотеками.

Вихідний код, написаний мовою Visual C#, компілюється в проміжну мову (IL) у відповідності зі специфікацією CLI. Код IL і ресурси, такі як растрові зображення й рядки, зберігаються на диску у файлі, що виконується, називаному складанням, з розширенням EXE або DLL у більшості випадків. Складання містить маніфест із відомостями про типи складання, версії, мови й регіональні параметри та вимоги безпеки.

При виконанні програми на Visual C# складання завантажується в середовище CLR залежно від відомостей у маніфесті. Далі, якщо вимоги безпеки дотримані, середовище CLR виконує JIT-компіляцію для перетворення коду IL в інструкції машинного коду. Середовище CLR також надає інші служби, що відносяться до автоматичного збору сміття, обробки виключень і керуванню ресурсами. Код, виконуваний середовищем CLR, іноді називають "керованим кодом" у протиставлення "некерованому коду", що компілюється в машинний код, призначений для певної системи. Далі показані відносини під час компіляції й час виконання між файлами з вихідним кодом Visual C#, бібліотеками класів .NET Framework, складаннями й середовищем CLR.

Взаємодія між мовами є ключовою особливістю .NET Framework. Оскільки код IL, створюваний компілятором Visual C# відповідає специфікації CTS, код IL на основі Visual C# може взаємодіяти з кодом, створюваним версіями мов Visual Basic, Visual C++, Visual J# платформи .NET Framework і ще більш ніж 20 CTS-сумісних мов. В одному складанні може бути кілька модулів, написаних на різних мовах платформи .NET Framework, і типи можуть посилатися один на одного, як якби вони були написані на одній мові.

Крім служб часу виконання, в.NET Framework також є велика бібліотека, що складається з більш ніж 4000 класів, організованих по просторах імен, які забезпечують різноманітні корисні функції для будь-яких дій, починаючи від введення й виведення файлів для керуванням рядками для розбивки XML, і закінчуючи елементами керування Windows Forms. У звичайному застосунку мовою Visual C# бібліотека класів .NET Framework інтенсивно використовується для "устрою" коду.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи хмарного сервісу з використанням алгоритму TDEA.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати

поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис алгоритму 3DES

Triple DES (3DES) – симетричний блоковий шифр, створений на основі алгоритму DES, з метою усунення головного недоліку останнього – малої довжини ключа (56 біт), що може бути зламаний методом повного перебору ключа. Швидкість роботи 3DES в 3 рази нижче, ніж в DES, але криптостійкість набагато вище – час, необхідний для криптоаналізу 3DES, може бути в мільярд раз більше, ніж час, потрібне для розкриття DES. 3DES використовується частіше, ніж DES, що легко ламається за допомогою сьогоднішніх технологій (в 1998 році організація Electronic Frontier Foundation, використовуючи спеціальний комп'ютер DES Cracker, розбила DES за 3 дні). 3DES є простим способом усунення недоліків DES. Алгоритм 3DES побудований на основі DES, тому для його реалізації можливо використовувати програми, створені для DES.

Алгоритм

Схема алгоритму 3DES має такий вид, як на рисунку 3.1. Простий варіант 3DES можна представити так:

$$DES(k_3;DES(k_2;DES(k_1;M))),$$

де k_1, k_2, k_3 – ключі для кожного DES-кроку, M – вхідні дані, які потрібно шифрувати. Це варіант відомий як в EEE, так як три DES операції є шифруванням. Існує 3 типи алгоритму 3DES:

– DES-EEE3: Шифрується три рази із трьома різними ключами (операції шифрування-шифрування-шифрування).

– DES-EDE3: 3DES операції шифрування-розшифрування-шифрування із трьома різними ключами.

– DES-EEE2 і DES-EDE2: Як і попередні, за винятком того, що на першому й третьому кроці використовується однаковий ключ.

Самий популярний різновид 3DES – це DES-EDE3, для нього алгоритм виглядає так:

Шифрування:

$$C = E_{k_3}(E_{k_2}^{\ 1}(E_{k_1}(P)))$$

Розшифрування:

$$P = E_{k_1}^{-1}(E_{k_2}(E_{k_3}^{-1}(C)))$$

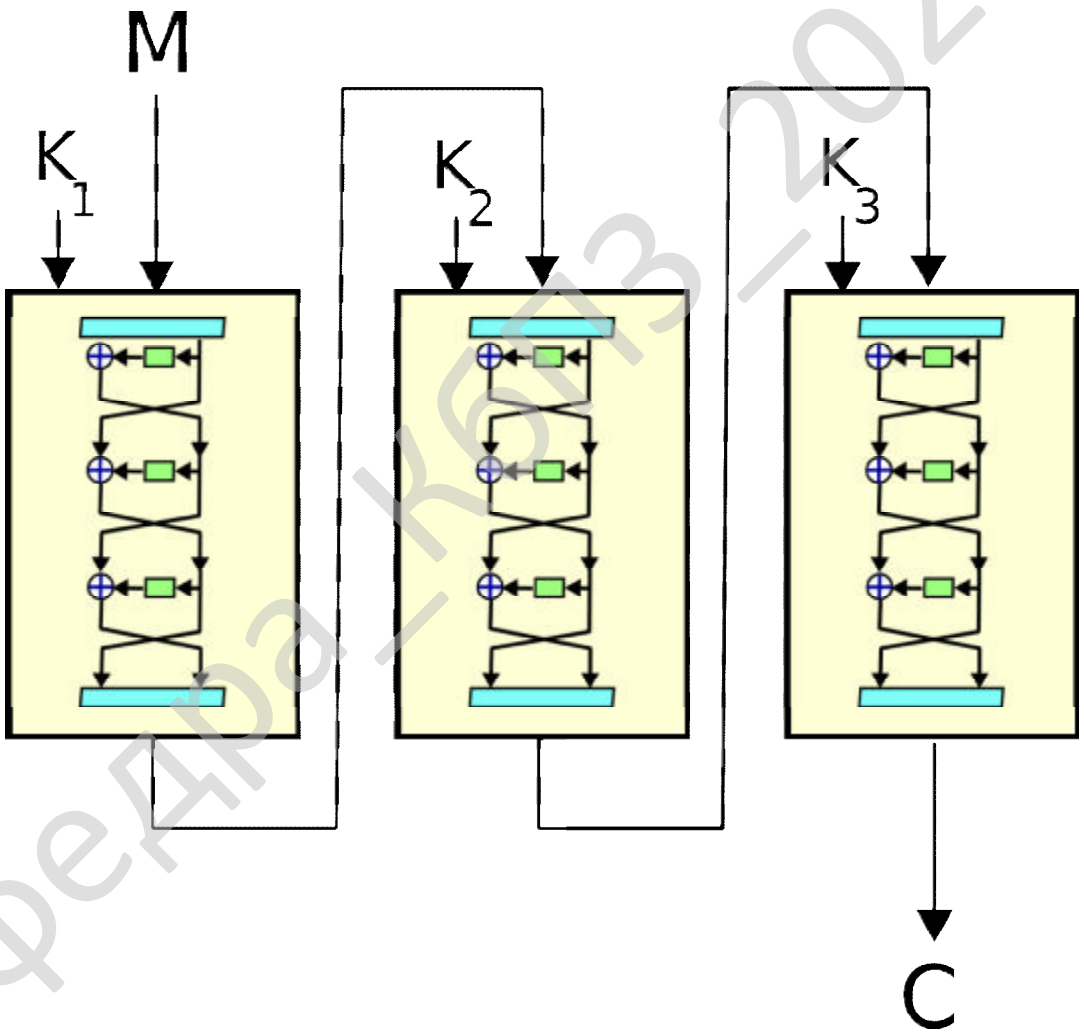

Рисунок 3.1 – Схема 3DES

При виконанні алгоритму 3DES ключі можуть бути обрані так:

– k_1, k_2, k_3 незалежні.

– k_1, k_2 незалежні, а $k_1 = k_3$.

– $k_1 = k_2 = k_3$.

3DES виконує 3 рази алгоритм DES, довжина ключа DES дорівнює 64 біт, а довжина 3DES в 3 рази більше, тобто дорівнює 192 біт. Для DES 64-розрядний ключ ділився на 8 байтів, у кожному байті використовується тільки 7 біт, тому насправді довжина ключа дорівнює 56 біт, а не 64, тому довжина ключа 3DES насправді дорівнює 168, а не 192 біта.

Так як за основу алгоритму 3DES узятий алгоритм DES, то приведемо його опис.

DES

DES (Data Encryption Standard – Стандарт Шифрування Даних) – назва Федерального Стандарту Обробки інформації (FIPS) 46-3, що описує алгоритм шифрування даних (Data Encryption Algorithm – DEA). У термінах ANSI DEA визначений як стандарт X9.32.

DEA – розвиток алгоритму Lucifer, що був розроблений на початку 1970-ых років компанією IBM; на заключних стадіях розробки активна участь приймала NSA і NBS (тепер NIST). З моменту опублікування DEA (більше відомий як DES), широко вивчався й відомий як один із кращих симетричних алгоритмів.

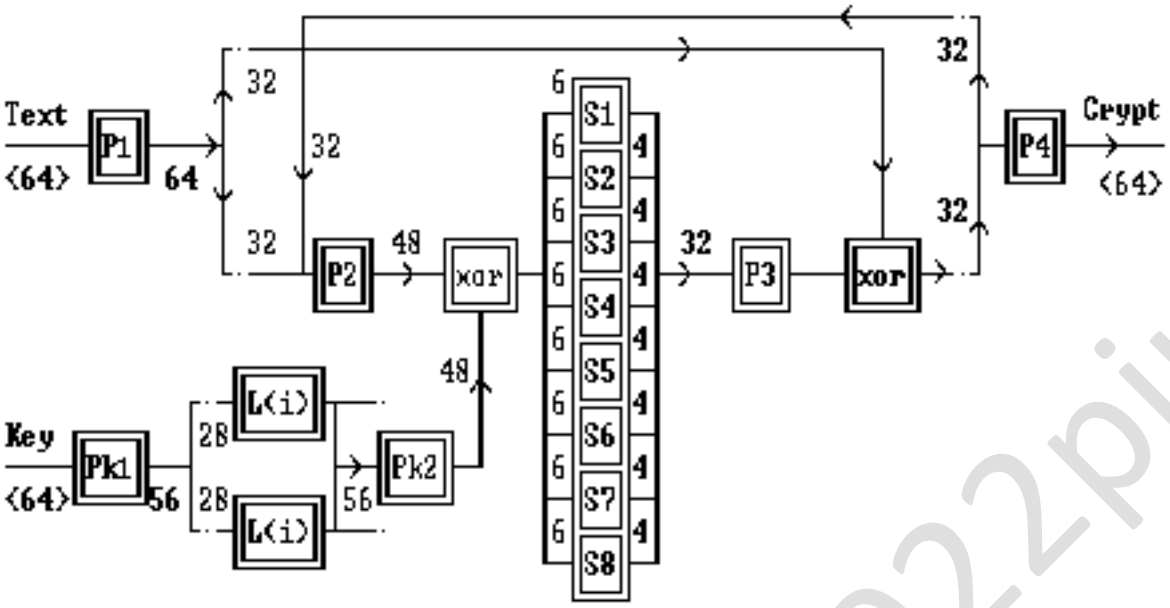

Рисунок 3.2 – Схема алгоритму DES

Де:

– Text – вихідний текст (блок 64 біта).

– Crypt – зашифрований блок.

– Key – 64-х розрядний ключ.

– Числа – розрядність на даній вітці алгоритму.

– P, P*k* – перестановки.

– S – підстановка 6 біт -> 4 біти.

– L(i) – зрушення (i – номер ітерації).

– xor – додавання по модулі 2.

– – конкатенація бітових рядків, причому верхній – попереду.

– – розбивка рядка на два, причому перший – нагорі.

– – обмежена точками ділянка повторюється 16 разів.

Перестановки виконуються по звичайній формулі D[i]=S[P[i]], де:

– S – вихідний рядок (масив символів, нумерація з одиниці).

– D – результат перестановки (масив символів, нумерація з одиниці).

– P – таблиця перестановок (масив індексів у рядку S).

S – підстановка 6->4. У відповідність шести бітам ставиться чотири. Підстановка виробляється за наступним правилом: нехай вихідний бітовий рядок – /abcdef/, тоді /af/ – номер рядка, а /bcde/ – номер стовпця. Рядок і стовпець визначають місцезнаходження результату в S-таблиці. Наприклад, при використанні таблицю S6, число 58 (111010) переводиться в 13 (1101).

Крім звичайного його застосування, цей алгоритм можна використовувати для створення "однобічних" функцій. Для цього ключ і вихідний текст міняються місцями: у формулі Crypt=DES(Text,Key) вихідний текст може бути несекретним і фіксованим. А Crypt розглядається як функція ключа – Key.

DEA оперує блоками 64-бітного розміру й використовує 56-бітний ключ (8 парних біт повного 64-бітного ключа не використовуються). DEA – симетрична криптосистема, визначена як 16-раундовий шифр Фейстеля (Feistel) була спочатку призначена для апаратної реалізації. Коли DEA використовується для передачі інформації, те щоб зашифрувати й розшифрувати повідомлення або щоб створити й перевірити код дійсності повідомлення (MAC) відправник і одержувач повинні знати секретний ключ. DEA може також використовуватися одним користувачем, наприклад для шифрування файлів на жорсткому диску. У багатокористувальницькому середовищі організувати захищений розподіл ключа складно; ідеальне рішення цієї проблеми пропонує криптографія загального ключа.

NIST сертифікує DES (FIPS 46-1, 46-2, 46-3) кожні п'ять років; так, FIPS 46-3 знову сертифікував використання DES у жовтні 1999, але в цей час DES одинарної довжини дозволений тільки для застарілих систем. FIPS 46-3 містить також визначення Triple DES (DES потрійної довжини) (TDEA, відповідно

X9.52). Найближчим часом DES і Triple DES будуть замінені алгоритмом AES (Advanced Encryption Standard – Розширений Стандарт Шифрування).

Алгоритм DES широко застосовується для захисту фінансової інформації: так, модуль THALES (Racal) HSM RG7000 повністю підтримує операції TripleDES для емісії й обробки кредитних карт VISA, EuroPay і інші. Канальні шифратори THALES (Racal) DataDryptor 2000 використовують TripleDES для прозорого шифрування потоків інформації. Також алгоритм DES використовується в багатьох інших пристроях і рішеннях THALES-ESECURITY.

Взлам DES

Незважаючи на багаторічні зусилля дослідників, ефективних атак на DES не виявлено. Очевидний метод атаки – повний перебір всіх можливих ключів; цей процес виконується в середньому 2^{55} кроків. Спочатку передбачалося, що заможний і досвідчений нападаючий може побудувати спеціалізовану ЕОМ, здатну зламати DES, перебравши всі ключі в плині розумного часу. Пізніше Hellman знайшов спосіб удосконалення повного перебору ключів за умови достатнього обсягу пам'яті. Крім того висувалися обвинувачення, що NSA навмисно зробило DES уразливим. Ці міркування дозволяли засумніватися в надійності DES, але незважаючи на все це, ніякого методу злому DES виявлено не було за винятком повного пошуку ключа. Вартість спеціалізованого комп'ютера для виконання такого повного пошуку (за умови знаходження ключа в середньому за 3.5 години) по оцінці Wiener становить один мільйон доларів.

Недавно Wiener уточнив, що тепер рівний за вартістю комп'ютер знайде ключ за 35 мінут. Першу атаку на DES, більше ефективна ніж повний пошук, заявили Biham і Shamir; у ній використовувався новий метод відомий як диференціальний криптоаналіз. Ця атака вимагає шифрування 2^{47} відкритих текстів обраних нападаючим. Теоретично будучи точкою розриву, ця атака непрактична через надмірні вимоги до підбора даних і складності організації атаки по обраному відкритому тексту. Самі автори цієї атаки Biham і Shamir заявили, що вважають DES захищеним.

Раніше Matsui розробив іншу атаку, відому як лінійний крипто аналіз. Цей метод дозволяє відновити ключ DES за допомогою аналізу 2^{43} відомих відкритих текстів. Перший експериментальний криптоаналіз DES, заснований на відкритті Matsui, був успішно виконаний у плині 50 днів на автоматизованих робочих місцях 12 HP 9735.

Зрозуміло, при таких витратах атака як і раніше вважається непрактичною. В одному з недавніх експериментів по злому DES ключ був знайдений за 22 години. По загальній думці криптографів алгоритм DES одинарної довжини вже не є захищеним оскільки на сучасному рівні розвитку обчислювальної техніки 56-бітний ключ став уразливий для повного пошуку. Фактично DES уже заборонений для використання в урядових структурах США й у цей час як стандарт використовується Triple DES (DES потрійної довжини), що буде замінений новим стандартом AES найближчим часом.

Надійне використання DES

Приведемо кілька практичних рекомендацій, що забезпечують безпеку зашифрованих даних. Ключі DES потрібно міняти досить часто, щоб запобігти атакам, що вимагають аналізу досить великої кількості даних. Якщо говорити про захист переданих даних, то необхідно знайти захищений спосіб передачі DES ключа від відправника до одержувача. Обидві ці проблеми вирішуються за допомогою алгоритму RSA або якої-небудь іншої асиметричної криптосистеми: для кожного сеансу зв'язку створюється новий DES ключ, що зашифровується загальним ключем одержувача й у такому виді передається одержувачеві. У таких обставинах криптосистема RSA виступає як інструмент підвищення захищеності DES (або будь-якого іншого секретно-ключового шифру).

Якщо ви використовуєте DES для шифрування файлів на жорсткому диску, то часто міняти ключі малореально, так як для цього необхідно розшифрувати а потім зашифрувати всі файли новим ключем. Замість цього можна створити головний ключ DES, яким буде зашифрований список ключів

DES, використовуваних для шифрування файлів; у цьому випадку головний ключ можна змінювати так часто як це потрібно. Але так як головний ключ більше привабливий для атаки чим окремі DES ключі, те його розумно шифрувати алгоритмом Triple DES.

На практиці використовуються декілька офіційно визначених режимів шифрування DES; кожний із цих режимів має різні властивості. Режим ECB (електронна кодова книга) послідовно шифрує кожний 64-бітний блок відкритого тексту тим самим 56-бітним ключем DES. У режимі CBC (формування ланцюжка блоку шифру) кожний 64-бітний блок відкритого тексту перед шифруванням DES ключем логічно підсумується (XOR) з попереднім блоком зашифрованого тексту. Таким чином, шифрування кожного наступного блоку залежить від попередніх блоків і тому той самий 64-бітний блок відкритого тексту може бути представлений різними блокам зашифрованого тексту залежно від його розміщення у вихідному повідомленні. CBC дозволяє захиститися від деяких атак, але не від повного пошуку або диференціального криптоаналізу. Режим CFB (шифрування зі зворотним зв'язком) дозволяє використовувати DES блоки довжиною менш 64 біт. Режим OFB по суті дозволяє використовувати DES як потоковий шифр. З перерахованих режимів DES найбільше широко на практиці використовується режим CBC, що є частиною декількох стандартів. Для підвищення захисту можна використовувати режим CBC з потрійним шифруванням.

Стійкість ключа DES

Для шифру DES існують чотири слабких ключі k при яких $Ek\ (Ek\ (m)) = m$; також існують дванадцять частково слабких ключів, які становлять пари $k1$ і $k2$ такі що $Ek1\ (Ek2\ (m)) = m$. Так як кількість всіх можливих ключів DES становить 2^{56}, то шанс випадково вибрати слабкий або частково слабкий ключ становить $1/(2^{52})$. Оскільки користувач вибирає ключ навмання, то слабкі ключі можна ігнорувати. Незважаючи на це, деякі користувачі воліють перевіряти, чи є

використовуваний ключ DES слабким ключем. Таке випробування не відніме багато часу й не впливає на швидкість шифрування.

Надійність ключів DES

Питання полягає в тому, що для двох довільних ключів $k1$ і $k2$ може існувати третій ключ k такий що шифрування двома першими ключами еквівалентно шифруванню третім ключем тобто $Ek\ (m) = Ek1\ (Ek2\ (m))$ для всіх повідомлень m. Якщо це так, то множина всіх ключів утворить абстрактну групу, де закон сполучення $k1$ і $k2$ заміщається законом $k.$, що досить знижує захищеність DES, оскільки допускає атаку "зустріч усередині", при якій ключ DES може бути знайдений у приблизно за 2^{28} кроків замість звичайних 2^{56}. Також при цьому стає марним множинне (подвійне й потрійне) шифрування двома різними ключами оскільки воно буде еквівалентно одинарному шифруванню деяким третім ключем. Однак, ключі DES не групуються. Ця проблема, очевидна спочатку, була зрештою вирішена в 1993 році. У результаті було встановлено, що методи подібні до потрійного шифрування дійсно збільшують захищеність DES. Проблема може бути сформульована в такий спосіб: нехай M позначає набір всіх можливих повідомлень і нехай K позначає набір всіх можливих ключів. Шифрування ключем k приналежної K виконується перестановкою Ek: M -> M. Набір $EK = \{Ek: k$ належить $K\}$ таких перестановок є під-набором групи SM всіх перестановок M -> M. Той факт, що EK (тобто DES) – не група, є тільки факт того, що EK створює підгрупу CM, що є більшої чим набір EK. Фактично, розмір цієї підгрупи – принаймні 2^{83}. Зокрема багаторазове шифрування задає більший простір ключів.

Криптостійкість 3DES

3DES з різними ключами має довжину ключа рівну 168 біт, але через атаки «зустріч у середині» (in-the-middle) ефективна криптостійкість становить тільки 112 біт. У варіанті DES-EDE, у якому $k_1 = k_3$, ефективний ключ має довжину 80 біт. Для успішної атаки на 3DES буде потрібно близько 2^{32} біт

відомого відкритого тексту, 2^{113} кроків, 2^{90} циклів DES-шифрування й 2^{88} біт пам'яті.

Застосування 3DES

3DES із трьома ключами реалізований у багатьох додатках, орієнтованих на роботу з Інтернет, у тому числі в PGP і S/mime. Потрійний DES є досить популярною альтернативою DES і використовується при керуванні ключами в стандартах ANSI X9.17 і ISO 8732 і в PEM (Privacy Enhanced Mail). Відомих криптографічних атак, застосовних на практиці, на 3DES не існує.

Проте, 3DES (який ще позначають як TDES) потроху виходить із уживання, замінний новим алгоритмом AES Rijndael. Rijndael, реалізований програмно, працює в шість разів швидше. Тому 3DES більше підходить для апаратних реалізацій.

3.2 Розробка структурної схеми

На рисунку 3.3 зображена структурна схема системи хмарного сервісу з використанням алгоритму TDEA.

Забезпечення безпеки інформації при зберіганні й обробці більших інформаційних масивів – одна із самих актуальних проблем сучасних інформаційних технологій. Інтенсивний розвиток методів розподіленої обробки даних і різке збільшення обсягів інформації, що накопичується в комп'ютерних системах, привели останнім часом до кардинальної зміни методів довгострокового зберігання даних. Традиційні підходи до організації зберігання більших інформаційних масивів перестали задовольняти зрослим вимогам до ємності носіїв і швидкості доступу до даних. Всі частіше споживач довіряє зберігання своєї власної інформації зовнішнім центрам або мережам зберігання даних (так званий аутсорсинг). Одна з основних сфер застосування мережного зберігання даних – формування банків даних електронних документів, а також електронних архівів і бібліотек. Такі сховища даних можуть бути як публічними, так і обмеженого доступу, залежно від характеру документів, що накопичуються

в них. Нерідко перед приміщенням документів у мережні сховища вони піддаються стиску або іншим спеціальним видам кодування. У зв'язку із цим загострюється необхідність забезпечення керованості, надійності й безпеці зберігання й доступу до електронних документів, а також процедур їхньої передачі між прикладними програмами й пристроями зберігання.

Якщо навіть дані зберігаються локально, виникає інша проблема: адміністратори, що управляють СУБД і персонал так чи інакше звичайно мають права доступу до всієї збереженої інформації. Для захисту від їхніх несанкціонованих дій у деяких випадках доцільне застосування апаратно-програмних засобів шифрування даних перед записом їх на засоби зберігання. Часто шифрувальні модулі вбудовуються, наприклад, у засоби резервного копіювання даних. Однак при зберіганні шифрованих масивів утруднений пошук окремих файлів і оперативний доступ до елементів масиву, необхідним для роботи прикладних програм. Тому що масив зберігається в зашифрованому виді, і серверу, на якому він зберігається (або СУБД), не можуть бути довірені ключі шифрування, користувач (або прикладна програма від його ім'я) змушений завантажувати копії всіх файлів масиву, розшифровувати їх і потім виконувати пошук на локальній машині. Очевидно, що такий спосіб пошуку дуже неекономічний. У зв'язку із цим вимальовується проблема забезпечення можливості пошуку даних по шифрованим і (або) стислим даним, що може бути конкретизована залежно від застосовуваної в системі моделі шифрування даних.

Для шифрування великих масивів даних, що поміщаються в зовнішні стосовно власника інформації сховища, ефективні лише симетричні схеми шифрування. Можливості їхнього практичного застосування, мабуть, визначаються можливостями організації схеми керування секретними ключами, для яких необхідно забезпечити виконання двох почасти суперечливих вимог: забезпечення високої схоронності ключів (зокрема, за рахунок резервування) і обмеження середовища їхнього поширення тільки тими пристроями, яким довіряє власник інформації.

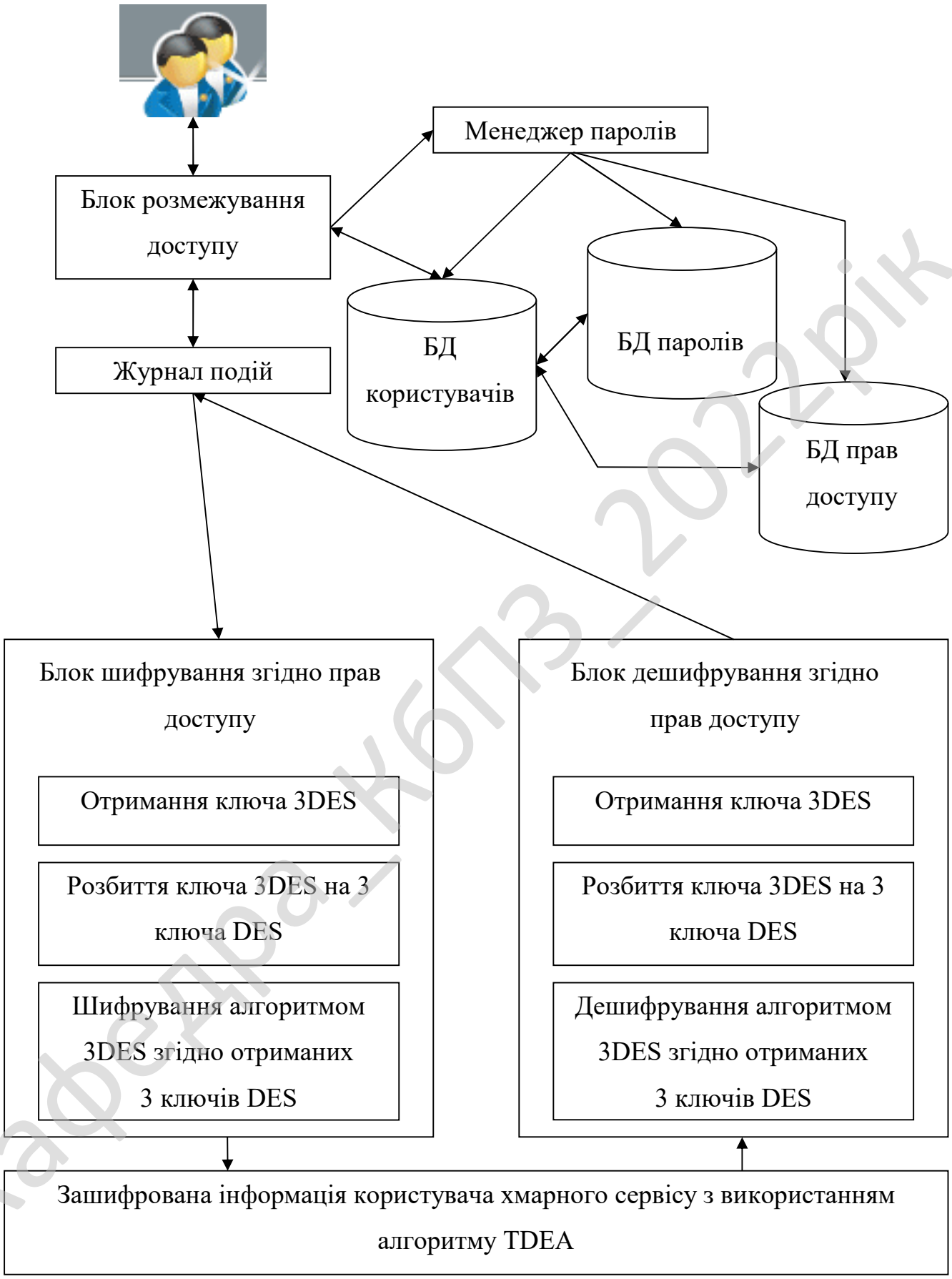

Рисунок 3.3 – Структурна схема системи

У зв'язку із цим у деяких випадках більше раціональним виглядає застосування схем відкритого шифрування, що дає можливість невизначеному колу осіб поміщати свої дані в сховище, але доступ до них залишати лише для власників секретного ключа. Така схема може бути корисна, наприклад, для систем електронної пошти або систем планування потоків завдань (workflows), де циркулюють переважно повідомлення невеликої довжини. Для таких схем тим більше необхідні механізми пошуку за шифрованим даними, що операції розшифрування в асиметричних криптосхемах, як відомо, виконуються на кілька порядків повільніше в порівнянні із симетричними.

Інша проблема, пов'язана із забезпеченням конфіденційності пошуку в масивах даних, пов'язана з бажанням унеможливити одержання адміністратором СУБД і сторонніми особами відомостей про те, до яких саме записів (або фрагментів) бази даних здійснювався доступ при кожному конкретному запиті. У закордонній літературі це завдання зветься “Private Information Retrieval” (PIR). Вона особливо актуальна, наприклад, при обробці й зберіганні електронних документів, що містять відомості приватного характеру: фінансові, юридичні, майнові, медичні й інші.

Якщо навіть самі поля бази даних зашифровані, характер і частота запитів до них уже можуть дати зловмисникові деяку непряму інформацію, розголошення якої небажано для власника. Ці завдання до визначеної міри аналогічні виникаючої в телекомунікаційних системах завданню маскування інтенсивності трафіка між вузлами, що, як відомо, вирішується шляхом суцільного заповнення каналу псевдовипадковими послідовностями.

Виходячи зі структурної схеми системи зображеної на рисунку 3.3, система хмарного сервісу з використанням алгоритму TDEA, працює наступним чином.

Спершу при вході в систему, користувач звертається до блоку розмежування доступу.

Блок розмежування доступу отримує пароль користувача, та звертається до менеджеру паролів, де отримує сеансовий пароль перевірки правильності

паролю користувача, та правильності прав доступу користувача, які зберігаються у відповідних зашифрованих базах даних.

Розмежування цих баз зроблено з метою підвищення стійкості системи зберігання інформації.

Після підтвердження прав доступу, та правильності введеного паролю, користувачеві видається сеансовий ключ 3DES для роботи з інформацією.

У блоці шифрування згідно прав доступу, з отриманого ключа 3DES добуваються 3 ключа алгоритму DES, за допомогою яких й відбувається шифрування інформації алгоритмом 3DES.

Процедура дешифрування відбувається аналогічним чином.

3.3 Розробка функціональної схеми

На рисунку 3.4 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема складається з наступних блоків:

– Головне вікно програми.

– Блок розмежування доступу.

– Блок менеджеру паролів.

– Блок журналювання подій.

– Допомога.

– Блоки шифрування та дешифрування інформації згідно алгоритму 3DES.

Розглянемо ці блоки більш детально.

Головне вікно програми. Головне вікно призначене для швидкого доступу до основних функцій програми й меню. Програма складається з головного вікна, розташованого у верхній частині екрана й набору незалежних дочірніх вікон. Розташування й розміри вікон можна змінювати за допомогою миші. Також існує можливість закрити непотрібні дочірні вікна (знову відобразити їх можна шляхом вибору відповідних пунктів у меню натисканням на аналогічні кнопки в головному вікні програми). Всі зроблені зміни

збережуться в наступному сеансі роботи. Призначення всіх кнопок у програмі пояснюється спливаючими підказками: підведіть покажчик миші до будь-якої кнопки й затримаєте його – з'явиться спливаюча підказка із призначенням кнопки. Головне меню надає доступ до основних списків і функцій системи.

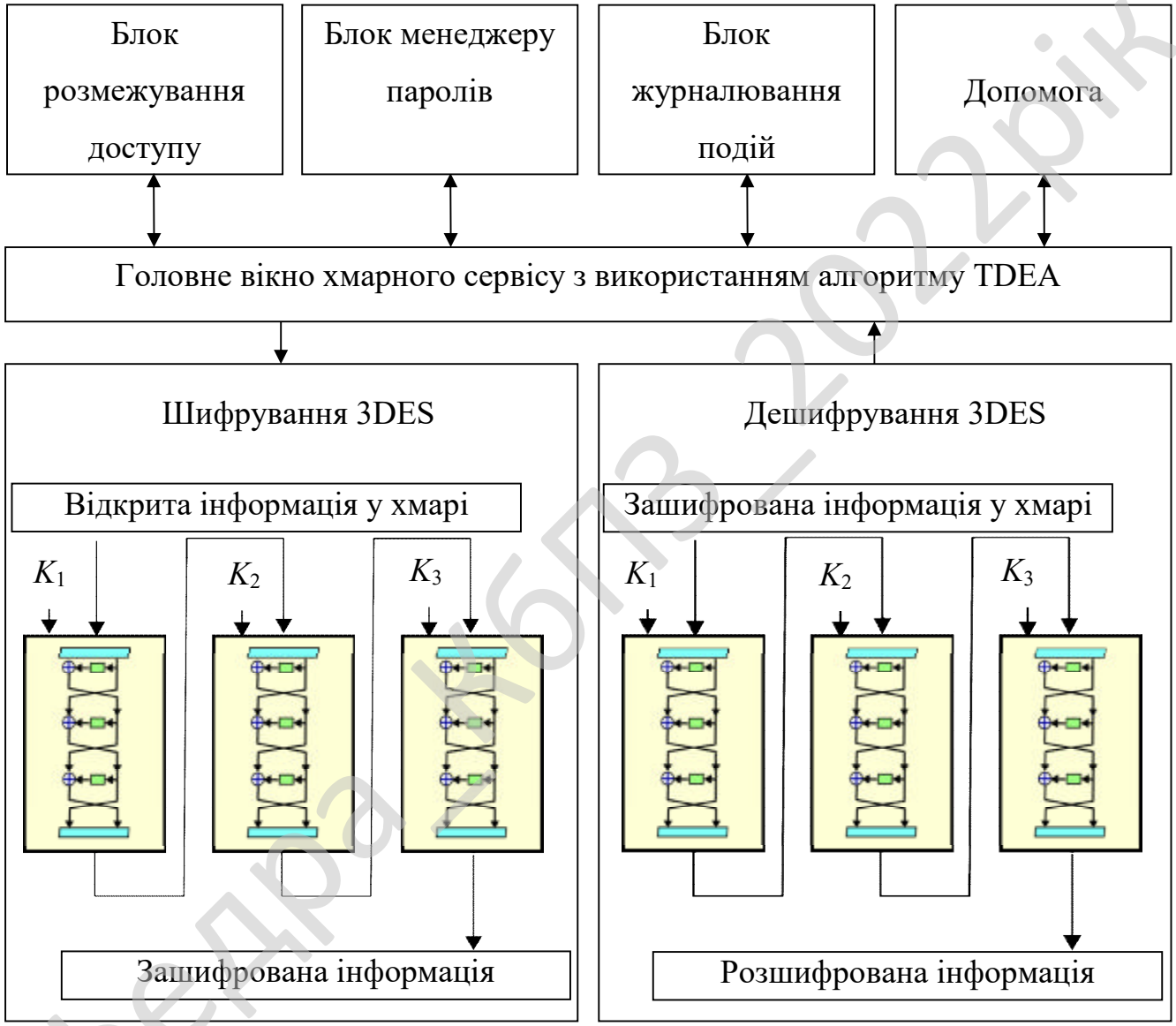

Рисунок 3.4 – Функціональна схема системи

Блок розмежування доступу. Призначений для організації безпечного доступу співтовариства користувачів до захищених ресурсів. Члени цього

співтовариства, використовуючи програму, одержують визначені переваги. Це дає наступні можливості:

– Надавати користувачам доступ до інформації (наприклад, групи структурних схем, адресні довідники відділів або пошук співробітників) і ресурсам (наприклад, устаткування або облікові записи у внутрішніх системах), у яких вони бідують, буквально з першого дня.

– Синхронізувати кілька паролів з одним ім'ям користувача для всіх систем.

– При необхідності оперативно змінювати або відзивати права на доступ (наприклад, при переході співробітника в іншу групу або при звільненні).

– Підтримувати відповідність урядовим постановам.

У цей блок включені наступні можливості.

Самообслуговування облікового запису, що дозволяє:

– відображати структурні схеми;

– повідомляти про додатки, пов'язані з користувачем, для адміністратора;

– змінювати дані профілю;

– виконувати пошук у каталозі;

– змінювати пароль, відповідь на запит-відповідь пароля і його підказку;

– переглядати стан політики й синхронізації пароля;

– створювати облікові записи для нових користувачів і груп (при наявності відповідних повноважень).

Запити й твердження, що дозволяють:

– запитувати ресурси;

– перевіряти підтвердження запитів на ресурси;

– працювати із призначеними завданнями підтвердження інших запитів на ресурси;

– виконувати запити й твердження в якості чиєїсь довіреної особи або делегата;

– призначати кого-небудь ще довіреною особою або делегатом (при наявності відповідних повноважень);

– управляти всіма цими функціями запитів і підтверджень в інтересах Вашої групи (при наявності відповідних повноважень);

– при необхідності для кожного запиту або підтвердження надавати цифровий підпис.

Ролі, що дозволяють виконувати наступні дії:

– запитувати призначення ролей і управляти процесом підтвердження запитів на призначення ролей;

– перевіряти стан Ваших запитів ролей;

– визначати ролі і їхні взаємини;

– визначати обмеження поділу обов'язків (SoD) і управляти процесом підтвердження у випадках, коли користувач запитує перевизначення обмеження;

– переглядати довідник ролей;

– переглядати докладні звіти, у яких перераховані ролі й обмеження поділу обов'язків, визначені в довіднику, а також поточний стан призначення ролей, виключення поділу обов'язків і повноваження користувача.

Модуль "Дотримання" дозволяє:

– Запитувати підтвердження профілю користувача.

– Запитувати підтвердження поділу обов'язків (SoD).

– Запитувати підтвердження призначення функцій.

– Запитувати підтвердження призначення користувача.

Блок менеджеру паролів. Надає можливості не тільки для простого збереження паролів, але й для повноцінної роботи з ними. Програма підтримує роботу з декількома аккаунтами, і працювати з нею можуть трохи користувачів. При цьому бази даних кожного користувача шифруються.

Додаткові можливості:

– Система пошуку по базі даних.

– Підтримка макросів.

– Можливість резервного копіювання бази даних.

– Можливість швидкого перемикання між користувачами.

– Швидкий доступ до часто використовуваних функцій.

– Генератор паролів.

– Можливість роздруківки паролів.

Блок журналювання подій. Призначений для запису у журнал усіх подій, які відбуваються у системі. Журнал дій користувачів містить форму для запуску архівації журналу. Форма архівації являє собою кнопку "Очистити журнал" і поле з датою "по:". Дату можна встановлювати будь-яку, але не раніше, ніж поточна дата мінус 1 місяць, щоб у системі завжди зберігалися дані про дії користувачів як мінімум за місяць.

Після натискання кнопки "Очистити журнал" у Системі генерується текстовий файл із архівом журналу за обраний період. Файл зберігається в зашифрованому виді, а посилання на цей файл показуються адміністраторові. Після створення файлу запису журналу за обраний період віддаляються з бази даних. У випадку помилки при створенні або збереженні файлу, записи не видаляються.

Допомога. Блок призначений про надання допомоги по роботі з системою, а також для надання інформації про розробників системи, версію та дату випуску.

Блоки шифрування та дешифрування інформації згідно алгоритму 3DES. Призначені для шифрування та дешифрування інформації, до якої користувач має доступ, згідно прав доступу.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.5. З нього ми бачимо, що

робота програмного продукту починається з запуску процесу початку/кінця роботи програми.

Цей процес взаємодіє з наступними процесами:

– Процесом створення облікових засобів.

– Процесом роботи з конфіденційними даними.

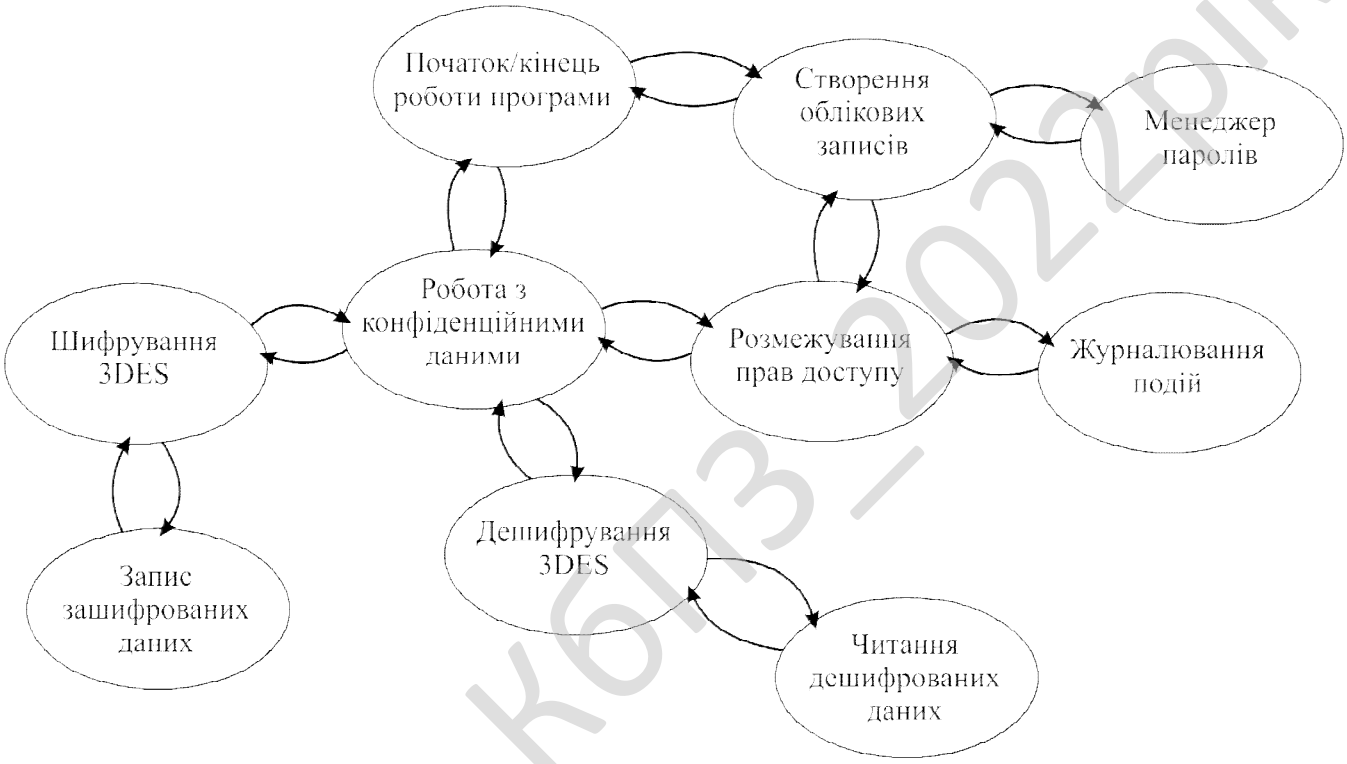

Рисунок 3.5 – Діаграма процесів системи

Процес створення облікових засобів взаємодіє з наступними процесами:

– Процесом менеджеру паролів.

– Процесом розмежування прав доступу.

Процес роботи з конфіденційними даними взаємодіє з наступними процесами:

– Процесом розмежування прав доступу.

– Процесом шифрування 3DES.

– Процесом дешифрування 3DES.

Процес шифрування 3DES взаємодіє з процесом запису зашифрованих даних.

Процес дешифрування 3DES взаємодіє з процесом читання дешифрованих даних.

На цьому програмний продукт закінчує свою роботу.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків.

Спершу виводиться головне вікно програми.

Після цього визначається, чи є потреба у створені нового облікового запису.

Якщо така потреба є, тоді відбувається створення облікового запису та його додавання у БД.

Після цього відбувається генерація та збереження ключів.

Наступним кроком є визначення того, чи необхідно записати конфіденційні дані.

Якщо потрібно, тоді відбувається виконання наступних дій:

– Виводяться дані та пароль.

– Відбувається шифрування даних алгоритмом 3DES, та їхнє збереження.

Після цього користувач визначає необхідність читання конфіденційних даних.

Для цього він реалізує наступну послідовність дій:

– Введення паролю.

– Дешифрування даних алгоритмом 3DES.

– Виведення даних.

Після цього користувач обирає працювати йому далі з системою, або ні.

Якщо він обирає, що ні, тоді програма закінчує свою роботу.

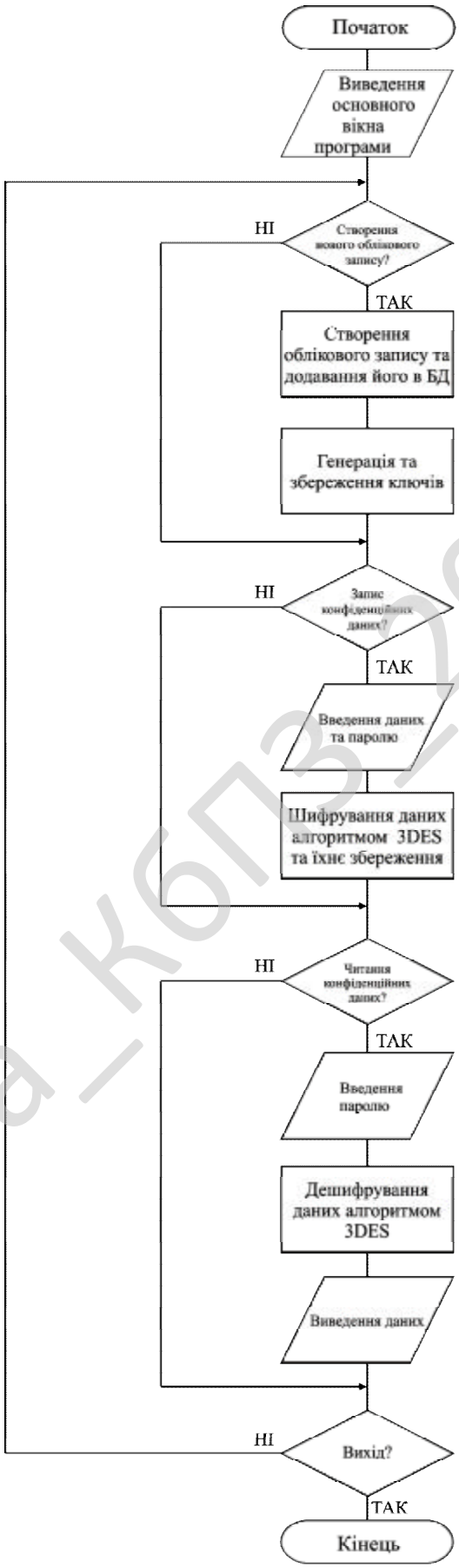

Рисунок 4.1 – Блок-схема роботи основної програми

Блок-схема роботи підпрограми шифрування/дешифрування даних зображена на рисунку 4.2.

Вона складається з виконання наступних кроків.

Спершу користувач обирає необхідність шифрування, згідно прав доступу.

Якщо шифрувати необхідно, тоді відбуваються наступні дії:

– Отримання ключа 3DES.

– Розбиття ключа 3DES на три ключа DES.

– Шифрування даних першим ключем.

– Шифрування даних другим ключем.

– Шифрування даних третім ключем.

– Створення файлу з зашифрованими даними.

У разі, якщо користувач обирає дешифрування, тоді відбувається виконання наступних дій:

– Отримання ключа 3DES.

– Розбиття ключа 3DES на три ключа DES.

– Дешифрування даних першим ключем.

– Дешифрування даних другим ключем.

– Дешифрування даних третім ключем.

– Створення файлу з дешифрованими даними.

На цьому підпрограма шифрування/дешифрування даних закінчує свою роботу.

Вкладка «Шифрування»

Тут ми можемо вибрати текст із файлу, або ж надрукувати текст в `TextBox` для тексту. Якщо ви відкриваєте текст із текстового файлу (*.doc не підтримуються), тоді текст із файлу автоматично відобразиться в `TextBox` для тексту.

Рисунок 4.2 – Блок-схема роботи підпрограми шифрування/дешифрування даних

Пароль – потрібний фіксованої довжини, котра дорівнює 24 буквам. На одну букву доводиться 8 байт пам'яті, і в підсумку пароль виходить довгої в 24 * 8 = 192 байта. У нашім випадку може використовуватися пароль кожної довжини, якщо тільки він менше 24 символів. У моїй програмі короткий пароль дублюється й дописується до уже існуючого кілька разів. Тобто ви ввели «ключ», а програма зробить «ключключключключключ».

Вектор ініціалізації (IV) – потрібний для завдання параметрів блокового шифрування. Він генерується при шифруванні, а також генерується зовсім іншим при дешифруванні. Тому ми запам'ятовуємо його в локальної змінної `IVector`.

```
protected byte[] IVector = null;
```

Шлях куди шифруємо – шлях, куди зберігаємо текстовий файл.

Вкладка «Дешифрування»

Шлях до файлу – шлях до файлу для дешифрування. Вектор (IV) – беремо зі змінної `IVector`. Для наочності відображає його через мітку `label9`.

```
// Створюємо новий TripleDESCryptoServiceProvider об’єкт
// для генерування вектора ініціалізації (IV).
TripleDESCryptoServiceProvider tDESalg = new TripleDESCryptoServiceProvider();
// Для наочності виводимо значення вектора ініціалізації
string temp = null;
for (int i = 0; i < tDESalg.IV.Length; i++)
{
    temp += tDESalg.IV[i].ToString();
}
tbVector.Text = temp;
lVector.Text = temp;
// Запам'ятовуємо вектор у локальної змінної
IVector IVector = tDESalg.IV;
```

Дешифрована інформація – наш текст після дешифрування.

Підведемо підсумки – щоб зашифрувати інформацію алгоритмом TripleDES, потрібний пароль (ключ) і вектор ініціалізації. Щоб розшифрувати інформацію, потрібні той же пароль і вектор ініціалізації.

"Шифрування"

Програмний код для оброблювачів подій:

— button1

```
OpenFileDialog ofd = new OpenFileDialog();
ofd.Filter = "Txt files (*.txt)|*.txt|All files (*.*)|*.*";
ofd.ShowDialog();
if (ofd.FileName != "")
{
    tbText.Text = new FileInfo(ofd.FileName).OpenText().ReadToEnd();
}
```

Замітки: Тут ми замість використання контрола `OpenFileDialog` прямо створюємо екземпляр класу `OpenFileDialog`. Також використовуємо властивість `Filter` для відкриття тільки текстових файлів. Вміст вікна `OpenFileDialog` відфільтровується так, що будуть показані тільки папки й текстові файли. Далі ми одним рядком відкриваємо текстовий файл і читаємо його з початку в кінець. Тут я використовував посилання на об'єкт `FileInfo`.

— button2

```
SaveFileDialog sfd = new SaveFileDialog();
sfd.InitialDirectory = "C:\\";
sfd.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
sfd.ShowDialog();
tbPath.Text = sfd.FileName;
```

Той же код, але для збереження.

— button3

```
try
            {
                // Створюємо новий TripleDESCryptoServiceProvider об’єкт
                // для генерування вектора ініціалізації (IV).
                TripleDESCryptoServiceProvider tDESalg = new
TripleDESCryptoServiceProvider();
                // Для наочності виводимо значення вектора ініціалізації
                string temp = null;
                for (int i = 0; i < tDESalg.IV.Length; i++)
                {
                    temp += tDESalg.IV[i].ToString();
                }
                textBox4.Text = temp;
```

```
            label9.Text = temp;
            // Запам'ятовуємо вектор у локальної змінної IVector
            IVector = tDESalg.IV;
            // Рядок для шифрування
            string sData = textBox1.Text;
            // Одержуємо ключ, перетворюємо в масив байтів і
            // доповнюємо до довжини 24. Не більше й не менше.
            string strKey = textBox2.Text;
            byte[] bKey = new byte[24];
            for (int i = 0, j = 0; i < 24; i++, j++)
            {
                if (j == strKey.Length)
                    j = 0;
                bKey[i] = (byte)strKey[j];
            }
            // Указуємо файл, куди будемо шифрувати
            string FileName = textBox3.Text;
            // Шифруємо текст у файл.
            // При цьому вказуємо ім'я файлу, ключ і вектор ініціалізації.
            CryptText.EncryptTextToFile(sData, FileName, bKey, tDESalg.IV);
            // Повідомляємо про результат
            MessageBox.Show("Текст успішно зашифрований !!!");
        }
        catch (Exception exc)
        {
            MessageBox.Show(exc.Message);
        }
```

Замітки. Для використання класу:

```
TripleDESCryptoServiceProvider,
```

потрібно підключить простір імен:

```
using System.Security.Cryptography.
```

Для зберігання вектора ініціалізації я використовував змінну IVector. Визначається вона відразу після:

```
protected byte[] IVector = null;
```

Тут використовується метод `EncryptTextToFile()` із класу `CryptText`. Як я вже говорив, цей клас перебуває в окремому модулі (або окремому файлі). Також використовується конструкція `try – catch` для обробки виключень і їхнього виводу на екран у вигляді `MessageBox`.

"Дешифрування"

Програмний код для оброблювачів подій:

— **button4**

```
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Filter = "Txt files (*.txt)|*.txt|All files (*.*)|*.*";
            ofd.ShowDialog();
            if (ofd.FileName != "")
            {
                textBox5.Text = new FileInfo(ofd.FileName).FullName;
            }
```

– **button5**

```
try
            {
                // Створюємо новий TripleDESCryptoServiceProvider об'єкт
                // для генерування вектора ініціалізації (IV).
                TripleDESCryptoServiceProvider tDESalg = new
TripleDESCryptoServiceProvider();
                // Одержуємо ключ, перетворюємо в масив байтів і
                // доповнюємо до довжини 24. Не більше й не менше.
                string strKey = textBox6.Text;
                byte[] bKey = new byte[24];
                for (int i = 0, j = 0; i < 24; i++, j++)
                {
                    if (j == strKey.Length)
                        j = 0;
                    bKey[i] = (byte)strKey[j];
                }
                // Указуємо файл для дешифрування
                string FileName = textBox5.Text;
                // Дешифруємо текст із файлу.
                // При цьому вказуємо ім'я файлу, ключ і вектор ініціалізації
IVector.
                textBox7.Clear();
                textBox7.Text += CryptText.DecryptTextFromFile(FileName, bKey,
IVector);
                // Повідомляємо про результат
                MessageBox.Show("Текст успішно дешифрований !!!");
            }
            catch (Exception exc)
            {
                MessageBox.Show(exc.Message);
```

```
        }
```

Тепер додаємо в проект клас у такий спосіб: відкриваємо `SolutionExplorer`, клацаємо по `MyTripleDES` правою кнопкою мишки й тиснемо додати-клас (`Add-Class`). Називаємо його `CryptText`.

Не забуваємо підключити в наш тільки що створений клас

```
using System.Security.Cryptography;
using System.IO;
using System.Windows.Forms;
```

Далі в клас вставляємо код:

```
        /// <summary>
        /// Метод шифрує текст у файл
        /// </summary>
        /// <param name="Data">Текст для шифрування</param>
        /// <param name="FileName">Повний шлях до файлу</param>
        /// <param name="Key">Ключ (пароль)</param>
        /// <param name="IV">Вектор ініціалізації</param>
        public static void EncryptTextToFile(String Data, String FileName, byte[]
Key, byte[] IV)
        {
            try
            {
                // Створює або відкриває визначений файл.
                FileStream fStream = File.Open(FileName, FileMode.OpenOrCreate);
                // Створює CryptoStream який використовує FileStream
                // та ключ шифрування й ініціалізує вектор(IV).
                CryptoStream cStream = new CryptoStream(fStream,
                    new TripleDESCryptoServiceProvider().CreateEncryptor(Key, IV),
                     CryptoStreamMode.Write);
                    // Створює StreamWriter який використовує CryptoStream.
                    StreamWriter sWriter = new StreamWriter(cStream);
                    // Записує дані до потоку
                    // для їх шифрування.
                    sWriter.WriteLine(Data);
                    // Закриваємо потік
                    // закриваємо файл.
                    sWriter.Close();
                    cStream.Close();
                    fStream.Close();
            }
            catch (CryptographicException e)
```

```
            {
                MessageBox.Show("Відбулася криптографічна помилка: {0}",
e.Message);
            }
            catch (UnauthorizedAccessException e)
            {
                MessageBox.Show("Відбулася помилка доступу до файлу: {0}",
e.Message);
            }
        }
        /// <summary>
        /// Метод дешифрує текст із файлу
        /// </summary>
        /// <param name="FileName">Повний шлях до файлу</param>
        /// <param name="Key">Ключ (пароль)</param>
        /// <param name="IV">Вектор ініціалізації</param>
        /// <returns>Текст у дешифрованому виді</returns>
        public static string DecryptTextFromFile(String FileName, byte[] Key,
byte[] IV)
        {
            try
            {
                // Створює або відкриває визначений файл.
                FileStream fStream = File.Open(FileName, FileMode.OpenOrCreate);
                // Створює CryptoStream який використовує FileStream
                // та ключ шифрування й ініціалізує вектор(IV).
                CryptoStream cStream = new CryptoStream(fStream,
                    new TripleDESCryptoServiceProvider().CreateDecryptor(Key, IV),
                    CryptoStreamMode.Read);
                // Створює StreamReader який використовує CryptoStream.
                StreamReader sReader = new StreamReader(cStream);
                // Читаємо файли з потоку
                // для їх дешифрування.
                string val = sReader.ReadToEnd();
                // Закриваємо потік
                // закриваємо файл.
                sReader.Close();
                cStream.Close();
                fStream.Close();
                // Повертаємо рядок.
                return val;
            }
```

```
            catch (CryptographicException e)
            {
                MessageBox.Show("Відбулася криптографічна помилка: {0}",
e.Message);
                return null;
            }
            catch (UnauthorizedAccessException e)
            {
                MessageBox.Show("Відбулася помилка доступу до файлу: {0}",
e.Message);
                return null;
            }
        }
    }
}
```

Створюємо оброблювач події на завантаження форми Form1. Вставляємо код:

```
textBox3.Text = @"C:\Encoded.txt";
textBox2.Text = "key";
textBox6.Text = "key";
```

Видаляємо простори імен, які не використовуємо. Для цього правою кнопкою мишки клацаємо по коду й тиснемо `Organize Usings – Remove Unused Usings`.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм MMB, в основі якого лежить змішування операцій різних алгебраїчних груп. MMB – ітеративний алгоритм, що складається з лінійних дій (XOR і використання ключа) і паралельного застосування чотирьох великих оборотних нелінійних підстановок. Ці підстановки визначаються за допомогою множення по модулю 2^{32}-1 з постійними множниками. У підсумку з'являється алгоритм, що використовує 128-бітовий ключ і 128-бітовий блок.

Алгоритм MMB оперує 32-бітовими підблоками тексту (x_0, x_1, x_2, x_3) і 32-бітовими підблоками ключу (k_0, k_1, k_2, k_3). Це спрощує реалізацію алгоритму на

сучасних 64-бітових процесорах. Чергуючись із операцією XOR, шість разів використовується нелінійна функція *f*. Запишемо операції алгоритму (всі операції з індексами виконуються по модулю 4):

$$x_i = x_i \oplus k_i \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+1} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+2} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_i \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+1} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+2} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

Функція *f* виконується в три кроки:

1. $x_i = c_i * x_i$ для $i = 0..3$ (Якщо на вході множення одні одиниці, то на виході – теж одні одиниці).

2. Якщо молодший значущий біт $x_0 = 1$, то $x_0 = x_0 \oplus C$. Якщо молодший значущий байт $x_3 = 0$, то $x_3 = x_3 \oplus C$.

3. $x_i = x_{i-1} \oplus x_i \oplus x_{i+1}$ для $i = 0..3$.

Всі операції з індексами виконуються по модулю 4. Операція множення на кроці 1 виконується по модулі 2^{32}-1. Спеціальний випадок для даного алгоритму: якщо другий операнд дорівнює 2^{32}-1, результат теж дорівнює 2^{32}-1. В алгоритмі використовуються наступні константи:

$$C = 2aaaaaaa,\ c_0 = 025f1cdb,\ c_1 = 2*c_0,\ c_2 = 2^3 * c_0,\ c_3 = 2^7 * c_0.$$

Константа *C* – «найпростіша» константа без кругової симетрії, високою трійковою вагою й нульовим молодшим значущим бітом. У константи c_0 є інші

особливі характеристики. Константи c_1, c_2 і c_3 – зрушені версії c_0, і служать для запобігання атак, заснованих на симетрії.

Розшифрування виконується у зворотному порядку, Етапи 2 і 3 інверсні їм самим. На етапі 1 замість c_i використовується c_i^{-1}. Значення c_0^{-1} = 0dad4694.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

У розробленій системі зберігання даних з розмежованим доступом, існують наступні користувальницькі меню:

– Файл.

– Ключі.

– Шифрування.

– Дешифрування.

– Параметри.

– Довідка.

Крім того існують дві закладки:

– Шифрування.

– Дешифрування.

Вид закладки «Шифрування» наведено на рисунку 5.1.

З нього ми бачимо, що у цій закладці розташовані наступні дані:

– Файл для шифрування.

– Огляд дерева каталогів, для знаходження файлу для шифрування.

– Вікно завдання пароля.

– Шлях куди шифрувати.

– Вектор (IV).

Кнопка «Шифрувати», по натисканню на яку відбувається шифрування обраного файлу.

Вид закладки «Дешифрування» наведено на рисунку 5.2.

З нього ми бачимо, що у цій закладці розташовані наступні дані:

– Файл для дешифрування.

– Огляд дерева каталогів, для знаходження файлу для дешифрування.

– Вікно завдання пароля.

– Вектор (IV).

Кнопка «Дешифрувати», по натисканню на яку відбувається дешифрування обраного файлу.

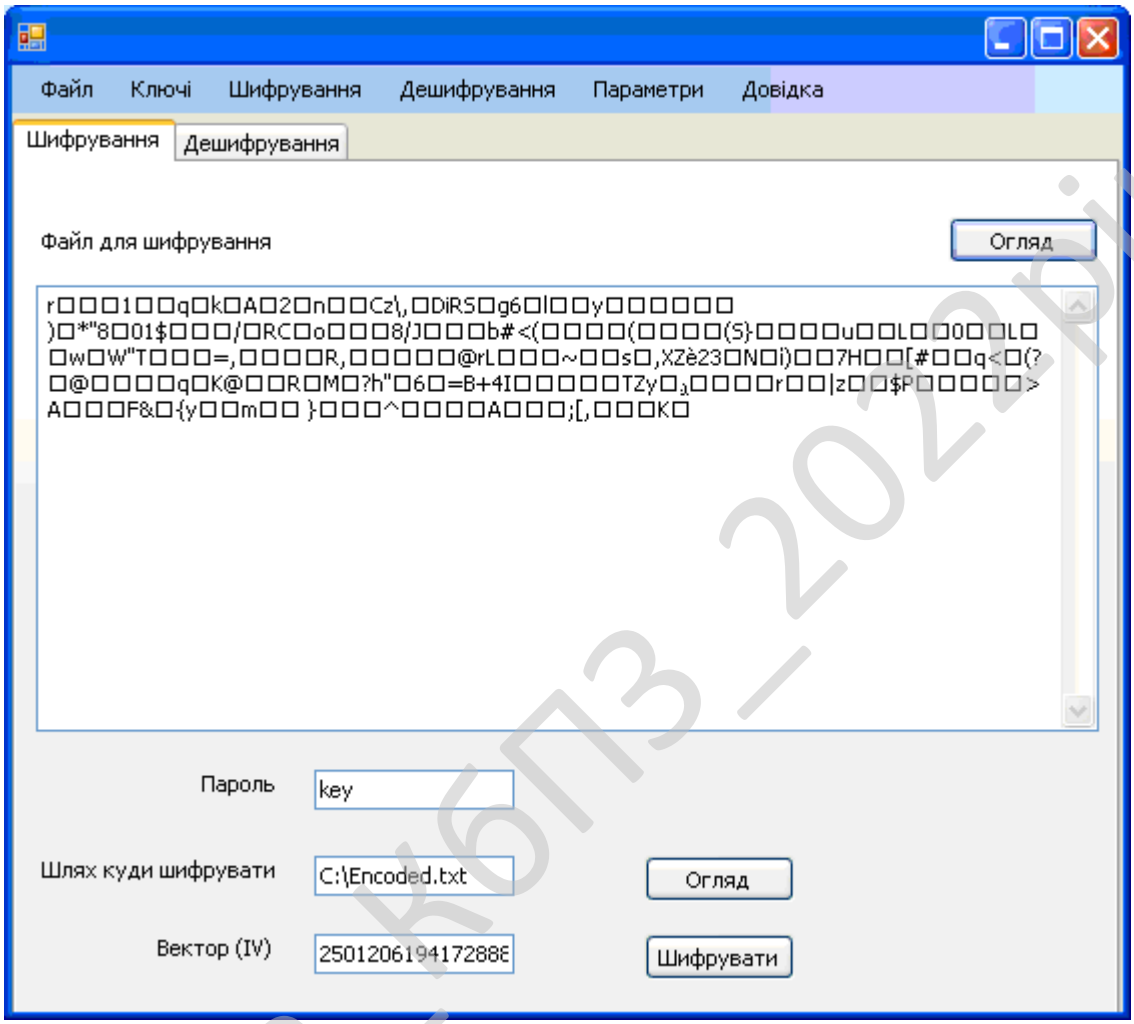

Рисунок 5.1 – Головне вікно програми (шифрування)

На рисунку 5.3 зображено вікно довідки на якому наведені наступні дані:

– Назву програми.

– Виконавець.

– Керівник.

– Місце розробки.

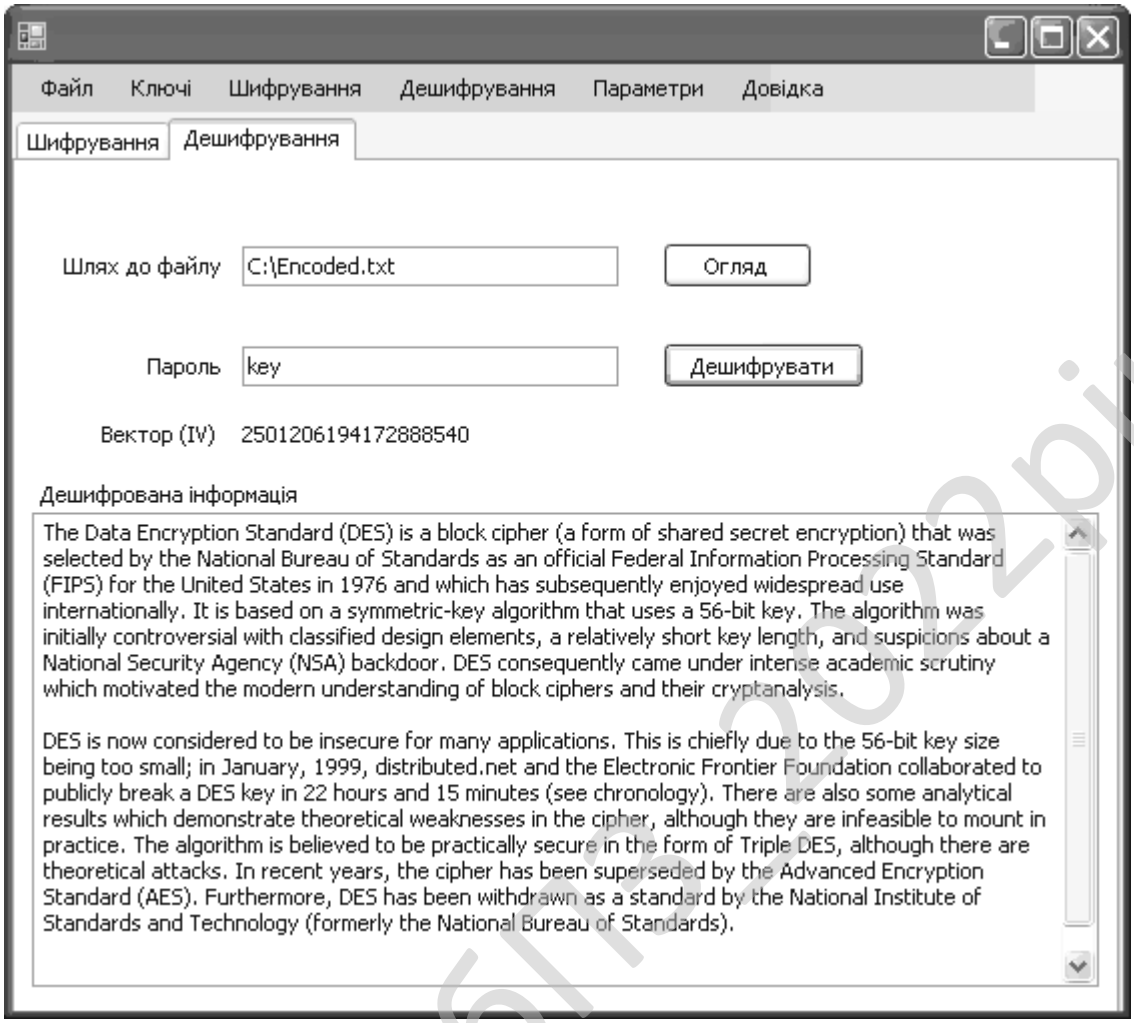

Рисунок 5.2 – Головне вікно програми (дешифрування)

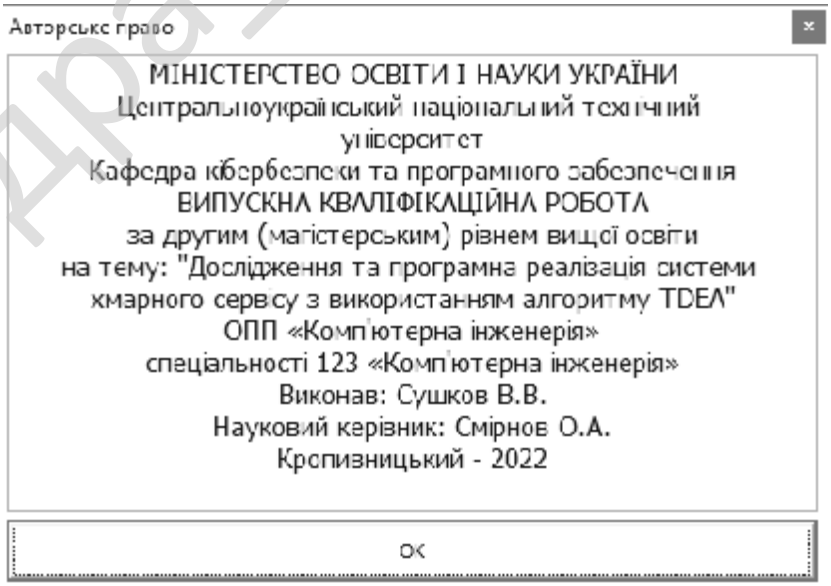

Рисунок 5.3 – Довідка

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи хмарного сервісу з використанням алгоритму TDEA.

Метою розробки є дослідження та програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA.

Об'єктом дослідження є процес хмарного сервісу з використанням алгоритму TDEA.

Предметом дослідження є методи хмарного сервісу з використанням алгоритму TDEA.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод хмарного сервісу з використанням алгоритму TDEA.

– Розроблено вітчизняний продукт хмарного сервісу з використанням алгоритму TDEA, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

ісля ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп’ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	120
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	120000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$
$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 50 = 84 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	84	Ф 7.1-7.4
Впровадження	13	Д13
Всього	125	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \qquad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{125 \cdot 1}{60 - 5} = 2,3 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	6	540	9
Монітор	60	6	360	6
Клавіатура	30	6	180	3
Маніпулятор «мишка»	30	6	180	3
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	1	30	0,5
Кабельні господарства ЛОМ на 1 м.п.	2,5	260	650	10,83
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	37,99

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1{,}2}, \tag{7.6}$$

$$Ф_{др}^{с} = \frac{38 \cdot 3}{1{,}2} = 95 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \tag{7.7}$$

$$Ч_{ел} = 95/(60 \cdot 8) = 1{,}2 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	0,5	18100	27150
Продакт-менеджер	0,25	10000	7500
Інженер-програміст	2,3	16000	110400
Інженер-електронщик	0,2	10000	6000
Інженер-системотехнік	0,25	10000	7500
Адміністратор мережі	0,5	10000	15000
Системний програміст	0,25	10000	7500
Дизайнер WEB	0,25	10000	7500
Інженер-верстальник	0,25	10000	7500
Бухгалтер-економіст	0,5	10000	15000
Всього за період розробки	$R_{сп}$ = 5,25	-	$Ф_{роб}$ = 211050

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{211050}{5,25 \cdot 60} = 670 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Brain за 26.10.22 – джерело http: //brain.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core i3 540 (3.067Ghz 4Mb_cache Clarkdale, 73W, socket1156) box	-
Системна плата	MB MSI H55M-E33 s1156 mATX (H55M E33)	-
Відеокарта	VC VTX Radeon HD6570 1GB GDDR3 128bit, 650 MHz/1334 MHz, PCI-E 2.1, DVI HDMI, VGA (VX6570 1GBK3-H)	-
Жорсткий диск	SSD: 240 Gb	-
Оперативна пам’ять	4Gb DDR3 PC3-12800 Patriot, 1600Mhz CL9, (9-9-9-28), 1.5V, Retail (PSD32G16002H) 2 модуля	-
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slim Black Bulk (DVR-TD10RS)	-
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA 489, PSU 350W(FSP Brand: ATX-350PNR 12cm), black, (front bezel – black+light silver body material – 0.6mm), 80mm fan (rear) 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-less chassis design,Thermally Advantaged Chassis	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 3.0 Card reader. 3.5", 2*USB3. +AUDIO+1394, multi: All Type Cards, black	-
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USE cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	0	20	0,0
6. Господарський інвентар	28000	25	7000
Всього по групі	33190	-	8297,5
7. Нематеріальні активи	120000	10	12000
Разом	$К_р$ = 1760367		$А_р$ = 190286

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 670 \cdot 125/120 = 698 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 698 \cdot 10 \cdot 0{,}01 = 70 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(698+70) = 288 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 698 \cdot 15 \cdot 0{,}01 = 105 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 210 \cdot 0{,}5 = 105 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 60):

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 24 грн./шт., DVD-R box – 35 грн./шт.

$$З_{М2} = 60 \cdot 24 = 1440 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (105 + 1440 + 1702)/120 = 27 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 698 \cdot 15 \cdot 0{,}01 = 105 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 120 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 190286 \cdot 3/(120 \cdot 12) = 396 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + C_{оц} + Г_{осп} + З_м + О_п + A_м. \quad (7.21)$$

$$C_п = 698+70+288+105+27+105+396 = 1689 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1. Основна зарплата виконавців	$З_о$	698
2. Додаткова зарплата виконавців	$З_д$	70
3. Відрахувння на соціальні потреби	$C_{оц}$	288
4. Загальногосподарські витрати	$Г_{осп}$	105
5. Витрати на матеріали	$З_М$	27
6. Освоєння нових операційних систем, мов програмування	$О_п$	105
7. Амортизація основних фондів	$A_м$	396
8. Повна собівартість програмного забезпечення	$C_п$	1689
9. Плановий прибуток	$П_р$	845
10.Ціна підприємства $Ц_п = C_п + П_р$	$Ц_п$	2534
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	506,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3040,8

Визначимо плановий прибуток за рівнем рентабельності (P_n) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_n \cdot C_n, \quad (7.22)$$

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 1689 = 845 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.9.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	3041
Всього капітальних витрат	–	3041

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	$З_р$	37576	21472
2. Витрати на електроенергію	$З_{ел}$	439	251
3. Витрати на амортизацію	$З_{ам}$	0	760
Всього витрат за рік	$І$	38015	22483

Витрати на обслуговування роботи системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість годин на обслуговування системи зменшилось з 350 год до 200 год на рік.

$$З_{р\,баз} = 350 \cdot 80 \cdot 1{,}1 \cdot 1{,}22 = 37576 \text{ грн},$$

$$З_{р\,нов} = 200 \cdot 80 \cdot 1{,}1 \cdot 1{,}22 = 21472 \text{ грн}.$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	3041	–	760,25
Всього відрахувань	-	–	3041	–	760,25

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($T_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot T_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = 0{,}545 \cdot 350 \cdot 2{,}3 = 439 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}545 \cdot 200 \cdot 2{,}3 = 251 \text{ грн.}$$

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (2534 - 1689) \cdot 120 - (0{,}05 \cdot 1408000 + 0{,}4 \cdot 199177 +$$
$$+0{,}25 \cdot 33190 + 0{,}1 \cdot 120000) \cdot 3/12 = 53828{,}5 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р$ – балансова вартість основних фондів розробника.

$$T_в = \frac{1760367}{(2534 - 1689) \cdot 120 \cdot 12 / 3} = 4 \text{ роки}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н (K_н - K_б), \quad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно; $К_б$, $К_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (38015\text{-}22483)\text{-}0{,}25 \cdot 3041 = 14772 \text{ грн.}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	120
2. Повна собівартість розробленої програми	Грн.	1689
3. Ціна розробленої програми	Грн.	2534
4. Плановий прибуток від реалізації розробленої програми	Грн.	845
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1760367
7. Загальний прибуток від реалізації програмної продукції	Грн.	101400
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	53828,5
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Рік	4
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3041
11.Величина економічного ефекту у користувача програмної продукції	Грн.	14772
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Рік	0,2

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{3041}{38015 - 22483} = 0,2 \ року .$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

8.2 Пожежна безпека

Пожежі в приміщеннях з оргтехнікою становлять особливу небезпеку, бо поєднані з великими матеріальними збитками. Пожежа може виникнути при взаємодії горючих речовин і джерел запалювання. Горючими речовинами є будівельні та опоряджувальні матеріали, пластмасові корпуси техніки, шнури тощо. Джерелами запалювання можуть бути електронні схеми комп'ютерів, принтерів, пристроїв електроживлення, де внаслідок різних порушень виникає перегрівання елементів, утворюються електричні іскри та дуги, здатні спричинити займання горючих матеріалів.

З метою виявлення початкової стадії займання необхідно використовувати пристрої систем автоматичного пожежогасіння там, де цього вимагають правила пожежної безпеки.

При обслуговуванні, ремонтних та профілактичних роботах використовуються різні легкозаймисті рідини, прокладаються тимчасові електропровідники, здійснюється паяння. Виникає додаткова пожежна небезпека, яка потребує відповідних заходів пожежного захисту. До засобів гасіння пожежі, призначених для локалізації невеликих займань, належать вогнегасники, сухий пісок, азбестові ковдри. Приміщення, в який встановлено комп'ютери і де немає необхідності влаштування систем автоматичного пожежогасіння, необхідно

оснащувати переносними вуглекислотними з розрахунку 2 шт. на кожні 20 м^2 в приміщеннях. Звуковбирне облицювання стін, стель приміщень треба виконувати з негорючих та важко горючих матеріалів.

Електроустановки (можливість їх застосування, монтаж, накладка експлуатація) повинні відповідати вимогам чинних правил улаштування електроустановок, правил технічної експлуатації, електроустановок та інших нормативних документів.

Ймовірність виникнення пожежі від електротехнічного та іншого одиничного виробу не повинна перевищувати 10^{-6} на рік. При короткому замиканні в місцях з'єднання проводів опір практично дорівнює нулю, звідси величина струму досягає дуже великих значень.

Персональні комп'ютери після закінчення роботи повинні відключатися від мережі не рідше 1 разу на квартал, необхідно очищати від пилу агрегати та вузли, кабельні канали та простір між підлогами. Не дозволяється розміщувати комп'ютерні зали ЕОМ у підвалах; проводити ремонт вузлів (блоків) ЕОМ безпосередньо у залах, де знаходяться ПК (персональні комп'ютери), залишати без нагляду ввімкнену в мережу електронну апаратуру, яка використовується для контролю ЕОМ.

Електричний струм силою 0,1 А є небезпечним для людини. Для попередження травм усе електричне обладнання повинне бути заземлене. Приступаючи до роботи необхідно перевірити справність обладнання, ізоляцію проводів і надійність заземлення. Доторкання до оголених струмоведучих і незахищених частин в електроустаткуванні забороняється. В разі виявлення порушень ізоляції електропроводів, відкритих струмоведучих частин електроустаткування або порушення заземлення треба негайно повідомити про це свого начальника для вжиття заходів щодо усунення несправності. Проводити самому ремонт електроустаткування забороняється.

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	5,4
Довжина	6
Висота	2,75

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого·

Геометрична характеристика	Одиниця виміру	Нормативне значення *	Фактичне значення
Площа, S	$м^2$	не менше 6.0	8,1
Обсяг, V	$м^3$	не менше 20.0	24,3

*Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин) [2].

У зазначеному приміщенні працюють 4 людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одно робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-

обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура,°С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	40-55	0,1
Тепла	23-25	50-70	0,1	24-25	50-65	0,11

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP 1100, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми України “Природне і штучне освітлення” – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4. Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Для захисного штучного заземлення застосовуються вертикальні електроди: металевий куток 50·50·5 мм., довжиною *L=3 м.,* та горизонтальний електрод – металева полоса з перетином 40·4 мм. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – у ряд.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: *H=0,4 м*. Відстань між вертикальними заземлювачами (електродами) *A*=3 *м*. Глибина закладення горизонтального контура заземлення *t=0,8 м*. Опір заземлювача, який нормується: $R_{3H} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок захисного заземлення можна автоматизувати за допомогою програми, сирцевий код якої опублікован на стр. 13-16 [6], або аналогічної.

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,8+3/2=2,3 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi \rho_2 = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [6];

$\rho_1 = 50$ *Ом·м.* – табличне значення питомого опору верхнього шару грунта [11];

$\rho_2 = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [11].

Зелёное выкинуть, заменить на:

$\rho = 40$ *Ом·м.* – табличне значення питомого опору нижнього шару грунта (глина) [11].

Еквівалентний діаметр ветикальньного електрода (кутка) [11]:

$$Dв=0,95 \cdot K= 0,95 \cdot 50 = 47,5 \text{ мм.} = 0,0475 \text{м.}$$

де $K = 50$ *мм.* – розмір металевого кутка (задан).

Опір розтіканню електричного струму одного електрода вертикального заземлювача (без врахуванням заглиблення заземлювача) [11]:

$$R_O = 0,366(\rho/L)\lg(4L/Dв) = 0,366(54,5/3)\lg(4 \cdot 3/0,0475) = 15,9 \text{ Ом.}$$

Відношення $A/L = 3/3 = 1$.

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [11]:

$$R_O = 0,366(\rho/L)[\lg(2L/Dв) + (1/2)\lg((4t+L)/(4t-L))] =$$

$$= 0,366(54,5/3)[(\lg(2 \cdot 3/0,0475) + (1/2)\lg((4 \cdot 2,3+3)/(4 \cdot 2,3-3))] =$$

$$= 14,9 \text{ Ом.}$$

Визначаємо коефіцієнт екранування вертикальних електродів $Кев = 0,8$ при орієнтовній кількості вертикальних електродів, яке дорівнює 4 [11].

Визначаємо необхідну кількість вертикальних заземлювачів (без врахування горизонтального заземлювача), при $R_{ЗН} = 4$ *Ом* :

$$N = R_O / (Кев \, R_{ЗН}) = 14,9 / (0,8 \cdot 4) = 4,66 \approx 5 \text{ шт.}$$

Визначаємо довжину з'єднуючої полоси:

$$L_П = 1,05 \cdot A \cdot N = 1,05 \cdot 3 \cdot 4,66 = 14,6 \approx 15 \text{ м.}$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [11]:

$$R_П = 0,366(\rho_2 \cdot K_П/L_П)\lg(2(L_П \cdot L_П)/(B \cdot t)) =$$

$$= 0,366(40 \cdot 5/14,6) \cdot \lg((2 \cdot 14,6^2)/(0,04 \cdot 0,8)) = 20,5 \text{ Ом.}$$

де $K_П = 5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної кліматичної зони для з'єднуючої полоси [11]: $B = 40$ *мм.* $= 0,04$ *м.*- ширина з'єднуючої полоси (задана).

Загальний опір розтіканню електричного струму заземлювача [11]:

$$R = (R_O \cdot R_П)/(R_O \cdot \eta_П + N \cdot R_П \cdot Кев) =$$

$$= (14,9 \cdot 20,5)/(14,9 \cdot 0,75 + 4,66 \cdot 20,5 \cdot 0,8) = 3,5 \text{ Ом}.$$

де $\eta_П = 0,75$ – табличне значення коефіцієнта екранування з'єднуючої полоси [11].

Умова $R \leq R_{ЗН}$ виконується ($3,5 \leq 4$).

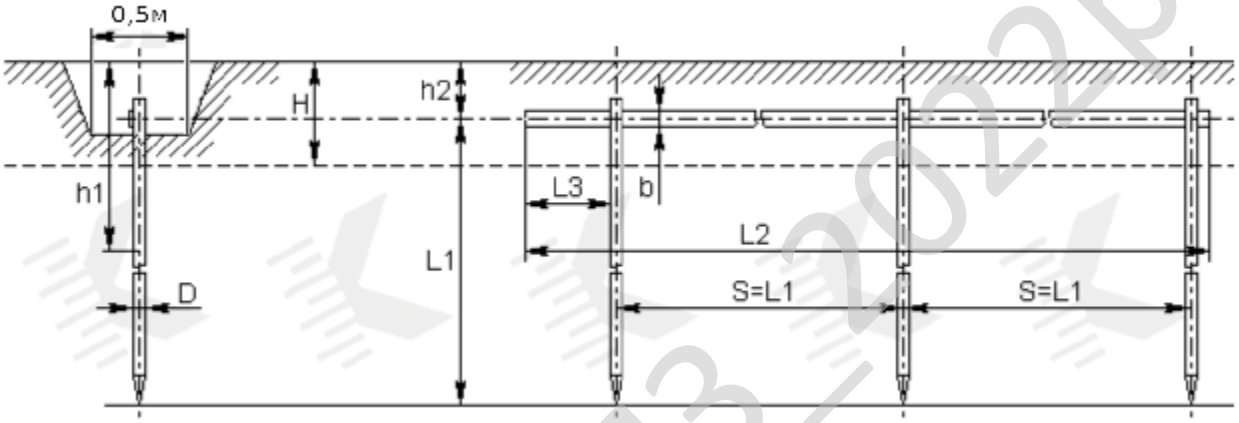

Рисунок 8.1 – Схема штучного заземлення

8.6 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи хмарного сервісу з використанням алгоритму TDEA.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів хмарного сервісу з використанням алгоритму TDEA.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем хмарного сервісу з використанням алгоритму TDEA.

– Досліджена система хмарного сервісу з використанням алгоритму TDEA.

– На основі отриманих результатів досліджень створена програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання хмарного сервісу з використанням алгоритму TDEA.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Visual C#. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ММВ.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 14772 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,2 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сушков В.В. Дослідження та програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022, pp. 1-12. **(Scopus).**

3. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». *Sensors (Basel, Switzerland)* Volume 22, Issue 16, 6223, 2022. (**Scopus**).

4. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. **Springer**, Singapore. pp. 21-34. **(Scopus).**

5. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. **Springer**, Cham. 2022, pp. 2463-2477. **(Scopus).**

6. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». *SN Computer Science,* Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w **(Scopus).**

7. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». *2021 IEEE 8th*

International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 **(Scopus).**

8. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». *CEUR Workshop Proceedings* Volume 3101, 2021, Pages 192-207. **(Scopus).**

9. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.* **(Scopus)**.

10. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256. **(Scopus)**.

11. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings.* Volume 2740, 2020, Pages 102-114. **(Scopus)**.

12. Smirnov O.A., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346. **(Scopus)**.

13. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131. **(Scopus)**.

14. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14. **(Scopus)**.

15. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions».

Lecture Notes in Networks and Systems, vol 152. **Springer**, Cham. 2021, pp 66-84. **(Scopus)**.

16. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T.**,** «Representation of Cascade Codes in the Frequency Domain»**.** In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. **Springer**, Cham. 2021. pp 557-587. **(Scopus)**.

17. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136. **(Scopus**).

18. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379. **(Scopus)**.

19. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43. **(Scopus)**.

20. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645. **(Scopus)**.

21. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660., **(Scopus)**.

22. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407. **(Scopus)**.

23. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during

Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019. **(Scopus)**.

24. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019. **(Scopus)**.

25. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629. **(Scopus)**.

26. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884. **(Scopus)**.

27. Smirnov, O., Kuznetsov, A., Prokopovych-Tkachenko, D. «Hiding Data in Images Using a Pseudo-Random Sequence». *ISCI'2020: Information Security in Critical Infrastructures.* ***Collective monograph***. Edited by Ivan D. Gorbenko, Victor A. Krasnobayev and Alexandr A. Kuznetsov. ASC Academic Publishing, USA, 2020. pp. 46-59. – ISBN: 978-1-7362833-0-1 (Hardback), ISBN: 978-1-7362833-1-8 (Ebook).

28. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures**. Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

29. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. **Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

30. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

31. Смірнов О.А., Дрєєва Г.М., «Метод генерування фрактального трафіку за допомогою моделі генератора на графі» у Інформаційна безпека та інформаційні технології: монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139.

32. Смирнов А.А., Коваленко А.В. Комплекс математических моделей технологии тестирования WEB-приложений. Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

33. Смирнов А.А., Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения. Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

34. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98. 2022.

35. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

36. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного

захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

37. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

38. Смирнов А., Кузнецов А., Кузнецова Т. «Шумоподобные дискретные сигналы для асинхронных систем кодового разделения радиоканалов». *Радиотехника*, № 2(205), 175–183. 2021.

39. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». *CEUR Workshop Proceedings* Volume 2732, 2020, Pages 214-227.

40. Смірнов, О.А., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю.Усік П.С., «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». *Проблеми телекомунікацій*. № 1(26). С. 83-96. 2020.

41. Смирнов А.А, Кузнецов А.А., Киян А.С., Кузнецова Е.А. «Сокрытие данных на основе адресации шумоподобных сигналов». *Всеукраїнський міжвідомчий науково-технічний збірник "Радіотехніка"* – Харьків: ХНУРЕ. – 2020. – Вип. 203. – С. 38-49.

42. Смирнов А.А., Дудан А.В., Смирнова Т.В. «Формализация структуры технологического процесса электродугового напыления». *Сборник научных трудов «Актуальные вопросы машиноведения».* Объединенный институт машиностроения Национальной Академии Наук Беларуси. №9. С. 308-312, 2020.

43. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

44. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

45. А.А. Смирнов, Т.В. Смирнова, А.Н. Дреев, А.В. Дудан. «Оптимизация технологического процесса восстановления и упрочнения поверхностей с заданными характеристиками в виде облачного сервиса». Вестник Полоцкого государственного университета. Серия В, Промышленность. Прикладные науки. Республика Беларусь – 2020. – № 3. – С. 50-61.

46. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

47. О.А. Смірнов, Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», *Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура* № 1 (11). с. 48-57, 2019.

48. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

49. Смірнов О.А., Смірнова Т.В., Солових Є.К., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

50. Смірнов О.А., Смірнова Т.В.,. Дрєєв О.М, «Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням», Системи управління, навігації та зв'язку, № 2 (54). с. 149-154, 2019.

51. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

52. Смирнов А.А., Лысенко И.А., Информационная технология проектирования тестовых наборов на основе требований к программному обеспечению, Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

53. Смірнов О.А., Мелешко Є.В., Хох В.Д., Дослідження методів аудиту систем управління інформаційною безпекою, Системи управління, навігації та зв'язку. – Випуск 1 (41). – Полтава: ПолтНТУ. – 2017. – С. 38-42.

54. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: https://goo.su/9AkQ

55. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

56. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

57. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

58. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

59. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград:

КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

60. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

61. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

62. Центр післядипломної освіти та підвищення кваліфікації. – Режим доступу до ресурсу: https://cpo.stu.cn.ua

63. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Економічні вимоги.....5
8 Вимоги щодо охорони праці.....5
9 Перелік документів, що розробляються.....6
10 Етапи розробки.....6
11 Порядок контролю та приймання.....6

					ВКРМ-123.22.0023.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Сушков В.В.			Дослідження та програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Смірнов О.А.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КІ-21М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи хмарного сервісу з використанням алгоритму TDEA.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 19-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи хмарного сервісу з використанням алгоритму TDEA.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи хмарного сервісу з використанням алгоритму TDEA;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Visual C#.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинна бути розглянута пожежна безпека.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 106 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 21.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Смірнов О.А.

Дослідження та програмна реалізація

системи хмарного сервісу з використанням алгоритму TDEA

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 19

Літера: РП

Кропивницький – 2022 року

Файл Program.cs - файл проекту

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace MyTripleDES
{
    static class Program
    {
        /// <summary>
        /// Головна точка входу до додатку.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Файл Form1.cs - головна програма

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Security.Cryptography;



namespace MyTripleDES
{
    public partial class Form1 : Form
    {
        protected byte[] IVector = null;

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Filter = "Txt files (*.txt)|*.txt|All files (*.*)|*.*";
            ofd.ShowDialog();
            if (ofd.FileName != "")
            {
                textBox1.Text = new
FileInfo(ofd.FileName).OpenText().ReadToEnd();
            }
        }


        private void button2_Click(object sender, EventArgs e)
        {
            SaveFileDialog sfd = new SaveFileDialog();
            sfd.InitialDirectory = "C:\\";
            sfd.Filter = "txt файли (*.txt)|*.txt|Усі файли (*.*)|*.*";
            sfd.ShowDialog();
            textBox3.Text = sfd.FileName;
        }

        private void button3_Click(object sender, EventArgs e)
        {
            try
            {
                // Створюємо новий TripleDESCryptoServiceProvider об’єкт
                // для генерування вектора ініціалізації (IV).
                TripleDESCryptoServiceProvider tDESalg = new
TripleDESCryptoServiceProvider();

                // Для наочності виводимо значення вектора ініціалізації
                string temp = null;
                for (int i = 0; i < tDESalg.IV.Length; i++)
                {
                    temp += tDESalg.IV[i].ToString();
                }
                textBox4.Text = temp;
                label9.Text = temp;
```

```
                // Запам'ятовуємо вектор у локальну змінну IVector
                IVector = tDESalg.IV;

                // Рядок для шифрування
                string sData = textBox1.Text;

                // Одержуємо ключ, перетворюємо в масив байтів і
                // доповнюємо до довжини 24. Не більше й не менше.
                string strKey = textBox2.Text;
                byte[] bKey = new byte[24];
                for (int i = 0, j = 0; i < 24; i++, j++)
                {
                    if (j == strKey.Length)
                        j = 0;
                    bKey[i] = (byte)strKey[j];
                }


                // Указуємо файл, куди будемо шифрувати
                string FileName = textBox3.Text;


                // Шифруємо текст у файл.
                // При цьому вказуємо ім'я файлу, ключ і вектор ініціалізації.
                CryptText.EncryptTextToFile(sData, FileName, bKey, tDESalg.IV);

                // Повідомляємо про результат
                MessageBox.Show("Текст успішно зашифрований !!!");
            }
            catch (Exception exc)
            {
                MessageBox.Show(exc.Message);
            }

        }


        private void button4_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Filter = "Txt files (*.txt)|*.txt|All files (*.*)|*.*";
            ofd.ShowDialog();
            if (ofd.FileName != "")
            {
                textBox5.Text = new FileInfo(ofd.FileName).FullName;
            }
        }



        private void button5_Click(object sender, EventArgs e)
        {
            try
            {
                // Створюємо новий TripleDESCryptoServiceProvider об’єкт
                // для генерування вектора ініціалізації (IV).
                TripleDESCryptoServiceProvider tDESalg = new
TripleDESCryptoServiceProvider();

                // Одержуємо ключ, перетворюємо в масив байтів і
                // доповнюємо до довжини 24. Не більше й не менше.
                string strKey = textBox6.Text;
                byte[] bKey = new byte[24];
                for (int i = 0, j = 0; i < 24; i++, j++)
                {
                    if (j == strKey.Length)
                        j = 0;
                    bKey[i] = (byte)strKey[j];
                }
```

```
                // Указуємо файл для дешифрування
                string FileName = textBox5.Text;


                // Дешифруємо текст із файлу.
                // При цьому вказуємо ім'я файлу, ключ і вектор ініціалізації
IVector.
                textBox7.Clear();
                textBox7.Text += CryptText.DecryptTextFromFile(FileName, bKey,
IVector);

                // Повідомляємо про результат
                MessageBox.Show("Текст успішно дешифрований !!!");
            }
            catch (Exception exc)
            {
                MessageBox.Show(exc.Message);
            }

        }



        private void Form1_Load(object sender, EventArgs e)
        {
            textBox3.Text = @"C:\Encoded.txt";
            textBox2.Text = "key";
            textBox6.Text = "key";
        }


    }
}
```

Файл Form1.Designer.cs - інтерфейс користувача

```
namespace MyTripleDES
{
    partial class Form1
    {
        /// <summary>
        /// Необхідні змінні розробника.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Очищуємо усі ресурси необхідні для використання.
        /// </summary>
        /// <param name="disposing">true якщо ресурси управління повинні бути
розташовані, інакше, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }


        #region Windows Form Designer generated code

        /// <summary>
        /// Необхідний метод для розробника - не модифікує
        /// вміст цього методу, з редактором коду.
        /// </summary>
        private void InitializeComponent()
        {
            this.tabControl1 = new System.Windows.Forms.TabControl();
            this.tabPage1 = new System.Windows.Forms.TabPage();
            this.label4 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.textBox4 = new System.Windows.Forms.TextBox();
            this.textBox3 = new System.Windows.Forms.TextBox();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.button3 = new System.Windows.Forms.Button();
            this.button2 = new System.Windows.Forms.Button();
            this.button1 = new System.Windows.Forms.Button();
            this.tabPage2 = new System.Windows.Forms.TabPage();
            this.label9 = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.textBox7 = new System.Windows.Forms.TextBox();
            this.textBox6 = new System.Windows.Forms.TextBox();
            this.textBox5 = new System.Windows.Forms.TextBox();
            this.button5 = new System.Windows.Forms.Button();
            this.button4 = new System.Windows.Forms.Button();
            this.tabControl1.SuspendLayout();
            this.tabPage1.SuspendLayout();
            this.tabPage2.SuspendLayout();
            this.SuspendLayout();
            //
            // tabControl1
            //
            this.tabControl1.Controls.Add(this.tabPage1);
            this.tabControl1.Controls.Add(this.tabPage2);
```

```
this.tabControl1.Dock = System.Windows.Forms.DockStyle.Fill;
this.tabControl1.Location = new System.Drawing.Point(0, 0);
this.tabControl1.Name = "tabControl1";
this.tabControl1.SelectedIndex = 0;
this.tabControl1.Size = new System.Drawing.Size(451, 456);
this.tabControl1.TabIndex = 0;
//
// tabPage1
//
this.tabPage1.Controls.Add(this.label4);
this.tabPage1.Controls.Add(this.label3);
this.tabPage1.Controls.Add(this.label2);
this.tabPage1.Controls.Add(this.label1);
this.tabPage1.Controls.Add(this.textBox4);
this.tabPage1.Controls.Add(this.textBox3);
this.tabPage1.Controls.Add(this.textBox2);
this.tabPage1.Controls.Add(this.textBox1);
this.tabPage1.Controls.Add(this.button3);
this.tabPage1.Controls.Add(this.button2);
this.tabPage1.Controls.Add(this.button1);
this.tabPage1.Location = new System.Drawing.Point(4, 22);
this.tabPage1.Name = "tabPage1";
this.tabPage1.Padding = new System.Windows.Forms.Padding(3);
this.tabPage1.Size = new System.Drawing.Size(443, 430);
this.tabPage1.TabIndex = 0;
this.tabPage1.Text = "Шифрування";
this.tabPage1.UseVisualStyleBackColor = true;
//
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(59, 392);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(62, 13);
this.label4.TabIndex = 10;
this.label4.Text = "Вектор (IV)";
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(14, 349);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(107, 13);
this.label3.TabIndex = 9;
this.label3.Text = "Шлях куди шифруємо";
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(76, 303);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(45, 13);
this.label2.TabIndex = 8;
this.label2.Text = "Пароль";
//
// label1
//
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(47, 37);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(180, 13);
this.label1.TabIndex = 7;
this.label1.Text = "Текстовий файл для шифрування";
//
// textBox4
//
this.textBox4.Location = new System.Drawing.Point(127, 389);
this.textBox4.Name = "textBox4";
this.textBox4.Size = new System.Drawing.Size(100, 20);
```

```
this.textBox4.TabIndex = 6;
//
// textBox3
//
this.textBox3.Location = new System.Drawing.Point(127, 346);
this.textBox3.Name = "textBox3";
this.textBox3.Size = new System.Drawing.Size(100, 20);
this.textBox3.TabIndex = 5;
//
// textBox2
//
this.textBox2.Location = new System.Drawing.Point(127, 300);
this.textBox2.Name = "textBox2";
this.textBox2.Size = new System.Drawing.Size(100, 20);
this.textBox2.TabIndex = 4;
//
// textBox1
//
this.textBox1.Location = new System.Drawing.Point(8, 61);
this.textBox1.Multiline = true;
this.textBox1.Name = "textBox1";
this.textBox1.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
this.textBox1.Size = new System.Drawing.Size(427, 223);
this.textBox1.TabIndex = 3;
//
// button3
//
this.button3.Location = new System.Drawing.Point(312, 387);
this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(75, 23);
this.button3.TabIndex = 2;
this.button3.Text = "Шифрувати";
this.button3.UseVisualStyleBackColor = true;
this.button3.Click += new System.EventHandler(this.button3_Click);
//
// button2
//
this.button2.Location = new System.Drawing.Point(312, 344);
this.button2.Name = "button2";
this.button2.Size = new System.Drawing.Size(75, 23);
this.button2.TabIndex = 1;
this.button2.Text = "Огляд";
this.button2.UseVisualStyleBackColor = true;
this.button2.Click += new System.EventHandler(this.button2_Click);
//
// button1
//
this.button1.Location = new System.Drawing.Point(312, 32);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(75, 23);
this.button1.TabIndex = 0;
this.button1.Text = "Огляд";
this.button1.UseVisualStyleBackColor = true;
this.button1.Click += new System.EventHandler(this.button1_Click);
//
// tabPage2
//
this.tabPage2.Controls.Add(this.label9);
this.tabPage2.Controls.Add(this.label8);
this.tabPage2.Controls.Add(this.label7);
this.tabPage2.Controls.Add(this.label6);
this.tabPage2.Controls.Add(this.label5);
this.tabPage2.Controls.Add(this.textBox7);
this.tabPage2.Controls.Add(this.textBox6);
this.tabPage2.Controls.Add(this.textBox5);
this.tabPage2.Controls.Add(this.button5);
this.tabPage2.Controls.Add(this.button4);
this.tabPage2.Location = new System.Drawing.Point(4, 22);
this.tabPage2.Name = "tabPage2";
```

```
this.tabPage2.Padding = new System.Windows.Forms.Padding(3);
this.tabPage2.Size = new System.Drawing.Size(443, 430);
this.tabPage2.TabIndex = 1;
this.tabPage2.Text = "Дешифрування";
this.tabPage2.UseVisualStyleBackColor = true;
//
// label9
//
this.label9.AutoSize = true;
this.label9.Location = new System.Drawing.Point(96, 129);
this.label9.Name = "label9";
this.label9.Size = new System.Drawing.Size(35, 13);
this.label9.TabIndex = 9;
this.label9.Text = "label9";
//
// label8
//
this.label8.AutoSize = true;
this.label8.Location = new System.Drawing.Point(19, 159);
this.label8.Name = "label8";
this.label8.Size = new System.Drawing.Size(112, 13);
this.label8.TabIndex = 8;
this.label8.Text = "Дешифрована інформація.";
//
// label7
//
this.label7.AutoSize = true;
this.label7.Location = new System.Drawing.Point(19, 129);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(62, 13);
this.label7.TabIndex = 7;
this.label7.Text = "Вектор (IV)";
//
// label6
//
this.label6.AutoSize = true;
this.label6.Location = new System.Drawing.Point(48, 96);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(45, 13);
this.label6.TabIndex = 6;
this.label6.Text = "Пароль";
//
// label5
//
this.label5.AutoSize = true;
this.label5.Location = new System.Drawing.Point(19, 45);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(74, 13);
this.label5.TabIndex = 5;
this.label5.Text = "Шлях до файлу";
//
// textBox7
//
this.textBox7.Location = new System.Drawing.Point(6, 175);
this.textBox7.Multiline = true;
this.textBox7.Name = "textBox7";
this.textBox7.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
this.textBox7.Size = new System.Drawing.Size(429, 249);
this.textBox7.TabIndex = 4;
//
// textBox6
//
this.textBox6.Location = new System.Drawing.Point(99, 93);
this.textBox6.Name = "textBox6";
this.textBox6.Size = new System.Drawing.Size(188, 20);
this.textBox6.TabIndex = 3;
//
// textBox5
//
```

```
        this.textBox5.Location = new System.Drawing.Point(99, 42);
        this.textBox5.Name = "textBox5";
        this.textBox5.Size = new System.Drawing.Size(188, 20);
        this.textBox5.TabIndex = 2;
        //
        // button5
        //
        this.button5.Location = new System.Drawing.Point(295, 91);
        this.button5.Name = "button5";
        this.button5.Size = new System.Drawing.Size(101, 23);
        this.button5.TabIndex = 1;
        this.button5.Text = "Дешифрувати";
        this.button5.UseVisualStyleBackColor = true;
        this.button5.Click += new System.EventHandler(this.button5_Click);
        //
        // button4
        //
        this.button4.Location = new System.Drawing.Point(295, 40);
        this.button4.Name = "button4";
        this.button4.Size = new System.Drawing.Size(75, 23);
        this.button4.TabIndex = 0;
        this.button4.Text = "Огляд";
        this.button4.UseVisualStyleBackColor = true;
        this.button4.Click += new System.EventHandler(this.button4_Click);
        //
        // Form1
        //
        this.AutoScaleDimensions = new System.Drawing.Size(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(451, 456);
        this.Controls.Add(this.tabControl1);
        this.Name = "Form1";
        this.Text = "Form1";
        this.Load += new System.EventHandler(this.Form1_Load);
        this.tabControl1.ResumeLayout(false);
        this.tabPage1.ResumeLayout(false);
        this.tabPage1.PerformLayout();
        this.tabPage2.ResumeLayout(false);
        this.tabPage2.PerformLayout();
        this.ResumeLayout(false);

    }

    #endregion

    private System.Windows.Forms.TabControl tabControl1;
    private System.Windows.Forms.TabPage tabPage1;
    private System.Windows.Forms.TabPage tabPage2;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.TextBox textBox4;
    private System.Windows.Forms.TextBox textBox3;
    private System.Windows.Forms.TextBox textBox2;
    private System.Windows.Forms.TextBox textBox1;
    private System.Windows.Forms.Button button3;
    private System.Windows.Forms.Button button2;
    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.Label label4;
    private System.Windows.Forms.Button button5;
    private System.Windows.Forms.Button button4;
    private System.Windows.Forms.Label label9;
    private System.Windows.Forms.Label label8;
    private System.Windows.Forms.Label label7;
    private System.Windows.Forms.Label label6;
    private System.Windows.Forms.Label label5;
    private System.Windows.Forms.TextBox textBox7;
    private System.Windows.Forms.TextBox textBox6;
    private System.Windows.Forms.TextBox textBox5;
```

```
        }
    }
```

Файл CryptText.cs - шифрування/дешифрування

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Security.Cryptography;
using System.IO;
using System.Windows.Forms;

namespace MyTripleDES
{
    class CryptText
    {
        // Конструктор
        public CryptText() { }

        /// <summary>
        /// Метод шифрує текст у файл
        /// </summary>
        /// <param name="Data">Текст для шифрування</param>
        /// <param name="FileName">Повний шлях до файлу</param>
        /// <param name="Key">Ключ (пароль)</param>
        /// <param name="IV">Вектор ініціалізації</param>
        public static void EncryptTextToFile(String Data, String FileName,
byte[] Key, byte[] IV)
        {
            try
            {
                // Створює або відкриває визначений файл.
                FileStream fStream = File.Open(FileName, FileMode.OpenOrCreate);

                // Створює CryptoStream який використовує FileStream
                // та ключ шифрування й ініціалізує вектор(IV).
                CryptoStream cStream = new CryptoStream(fStream,
                    new TripleDESCryptoServiceProvider().CreateEncryptor(Key,
IV),
                    CryptoStreamMode.Write);

                // Створює StreamWriter який використовує CryptoStream.
                StreamWriter sWriter = new StreamWriter(cStream);

                // Записує дані до потоку
                // для їх шифрування.
                sWriter.WriteLine(Data);

                // Закриваємо поток
                // закриваємо файл.
                sWriter.Close();
                cStream.Close();
                fStream.Close();
            }
            catch (CryptographicException e)
            {
                MessageBox.Show("Відбулася криптографічна помилка: {0}",
e.Message);
            }
            catch (UnauthorizedAccessException e)
            {
                MessageBox.Show("Відбулася помилка доступу до файлу: {0}",
e.Message);
            }

        }

        /// <summary>
        /// Метод дешифрує текст із файлу
```

```
        /// </summary>
        /// <param name="FileName">Повний шлях до файлу</param>
        /// <param name="Key">Ключ (пароль)</param>
        /// <param name="IV">Вектор ініціалізації</param>
        /// <returns>Текст у дешифрованому виді</returns>
        public static string DecryptTextFromFile(String FileName, byte[] Key,
byte[] IV)
        {
            try
            {
                // Створює або відкриває визначений файл.
                FileStream fStream = File.Open(FileName, FileMode.OpenOrCreate);

                // Створює CryptoStream який використовує FileStream
                // та ключ шифрування й ініціалізує вектор(IV).
                CryptoStream cStream = new CryptoStream(fStream,
                    new TripleDESCryptoServiceProvider().CreateDecryptor(Key,
IV),
                    CryptoStreamMode.Read);

                // Створює StreamReader який використовує CryptoStream.
                StreamReader sReader = new StreamReader(cStream);

                // Читаємо файли з потоку
                // для їх дешифрування.
                string val = sReader.ReadToEnd();

                // Закриваємо поток
                // закриваємо файл.
                sReader.Close();
                cStream.Close();
                fStream.Close();

                // Повертаємо рядок.
                return val;
            }
            catch (CryptographicException e)
            {
                MessageBox.Show("Відбулася криптографічна помилка: {0}",
e.Message);
                return null;
            }
            catch (UnauthorizedAccessException e)
            {
                MessageBox.Show("Відбулася помилка доступу до файлу: {0}",
e.Message);
                return null;
            }
        }
    }
}
```

Файл AssemblyInfo.cs - інформація про версію програми

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// Голона інформація про параметри
[assembly: AssemblyTitle("MyTripleDES")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("MyTripleDES")]
[assembly: AssemblyCopyright("Copyright ©  2010")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Установка  COM компонентів.
[assembly: ComVisible(false)]

// Записуємо GUID для ID проекту у бібліотеці COM
[assembly: Guid("95c881e 5-c815-498b-924 a-5fdf3ce3d355")]

// Інформація про версію
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Файл Resources.Designer.cs - робота з інтерфейсом

```
// - - - - - - - - - - - - - ----------------------
// < auto-generated>
//
// </ auto-generated>
// - - - - - - - - - - - - - ----------------------

namespace MyTripleDES.Properties
{


    /// <summary>
    ///
    /// </summary>
    // Цей class автогенерує StronglyTypedResourceBuilder
    //.

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.
StronglyTypedResourceBuilder", "2.0.0.0")]
    [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
    internal class Resources
    {

        private static global::System.Resources.ResourceManager resourceMan;

        private static global::System.Globalization.CultureInfo resourceCulture;


[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Per
formance", "CA1811:AvoidUncalledPrivateCode")]
        internal Resources()
        {
        }

        /// <summary>
        ///   повертає кеш  ResourceManager використаний цим класом.
        /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.Component
Model.EditorBrowsableState.Advanced)]
        internal static global::System.Resources.ResourceManager ResourceManager
        {
            get
            {
                if ((resourceMan == null))
                {
                    global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("MyTripleDES.Properties.Resources",
typeof(Resources).Assembly);
                    resourceMan = temp;
                }
                return resourceMan;
            }
        }

        /// <summary>
        ///   Анулює CurrentUICulture властивості.
        /// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.Component
Model.EditorBrowsableState.Advanced)]
        internal static global::System.Globalization.CultureInfo Culture
        {
            get
            {
```

```
                return resourceCulture;
            }
            set
            {
                resourceCulture = value;
            }
        }
    }
}
```

Файл Settings.Designer.cs - параметри

```
//------------------------------------------------------------------------------
// <auto-generated>
//
// </auto-generated>
//------------------------------------------------------------------------------

namespace MyTripleDES.Properties
{


    [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.
Editors.SettingsDesigner.SettingsSingleFileGenerator", "9.0.0.0")]
    internal sealed partial class Settings :
global::System.Configuration.ApplicationSettingsBase
    {

        private static Settings defaultInstance =
((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(ne
w Settings())));

        public static Settings Default
        {
            get
            {
                return defaultInstance;
            }
        }
    }
}
```

Файл About.cs - вікно довідки про програму

```
namespace About
{
    partial class AboutForm
    {
        /// <summary>
        /// Необхідні змінні розробника.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Очищуємо усі ресурси необхідні для використання.
        /// </summary>
        /// <param name="disposing">true якщо ресурси управління повинні бути
розташовані, інакше, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Необхідний метод для розробника - не модифікує
        /// вміст цього методу, з редактором коду.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.developersListBoxdevelopersListBox = new
System.Windows.Forms.ListBox();
            this.RSIconTimer = new System.Windows.Forms.Timer(this.components);
            this.okButtonXP = new PinkieControls.ButtonXP();
            this.SuspendLayout();
            //
            // developersListBoxdevelopersListBox
            //
            this.developersListBoxdevelopersListBox.BackColor =
System.Drawing.SystemColors.Control;
            this.developersListBoxdevelopersListBox.BorderStyle =
System.Windows.Forms.BorderStyle.None;
            this.developersListBoxdevelopersListBox.FormattingEnabled = true;


           this.developersListBoxdevelopersListBox.Items.AddRange(new object[] {
            "МАГІСТЕРСЬКА РОБОТА",
            "",
            "На тему:",
            "",
            "Дослідження та програмна реалізація системи хмарного сервісу з
використанням алгоритму TDEA",
            "",
            "",
            "Керівник: Смірнов О.А.",
            "",
            "Розробив: студент Сушков Вадим Вадимович ",
            "                  гр. КІ-21М-1,4
",
            "",
            "м. Кропивницький 2022"});
            this.developersListBoxdevelopersListBox.Location = new
System.Drawing.Point(11, 6);
```

```
            this.developersListBoxdevelopersListBox.Name =
"developersListBoxdevelopersListBox";
            this.developersListBoxdevelopersListBox.SelectionMode =
System.Windows.Forms.SelectionMode.None;
            this.developersListBoxdevelopersListBox.Size = new
System.Drawing.Size(330, 182);
            this.developersListBoxdevelopersListBox.TabIndex = 0;
            this.developersListBoxdevelopersListBox.TabStop = false;
            this.developersListBoxdevelopersListBox.SelectedIndexChanged += new
System.EventHandler(this.developersListBoxdevelopersListBox_SelectedIndexChanged
);
            //
            // RSIconTimer
            //
            this.RSIconTimer.Interval = 40;
            this.RSIconTimer.Tick += new
System.EventHandler(this.RSIconTimer_Tick);
            //
            // okButtonXP
            //
            this.okButtonXP.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(236)))),
((int)(((byte)(233)))), ((int)(((byte)(216)))));
            this.okButtonXP.DefaultScheme = true;
            this.okButtonXP.DialogResult =
System.Windows.Forms.DialogResult.None;
            this.okButtonXP.Hint = "";
            this.okButtonXP.Location = new System.Drawing.Point(266, 177);
            this.okButtonXP.Name = "okButtonXP";
            this.okButtonXP.Scheme = PinkieControls.ButtonXP.Schemes.Blue;
            this.okButtonXP.Size = new System.Drawing.Size(75, 23);
            this.okButtonXP.TabIndex = 0;
            this.okButtonXP.Text = "OK";
            this.okButtonXP.Click += new
System.EventHandler(this.okButtonXP_Click);
            //
            // AboutForm
            //
            this.AutoScaleDimensions = new System.Drawing.Size(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(350, 212);
            this.Controls.Add(this.okButtonXP);
            this.Controls.Add(this.developersListBoxdevelopersListBox);
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedDialog;
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "AboutForm";
            this.ShowInTaskbar = false;
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterParent;
            this.Text = "Про програму...";
            this.Load += new System.EventHandler(this.AboutForm_Load);
            this.FormClosing += new
System.Windows.Forms.FormClosingEventHandler(this.AboutForm_FormClosing);
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.ListBox developersListBoxdevelopersListBox;
        private System.Windows.Forms.Timer RSIconTimer;
        private PinkieControls.ButtonXP okButtonXP;

    }
}
```