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
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу”

Виконав здобувач вищої освіти
IV курсу, групи КБ-20-3СК
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Ковальов Є.В.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Ковальову Євгену Валерійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 13-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

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

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Ковальов Є.В.
(прізвище та ініціали)

АНОТАЦІЯ

Ковальов Є.В. Програмне забезпечення системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу.

Метою розробки є програмне забезпечення системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу.

Результат роботи – програмна реалізація системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу.

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: кібербезпека, хмара, розмежуванням доступу

ABSTRACT

Kovalov E.V. Cyber security system software to protect information in the cloud with restricted access. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for a cyber security system to protect information in the cloud with access delimitation.

The goal of the development is a cyber security system software to protect information in the cloud with delimitation of access.

The result of the work is a software implementation of a cyber security system for protecting information in the cloud with restricted access.

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: cyber security, cloud, demarcation of access

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕН	–	електронний нотаріус
ЕЦП	–	електронний цифровий підпис
СУБД	–	система управління базами даних
УЦ	–	удостовірюючий центр
CRL	–	список відкликаних сертифікатів
DVCS	–	Data Validation and Certification Server Protocols
http	–	HyperText Transfer Protocol – протокол передачі гіпертексту
HTTPS	–	розширення протоколу HTTP, підтримуюче шифрування
IMAP	–	протокол доступу до електронної пошти Інтернету
MAC	–	код автентифікації повідомлення
OCSP	–	Online Certificate Status Protocol
PKI	–	інфраструктура відкритих ключів
POP3	–	Post Office Protocol Version 3 – протокол поштового відділення
SMTP	–	простий протокол передачі пошти
SSL	–	Secure Sockets Layer – рівень захищених сокетів
SQL	–	Structured Query Language – мова структурованих запитів
TCP	–	протокол управління передачею
TLS	–	криптографічний протокол, захищаючий передачу даних
TSP	–	Time-Stamp Protocol
URL	–	єдиний показник ресурсів
XMPP	–	росширяємий протокол обміну повідомленнями й інформацією про присутність

ВСТУП

Актуальність теми. Хмарний захист даних – це модель захисту даних, яка зосереджена на організаційних даних, які зберігаються, обробляються та керуються в хмарному або гібридному середовищі. Модель потребує багатьох політик щодо даних, стратегій і рішень для роботи в тандемі. Захист даних – це процес захисту даних організації від компрометації, втрати чи крадіжки. Хоча традиційні моделі захисту даних добре працюють для простих локальних розгортань, вони можуть бути складнішими, якщо дані зберігаються в хмарних або гібридних середовищах. Ця стаття присвячена саме захисту хмарних даних. Коли експерти говорять про захист даних, це охоплює:

– Захист даних: боротьба з втратою даних шляхом вивчення систем і процесів резервного копіювання та відновлення.

– Безпека даних: захист даних компанії та клієнтів від внутрішніх і зовнішніх загроз.

– Конфіденційність даних: Контроль і керування доступом до різних сегментів даних, щоб лише потрібні люди могли отримати доступ до відповідної інформації.

Хмарні моделі захисту даних також повинні враховувати різні категорії даних, з якими організації зазвичай працюють. До них належать загальнодоступні дані, внутрішні дані компанії, до яких мають доступ лише працівники, конфіденційні дані, які можуть бути будь-якими: від відомостей про заробітну плату працівників до номерів соціального страхування клієнтів, а також дані з обмеженим доступом, які контролюються урядовими постановами.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу.

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

– Огляд існуючих систем для захисту інформації у хмарі з розмежуванням доступу.

– Дослідження системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу.

– Програмна реалізація системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для захисту інформації у хмарі з розмежуванням доступу.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Призначенням розробляємого, у результаті виконання бакалаврського проекту, програмного забезпечення є захист інформації у мережі хмари з розмежуванням доступу. Це досягається за рахунок введення технології інфраструктури відкритих ключів (Public Key Infrastructure, PKI). Технологія PKI дозволяє перевіряти й засвідчувати дійсність користувача. PKI забезпечує єдину ідентифікацію, автентифікацію й авторизацію користувачів системи, додатків і процесів і разом із цим гарантує доступність, цілісність і конфіденційність інформації.

IT-команди та команди DevOps мають можливість переглядати дані з висоти пташиного польоту, коли інфраструктура організації є повністю локальною. Однак це не той випадок, коли задіяні постачальники хмар, особливо з публічними хмарами. Хмарні постачальники працюють за моделлю «спільної відповідальності». Це означає, що в той час як хмарні постачальники беруть на себе відповідальність за сценарії захисту, такі як резервне копіювання та відновлення, клієнти (організації) відповідають за захист своїх власних даних і трафіку. Перехід до хмари дійсно зменшує накладні витрати, але це відбувається за рахунок контролю. Організації повинні покладатися на постачальників, щоб забезпечити безпеку основного фізичного обладнання та мережі. Незважаючи на те, що постачальники надають набір політик безпеки та елементів керування, частина сфери безпеки залишається за ними. Сьогодні інфраструктура побудована на кількох платформах , службах і програмах сторонніх розробників. В одній установці може бути задіяно кілька постачальників хмарних технологій. Це означає, що існує неузгодженість у тому, як дані зберігаються, керуються та захищаються. Інтеграція між цими кількома компонентами також має бути

безперебійною на рівні захисту даних. Ці виклики сигналізують про необхідність узгодженої та єдиної стратегії захисту даних або рішення. Ось тут і з'являються моделі захисту даних.

1.2 Область застосування

Областю застосування є захист інформації у мережі хмари з розмежуванням доступу.

Організації все більше визнають рентабельність і простоту роботи, які є результатом переміщення інфраструктури та активів у хмару або принаймні в гібридне середовище. Фактично, ще п'ять років тому хмарний трафік оцінювався в 3851 екзабайт.

Пандемія лише прискорила впровадження хмарних технологій організаціями будь-якого розміру. Дослідження Gartner показує, що цей сплеск впровадження хмарних технологій може призвести до того, що до кінця 2021 року 94% робочого навантаження компанії оброблятиметься хмарними центрами обробки даних.

Важливість захисту хмарних даних

Зі збільшенням кількості віддалених працівників зростає потреба у захисті доступу та переміщенні даних компанії в різних відомих і невідомих мережах. Надійна хмарна модель захисту даних слугує для:

1. Підтримує видимість даних

У більшості організацій дані передаються та зберігаються в кількох хмарах, службах і програмах. Сегменти даних легко провалитися крізь тріщини, коли задіяно так багато гравців. Хороша модель захисту даних у хмарі починається з карти кожного окремого фрагмента даних, яка періодично повторюється. Це забезпечує кращу гігієну даних і навіть допомагає скоротити витрати на зберігання даних завдяки ідентифікації несуттєвих даних.

2. Забезпечує цілісність даних

У сучасному ландшафті дані можуть зробити або порушити успіх компанії. Отже, дані мають бути точними, повними та надійними на будь-якому етапі свого життєвого циклу. Ось чому хмарний захист даних має сенс. Однією з невід'ємних цілей захисту хмарних даних є забезпечення того, щоб жодні дані всередині компанії не застаріли, не були маніпуляційні чи втрачені.

3. Підтримує комплаєнс

Правила відповідності відрізняються залежно від розташування даних. Хмарні постачальники несуть відповідальність за реалізацію політики відповідності. Але аудит на відповідність підлягає клієнтській організації. Це означає, що організації повинні мати цю інформацію з усіх хмарних інтеграцій, і вона має бути прозорою та достатньо зрозумілою для зовнішнього та внутрішнього аудиту.

Насправді «захист даних» набув набагато більшого значення, коли в середині 2018 року набув чинності загальний регламент захисту даних (GDPR). Це було одне з багатьох правил, прийнятих для забезпечення конфіденційності та безпеки організаційних даних, особливо з точки зору клієнта. Будь-яка компанія, яка використовує хмарних постачальників, розглядається як «контролер даних». За те, як обробляються дані, відповідає компанія, а не постачальник.

Кожен контролер даних повинен забезпечити:

– Усі дані компанії, що містять ідентифікаційну інформацію, які обробляються в хмарі, захищені.

– Дані можна передавати лише з території ЄС до попередньо схваленої території з аналогічним рівнем захисту.

– Угоди про рівень обслуговування для високого рівня захисту даних від хмарного постачальника.

Недотримання цих законів може призвести до судових позовів і великих штрафів з боку держави.

4. Забезпечує безпеку даних

Ще одним плюсом захисту даних у хмарі є систематичне визначення уніфікованих політик, які застосовуються на всіх рівнях. Політики щодо зберігання даних, словника даних, правил доступу та дозволів на основі ролей запобігають будь-якій формі вторгнення, захищаючи конфіденційні дані.

Хмарні моделі захисту даних зазвичай захищають дані в багатьох середовищах, надаючи засоби для шифрування та моніторингу даних у кожному файлі та на кожному рівні користувача. Така видимість допомагає IT-персоналу та командам DevOps контролювати складну установку.

Хмарний захист даних пропонує покрокову стратегію усунення будь-яких загроз для даних компанії. Сучасні хмарні рішення для захисту даних також завчасно виявляють ризики та аномалії даних, дозволяючи командам безпеки зупиняти будь-які спроби кібератак або впровадження шкідливого програмного забезпечення.

5. Керує зберіганням даних

Хмарна модель захисту даних забезпечує централізований спосіб моніторингу та доступу до даних у хмарній інфраструктурі. Це означає реєстрацію та звітування про кожен доступ до даних і маніпуляції. Це дозволяє IT-командам виявляти та знищувати шахрайські та підозрілі дані, гарантуючи, що сегменти даних зберігаються з відповідними засобами контролю на основі категорії, до якої вони належать.

6. Допомагає скласти план аварійного відновлення

Технічно план аварійного відновлення належить до плану безперервності бізнесу організації. Але це також одна із визначених цілей процесу захисту даних. Хмарні процеси захисту даних дозволяють:

– **Резервне копіювання:** резервне копіювання – це образ живої копії даних, який наразі запускає всю систему. Він використовується щоразу, коли програму або систему потрібно перезавантажити.

– **Відновлення:** це передбачає відновлення даних , а також серверів і мереж, які забезпечують потік цих даних, таким чином зберігаючи цілісність і узгодженість даних.

– **Доступність:** кожен план захисту даних повинен враховувати максимальний час простою, який може витримати система, не впливаючи на щоденні бізнес-операції. Цей показник досягається завдяки підтримці систем високої доступності, часто зі схожими та резервними компонентами інфраструктури.

Побудова хмарної моделі захисту даних спонукає компанії звертатися до своїх рішень DRP. Має сенс лише те, що дії DRP підпадають під захист даних, оскільки першим кроком захисту даних у хмарі є ідентифікація критичних активів даних.

7. Допомагає приймати зважені рішення

Одна з переваг використання аерофотозйому даних полягає в тому, що ключові аспекти цих даних можна ідентифікувати та формувати відповідно до потреб бізнесу. Це може дозволити керівництву приймати обґрунтовані рішення, маркетинговим групам відточувати свої цільові дослідження, а фінансовим командам зменшувати операційні витрати.

Оскільки переваги хмарної моделі захисту даних значно переважують труднощі переходу на хмарні або гібридні рішення, не дивно, що до 2024 року ринок захисту даних, за оцінками, перевищить 158 мільярдів доларів США

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Free Hide Folder

Free Hide Folder – безкоштовна комп'ютерна програма для забезпечення безпеки персональних, особистих даних, що зберігаються на ПК.

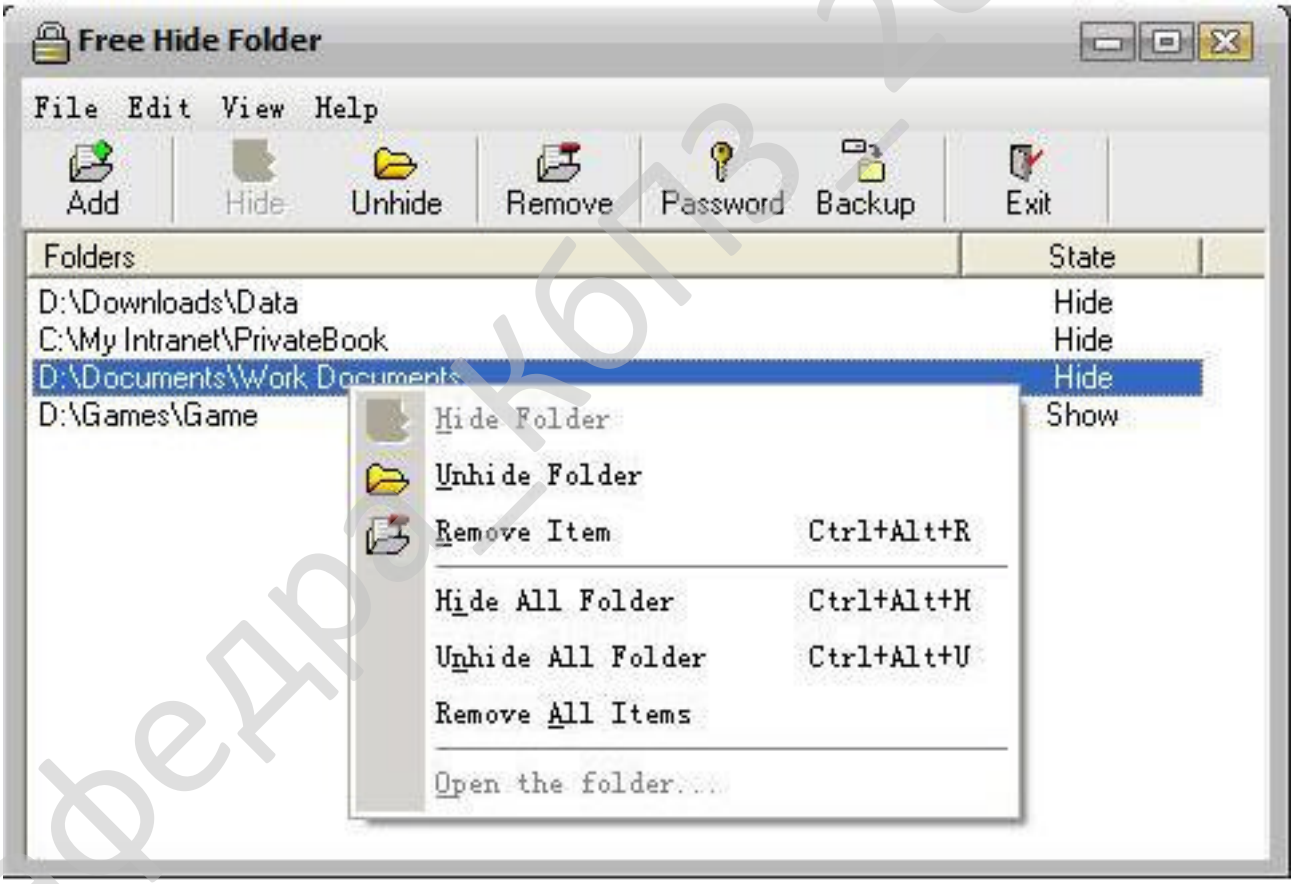

Рисунок 2.1 – Інтерфейс користувача Free Hide Folder

Free Hide Folder – дана програма захищає вашу інформацію, роблячи її схованою від чужих поглядів, тобто невидимою для сторонніх, плюс захист паролем. Збереже конфіденційність персональних даних, що зберігаються на комп'ютері, сховає особисті папки від випадкового перегляду, використання, внесення змін або видалення їхніми іншими користувачами ПК. Ніхто крім вас не буде знати де перебувають потрібні вам файли (папки). Крім того, Free Hide Folder має захист паролем, що ви можете змінити або видалити в будь-який час.

Приховуйте папки просто декількома клацаннями миші, запустіть програму Free Hide Folder, знайдіть і виберіть папку(і), що ви хочете сховати, натисніть кнопку "Сховати папку" – готово.

Основні характеристики Free Hide Folder:

– приховання папок повністю. Без вашого самостійного показу, ніхто не зможе їх знайти;

– захист паролем при запуску програми;

– одночасне приховування будь-якої кількості папок;

– простій у використанні користувальницький інтерфейс;

– програма безкоштовна;

– 100% відсутність шпигунських програм, програмне забезпечення не містить Spyware, Adware і інших вірусів.

Folder Lock

Folder Lock – швидке шифрування, захист паролем файлів і папок. Захищає важливі для вас особисті дані від випадкових користувачів. Програма Folder Lock пропонує для цього швидкий спосіб шифрування й захисту паролем файлів і папок. Крім того, Folder Lock має додаткові особливості, такі як, блокування, схований (невидимий) режим, запобігання спроб злому, мобільність, очищення історій (усього більше 20), що дозволяє зберегти конфіденційність і безпеку особистих даних з обліком навіть особливих потреб кожного.

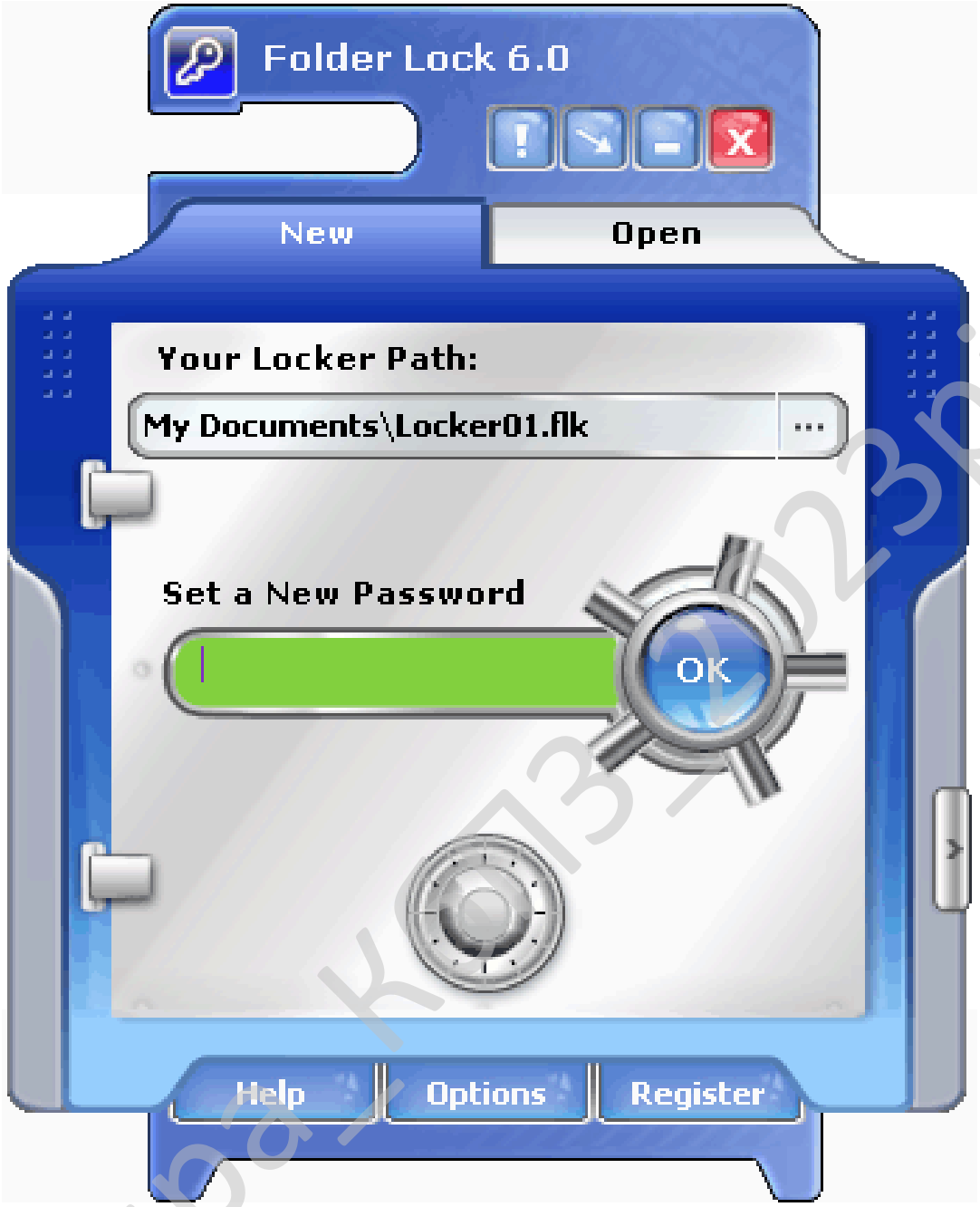

Рисунок 2.2 – Інтерфейс користувача Folder Lock

Функції Folder Lock:

– програма для швидкого блокування файлів, програмне забезпечення для шифрування файлів, захисту паролем папок, захисту USB накопичувачів, замок на CD / DVD диски;

– програма підтримує всі формати Windows 7, Vista, XP, 2000 , у тому числі їх 64-бітні версії;

– програма створює зашифроване сховище, називане "Сейф". У ньому ви можете зберігати безліч своїх особистих файлів / папок і захистити їхнім паролем одним натисканням кнопки. "Сейфи" є портативними, так що ви можете безпечно їх передавати, робити резервне копіювання, тримати на USB, CD / DVD, ноутбуці або передавати по електронній пошті. Ці "Сейфи" є Undeletable на комп'ютері, де Folder Lock установлений;

– ви можете створити будь-яку кількість "Сейфів", у тому числі з різними паролями або, наприклад, два-в-одному (замок і шифрування). Це дозволяє вибрати або 256-біт AES шифрування "на льоту" або блокування файлів, папок, дисків у будь-якому місці на комп'ютері;

– програма має додаткові особливості, які не пропонує ніяке інше ПЗ для шифрування, а саме: моніторинг злому, схований режим, автоматичний захист, підтримка автозапуска портативних USB, очищення історій, віртуальна клавіатура. Крім того, неможливо перейменувати, перемістити або видалити файли / папки без правильного пароля, що запобігає втраті даних.

Особливості й переваги Folder Lock:

– швидке 256-бітне AES шифрування файлів "на льоту";

– шифрування файлів і папок будь-якого типу;

– шифрування даних на дисках;

– легкий захист папок паролем;

– блокування файлів, папок, дисків ;

– підтримка 64-розрядних ОС;

– підтримка контекстного меню в провіднику;

– File Lock , Folder Lock, Drive Lock – повний пакет, усе в одному;

– повна переносимість (пароль, захист на USB, CD / DVD і ін. накопичувачах / пристроях);

– резервне копіювання, передача захищених паролем "Сейфів" у будь-якому місці. Видалення "Сейфа" вимагає входу в Folder Lock;

– режим невидимості. Ця функція може ефективно приховувати всі сліди установки Folder Lock на комп'ютері;

– очищення історій, видалення ярликів. Автоматичне видалення слідів діяльності (історій) на ПК при закритті програми;

– контроль підроблених спроб уведення пароля з можливістю автоматичного вимикання комп'ютера;

– найшвидше File Encryption Software.

Програма Folder Lock – це забезпечення повної конфіденційності. Якщо ви стурбовані крадіжкою, втратою особистих даних, витоком інформації, небезпекою від вірусної атаки, то Folder Lock – для Вас!

Folder Lock найбільше часто завантажуєме File Encryption Software на ринку аналогічного ПЗ.

Крім Folder Lock на сайті представлені наступні програми для забезпечення повної конфіденційності особистих даних, захисту їх від витоку / крадіжки, копіювання й ін. втрат, для установки різних прав доступу до ваших файлів (папкам, дискам, установленим програмам) або для забезпечення неприступності до файлів, їхнього приховання, захистом від запису або видалення.

– USB Secure – захист паролем USB і інших флеш-накопичувачів. Програма не вимагає установки, не вимагає прав адміністратора й працює з усіма типами портативних пристроїв (USB, флеш-диски, флеш-карти, карти пам'яті, зовнішні жорсткі диски й т.д.). Більше не треба турбуватися про втрату або крадіжки даних з flash накопичувачів, якщо ви захистили їх USB Secure.

– USB Block – запобігає витоку й копіюванню ваших даних, що зберігаються на USB, зовнішніх дисках, CD / DVD і інших типах переносних (портативних) пристроїв. Установка Device Block дозволить вам блокувати всі такі диски й пристрої, які не належать вам.

– Folder Protect – захист за допомогою пароля й установка різних прав доступу до ваших файлів, папкам, дискам, установленим програмам і різним

типам файлів. Folder Protect дозволяє вам настроїти вашу безпеку й зробити файли недоступними, схованого або захищеними від запису або видалення.

– Copy Protect – захист від копіювання, що дозволяє запобігти незаконному копіюванню / поширення аудіо, відео й ін. файлів. Програма підтримує більшість форматів документів, аудіо, відео, файлів зображень. Захист від копіювання здійснюється шляхом перетворення мультимедійних файлів у виконуються приложения, що, які працюють тільки на диску, на якому вони зроблені.

– History Clean – дозволить вам очистити сліди вашої роботи на комп'ютері й видалить кукі, кеш, історію, уведені URL, автозаповнення даних, останні запущені документи, відвідані вебсайти, видалені файли й багато чого іншого.

SCARABAY

SCARABAY – зручний менеджер паролів з автозаповненням. SCARABAY – програма для зберігання ваших секретних відомостей:

– одним кліком заповнить у браузері логін і пароль;

– перетягніть мишею логін, пароль, E-mail у будь-які поля будь-яких вікон.

SCARABAY – менеджер паролів, збереже й захистить вашу конфіденційність (особисті дані, персональну інформацію) на комп'ютері, таку, як логіни, паролі, серійні номери, пін коди, номера карт, телефони й т.д..

Дана програма має потужний генератор паролів, портативна, можна скопіювати на флешку й користуватися на будь-якому комп'ютері й у будь-якому місці (будинку, на роботі, в інтернет-кафе).

Рисунок 2.3 – Інтерфейс користувача SCARABAY

Основні особливості:

– зберігає будь-яку конфіденційну інформацію;

– створення, копіювання, редагування, видалення даних;

– шифрування даних;

– захист даних паролем;

– створення паролів будь-якої складності;

– автозаповнення логінів і паролів на веб сторінках. Підтримує браузери: Internet Explorer, Maxthon Browser, Avant Browser, NetCaptor, Netscape;

– створення необмеженої кількості користувачів;

– деревоподібне подання даних. Можна змінювати папки, створювати нові директорії, директорії в директоріях і т.д.;

– екранна клавіатура;

– створює резервну копію даних;

– згортається в системний трей;

– працює з портативних носіїв;

– зміна зовнішнього вигляду за допомогою скинів і т.п.

Password Folder

Password Folder – спосіб захисту вашого приватного життя – особистої інформації (файлів і папок на ПК) від сторонніх очей. Password Folder – програма призначена для захисту паролем файлів і папок від сторонніх очей, читання або зміни їх в ОС Windows 2000, XP, Vista, 7. Password Folder працює як сейф, просто перетягніть папки або файли, які ви хочете сховати або захистити паролем у папку, після чого ніхто не зможе бачити, читати або змінювати їх. Якщо ви стурбовані крадіжкою, втратою або витоком даних, то Password Folder є ідеальним інструментом для вас.

Основні переваги:

– програма вкрай проста у використанні. Просто перетягніть файл в Password Folder, щоб захистити його. Інтуїтивно зрозумілий інтерфейс робить утиліту ідеальної навіть для починаючих користувачів ПК;

– приховання від перегляду. Приховуйте папки, файли, фотографії й відео на своєму комп'ютері від пошуку їхніми сторонніми особами. Ніхто не зможе їх бачити, крім вас;

– блокування доступу. Легко блокувати доступ до файлів, папкам і програмам, які вами обрані. Ви можете захистити паролем папку й файли, при цьому інші користувачі хоч і можуть їх бачити, але неможливо одержати доступ для використання, читання або перегляду змісту;

– створення захисту. Захистіть ваші файли від зміни іншими. Ніхто не зможе змінити зміст ваших файлів без дозволу. Блокування файлів і папок буде незмінна;

– захист. Зберігаєте особисті дані, файли, фотографії або відео в Password Folder, єдиний спосіб одержати доступ до цих файлів – ваш пароль;

– безпечно й надійно. Password Folder забезпечує безпечний і надійний захист без яких-небудь вірусів, рекламного й шпигунського програмного забезпечення.

Password Folder – більше немає крадіжки, втрати, витоку потрібних саме вам даних, особистої інформації, що зберігаються на комп'ютері. Важливі для вас файли або папки захищені від випадкового видалення, заміни, перегляду або використання іншими.

Крім Password Folder, на сайті розроблювача IObit представлена програма Random Password Generator (генератор випадкових паролів) – безпечний генератор паролів і менеджер:

– генерація випадкових паролів;

– керування згенерованими паролями;

– настроювання паролів.

Random Password Generator призначень, щоб допомогти вам створити безпечні паролі, які вкрай важко зламати або вгадати. Генерація заснована на сполученні верхнього й нижнього регістра, цифр і розділових знаків. Основні переваги:

– Password Generator (генератор паролів). Безпечне й швидке створення потужних випадкових паролів, які практично неможливо зламати.

– Password Manager (менеджер паролів). Керування вашими паролями. Зберігаєте свої згенеровані паролі в базі даних з короткими коментарями, плюс ідентифікація й редагування (безпечно й впорядковано). Редагуванням ви можете змінити пароль відповідно до попереджень Random Password Generator, щоб одержати сильний і безпечний пароль.

– Password Checker – рада про згенерований пароль, тобто чи є він "сильним" або "слабким", виділення "слабких" паролів.

– Privacy Protected – установка пароля на запуск / відкриття Random Password Generator.

Таким чином, це надзвичайно простий і зручний спосіб для користувачів ПК автоматично генерувати багато важких і складних паролів для захисту їхньої особистої інформації.

Hide Folders

Hide Folders – програма приховує, блокує й захищає паролем ваші особисті файли й папки від інших користувачів ПК.

Рисунок 2.4 – Інтерфейс користувача Hide Folders

Hide Folders являє собою інноваційне програмне забезпечення, що дозволить вам захистити потрібну інформацію, що зберігається на жорсткому диску вашого комп'ютера.

Можливість легко захистити будь-яку, необмежену кількість папок і файлів на ПК. Ви можете зробити файли й папки недоступними, невидимими, захистити їх від модифікації

Доступно 4 методи захисту:

– Сховати.

– Блокування.

– Сховати й замок.

– Тільки читання.

Захищені папки й файли недоступні будь-яким користувачам, незалежно від того, яким образом вони намагаються одержати до них доступ (локально або по мережі хмари).

Програма має:

– ефективний механізм захисту;

– інтуїтивно зрозумілий інтерфейс;

– набір з тонкого настроювання параметрів, що відповідає потребам кожного користувача Windows від початківця до досвідченого;

– захист паролем при запуску й видаленні програми;

– видалення Hide Folders із системи не виявить сховані папки;

– папки залишаються схованими, навіть якщо ваш комп'ютер працює в безпечному режимі;

– підтримуються файлові системи NTFS, FAT32 і FAT плюс ряд інших значних функцій і особливостей програми.

Hide Folders – це багатомовне програмне забезпечення, користувальницький інтерфейс не тільки підтримує різні мови, але й має можливість легкого перекладу на нові, а також підтримка папок із символами національних мов.

Необхідно відзначити, що програмні продукти цих розроблювачів по захисту особистих файлів і папок ефективно використовується користувачами з 2001 року. Ніхто, крім Вас не буде мати доступ до схованих папок.

My Lockbox

My Lockbox™ – програма для надійного зберігання особистих файлів.

Рисунок 2.5 – Інтерфейс користувача My Lockbox

My Lockbox™ – безкоштовна програма, що дозволяє встановлювати пароль і забирати із провідника Windows директорії й вхідні в них папки й файли, тому що начебто їх там ніколи й не було.

Програма My Lockbox™ забезпечує безпеку будь-якої папки на комп'ютері захистом паролем. Також, захищену папку можна сховати від сторонніх користувачів ПК (зробити невидимої), у тому числі й від системного адміністратора. Після установки захисту, неможливо одержати доступ до "сейфа" не тільки на локальному комп'ютері, але також і віддалено, з мережі хмари Інтернет. Програма забезпечує захист папки навіть у безпечному режимі Windows.

My Lockbox™ дуже простий у використанні, панель керування дозволяє легко змінювати основні параметри: місце розташування, статус захисту, пароль

My Lockbox™ – це ефективний захист, простий у використанні й безкоштовний.

Особливості й переваги програми:

– дуже проста у використанні;

– захист паролем практично будь-якої папки на комп'ютері;

– миттєвий захист;

– захищені папки недоступні навіть системним адміністраторам;

– Lockbox папки недоступні як локально, так і віддалено;

– Lockbox папки можуть бути захищені в безпечному режимі Windows;

– підтримка Windows XP x64;

– підтримка гарячих клавіш;

– дружелюбний і простій у використанні користувальницький інтерфейс;

– безкоштовна програма.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Програмне забезпечення написане мовою C#. Ця мова обрана виходячи з наступних міркувань. C# – строго типізована об’єктно-орієнтована мова, призначена для розробки різноманітних безпечних і потужних додатків, виконуваних у середовищі .NET Framework. Мовою C# можна розробляти звичайні клієнтські додатки Windows, веб-служби XML, розподілені компоненти, додатки типу “ сервер-клієнт”, додатки баз даних і багато яких інших. В Visual C# 2008 є розширений редактор коду, конструктори зі зручним користувальницьким інтерфейсом, вбудований відладник і багато інших засобів, покликані спростити розробку додатків мовою C# версії 3.0 і .NET Framework версії 3.5.

Синтаксис 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 C#. Розроблювачі, що знають кожну із цих мов, як

правило, зможуть домогтися ефективної роботи з мовою C# за дуже короткий час. Синтаксис 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. 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 C# 3.0 вираження LINQ (Language-Integrated Query) роблять строго-типізований запит першокласною конструкцією мови.

Як об'єктно-орієнтована мова, C# підтримує поняття інкапсуляції, спадкування й поліморфізму. Всі змінні й методи, включаючи метод `Main` – крапку входу додатка – інкапсуюється 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 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 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 C# можна використовувати процес, що називається "Interop". Процес Interop дозволяє програмам на C# виконувати практично будь-які дії, які може виконувати вихідний додаток на C#. Мова C# підтримує навіть покажчики й поняття "небезпечного" коду для тих випадків, коли прямий доступ до пам'яті має вкрай важливе значення.

Процес побудови 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 C# може бути визначене будь-яке число класів, структур, інтерфейсів і подій.

Архітектура платформи .NET Framework

Програма мовою 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 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 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, призначень для певної системи. Далі показані відносини під час компіляції й час виконання між файлами з вихідним кодом C#, бібліотеками класів .NET Framework, складаннями й середовищем CLR.

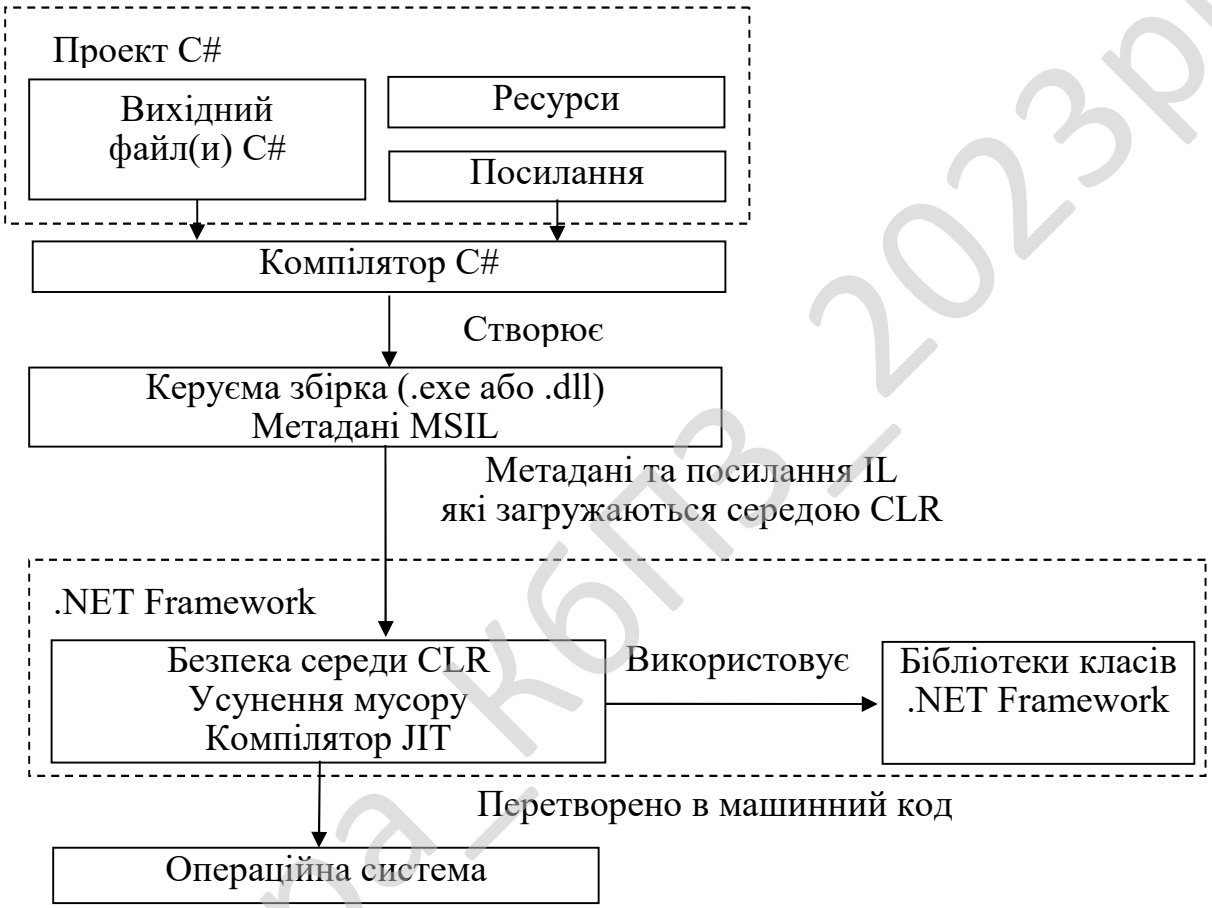

Рисунок 2.6 – Створення програми на C#

Взаємодія між мовами є ключовою особливістю .NET Framework. Оскільки код IL, створюваний компілятором C# відповідає специфікації CTS, код IL на основі 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 додатку мовою C# бібліотека класів .NET Framework інтенсивно використовується для "устрою" коду.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) 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 кібербезпеки 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 кібербезпеки 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

Де точка розмежування з хмарними сервісами?

Точка розмежування важлива для постачальників послуг, оскільки вона обмежує їхній ризик і таким чином обмежує їх сферу відповідальності та витрати, але зростаюча популярність хмарних обчислень і хостингових комунікаційних послуг обіцяє кардинально змінити картину – можливо, найбільш драматично в електромонтажна шафа.

До розміщення комунікацій на хостингу комунікаційні шафи більшості підприємств були заповнені каліопами обладнання для передачі даних і голосу, включаючи CSU/DSU, маршрутизатори, брандмауери, пристрої безпеки мережі, застарілі KSU/PBX, а іноді й відносно нові IP-PBX.

Завдяки хмарним обчисленням і хостинг-сервісам більша частина логічного та комунікаційного програмного забезпечення переміщується до центру обробки даних постачальника послуг, що дозволяє видалити значну частину старого обладнання з монтажної шафи та замінити на нове покоління клієнтського обладнання, Multi-Service Бізнес-шлюз (MSBG).

Виготовлений компаніями, що займаються обробкою даних і обладнанням VoIP , MSBG – це єдиний пристрій, який розриває фізичне з’єднання з постачальником послуг, перетворюючи фізичні протоколи та протоколи даних відповідно до потреб клієнта. У багатьох випадках MSBG також включає маршрутизатор, брандмауер і програмне забезпечення безпеки, а також забезпечує певну локальну логіку для виживання у випадках, коли підключення до мережі сильно перевантажене або втрачено.

Переваги заміни всього окремого обладнання пропозицією розміщених послуг і MSBG включають зменшення займаної площі та менше місця на

полицях, зменшення споживання електроенергії, меншу ймовірність випадкового від'єднання кабелів, простішу можливість керування та, як правило, більшу вартість. ефективний спосіб надання послуг.

З меншою кількістю обладнання, меншим простором і нижчою ціною – що не подобається в цьому новому пристрої?

Для постачальника послуг розміщені служби та MSBG значно розширюють свою відповідальність і змінюють точку розмежування. З розміщеною системою зв'язку їхня відповідальність закінчується MSBG, настільним комп'ютером чи, можливо, навіть IP-телефоном?

Звичайно, це буде відрізнятися залежно від постачальника послуг, але здається логічним, що MSBG буде природним вибором для пункту розмежування. Це змусить клієнтів відповідати за їхній мережевий комутатор, кабелі та IP-телефони. Але чи означає це, що постачальник послуг відповідає за маршрутизатор даних, брандмауер і безпеку мережі, якими часто керує кінцевий клієнт?

Можливо, це підходить для деяких малих і середніх підприємств, але багато IT-фахівців мали б проблеми, якби вони не могли керувати призначенням IP-адреси, переадресацією портів та іншими політиками безпеки у власній мережі. Уявіть, що вам потрібно зателефонувати своєму постачальнику послуг, щоб налаштувати параметри брандмауера, оскільки ви придбали IP-камеру безпеки. Це було б програшно для обох сторін.

Рішенням є «віртуальна точка розмежування» – точка в MSBG, яка позначає кінець відповідальності постачальника послуг. Один розумний момент буде між модемом даних і брандмауером . Це призвело б до того, що постачальник послуг відповідав би за модем, а клієнт відповідав би за брандмауер, маршрутизатор та всі інші функції з цього моменту.

Для багатьох компаній добре підходять розміщені або «хмарні комунікації», оскільки вони перекладають більшу частину відповідальності та обладнання на домен постачальника послуг. Для інших відмова від контролю та

нагляду за своєю системою зв’язку та безпекою є надто важкою. Знання точки розмежування є важливим як для постачальника послуг, так і для кінцевого споживача, а запровадження MSBG відкриває нові можливості та виклики для обох сторін.

Впровадження розміщених і хмарних служб разом із MSBG створюють чудову комбінацію, яка покращує швидкість розгортання, зменшує витрати та покращує обслуговування. Але перш ніж перейти до розміщеного рішення, уважно подивіться на точку розмежування та переконайтеся, що вона відповідає вашим планам контролю та безпеки.

Правильна комбінація керування дозволить вам зробити вибір щодо вашої локальної конфігурації, мережевого обладнання та кінцевих точок, що дозволить вам не застрягти з версією телефону Bell Black 21 століття.

Історія моделей обслуговування (SaaS, PaaS та IaaS)

У 2011 році Національний інститут стандартів і технологій (NIST) надав визначення хмарних обчислень, яке складається з трьох моделей обслуговування, чотирьох моделей розгортання та п’яти основних характеристик (спеціальна публікація NIST 800-145).

Цей документ мав на меті слугувати засобом для надання стандартів і рекомендацій, особливо під час порівняння хмарних служб і стратегій розгортання, а також забезпечити базову лінію для найкращого використання хмарних обчислень.

Три моделі обслуговування: SaaS (програмне забезпечення як послуга), PaaS (платформа як послуга) та IaaS (інфраструктура як послуга). Це було минуле, і моделі мають розвиватися, щоб охопити нові платформи.

Інновації та розробка програмного забезпечення як ключовий драйвер змін

Виведення на ринок нової та диференційованої цінності зараз є конкуренційною необхідністю, і підприємства, які найкраще організовані для того, щоб швидше впроваджувати інновації в повторюваний спосіб, є лідерами

ринку. Корпоративне розгортання хмарних обчислень дозріло та змінилося з нагальною потребою привнести нові цінності, а програмне забезпечення є драйвером змін.

Це справедливо для рівня інфраструктури, рівня обслуговування та рівня додатків, де ми спостерігаємо поширення контейнерів, розвиток Kubernetes (K8s), появу периферійних обчислень і широке впровадження безсерверної архітектури, все це в службі розробників, щоб вони могли швидше вивести цінність на ринок.

Спроба вставити нові архітектури в структуру SaaS-PaaS-IaaS 2011 року – це все одно, що вставити квадратний кілочок у круглий отвір!

Нові моделі обслуговування

У своїй основі *хмарна* модель спільної відповідальності забезпечує чітке розмежування обов'язків між хмарними постачальниками (Amazon Web Services, Microsoft Azure, Google Cloud Platform або, у більш загальному сенсі, постачальниками платформи) і споживачами хмарних технологій або власниками програм (підприємствами та стартапами). однакові).

На діаграмі нижче показано відмінності у відповідальності в різних моделях послуг, які ми бачимо зараз.

Деякі ключові моменти:

Поступово все більше й **більше відповідальності беруть на себе постачальники платформ** , повертаючи власникам додатків обов'язки, не пов'язані з логікою додатків.

Якщо один рух праворуч зменшить **операційні витрати та накладні витрати, оскільки постачальник платформи бере на себе більше відповідальності** .

Коли ми досягаємо таких платформ, як NoCode/SaaS, відповідальність самих розробників зменшується. Сприяє **виходу на новий рівень розробників, які не є хардкорними програмістами** .

Нові моделі послуг, які розвинулися з того часу, окрім IaaS, PaaS і SaaS, визначені нижче.

1. Керовані K8s як послуга (K8s-aaS)

Керований Kubernetes – це найпоширеніша площина керування керованими послугами як послуга (CPaaS), яку надають більшість хмарних постачальників. У цьому випадку площиною керування Kubernetes керує постачальник платформи з певною конфігурацією площини керування (також відомої як K8s Master node), наданою власником програми. Життєвий цикл площини даних і керування нею виконує власник програми.

Це найкраще працює, коли програма має певні потреби з площини даних, оптимальне масштабування за вартістю є більш важливим, ніж додаткові операційні накладні витрати, або коли програма має бути мультихмарною портативною програмою.

Прикладами є Amazon Elastic Kubernetes Service (EKS), Azure Kubernetes Service (AKS) і Google Kubernetes Engine (GKE). Сервіс AWS Elastic Compute Service (ECS) є прикладом керованого рівня керування, який не належить Kubernetes.

2. Контейнер як послуга (CaaS)

У випадку CaaS власник програми надає контейнери програми, а постачальник платформи керує площиною керування та даних. Це означає, що користувачам додатків не потрібно керувати серверами (VM), масштабувати та виправляти ОС хоста або запускати та виключати сервери, крім усіх функцій, які надає CPaaS.

Ці служби також називаються безсерверними, оскільки власник програми звільняється від багатьох обов'язків щодо керування сервером. Послуги найкраще підходять для випадків, коли це не архітектура приводу подій і власник програми менш чутливий до масштабування витрат.

Прикладами рішень Containers -as-a-Service (CaaS) є такі рішення, як Amazon Web Services (AWS) Fargate (як ECS Fargate, так і EKS Fargate), Azure Container Instances (ACI) і Google CloudRun.

3. Функція як послуга (FaaS)

У випадку FaaS власник програми надає бізнес-логіку разом із рівнями, на яких виконується функція. Ці функції створені, упаковані та запущені постачальником послуг. Площина керування послугами та площина даних повністю опікуються постачальником послуг.

Ця служба найкраще підходить для додатків без стану, керованих подіями.

Прикладами цієї служби є AWS Lambda, Azure Functions і Google Cloud Functions.

4. NoCode-as-a-Service (NCaaS)

У NCaaS логіка коду надається власником програми. Постачальник послуг створює код із специфікації та конфігурації, а потім збирає, пакує та запускає програмне забезпечення.

Ще одна подібна, але дещо інша версія цього – Low-Code-as-a-Service (LCaaS).

Оскільки програмування потребує мало, **ці платформи найкраще розроблені навіть для нетехнічних користувачів для створення програм** . Це призведе до величезного зростання в найближчі роки та **призведе до величезного зростання розробників програмного забезпечення** .

Прикладами цієї служби є Azure Power Apps, Google AppSheet і AWS Honeycode.

5. Безсерверний

Безсерверні платформи дозволяють розробникам швидше розробляти та розгортати, надаючи можливість легкого переходу до власних хмарних служб без необхідності керувати інфраструктурою, включаючи кластери контейнерів або віртуальні машини.

Приклади: у наведеній вище моделі платформи CaaS/FaaS і NCaaS розглядатимуться як безсерверні.

Вибір платформи для ваших програм

Підсумовуючи, майбутній ландшафт додатків дуже різноманітний, високогібридний і багатохмарний. Платформи корпоративних хмарних обчислень включатимуть широкий спектр інфраструктури, рівнів обслуговування та API, включаючи безсерверні та серверні програми, локальні та хмарні програми.

Не буде моделі «один розмір для всіх» або єдиного практичного правила. У справжньому хмарному стилі це гнучке та еластичне рішення для підтримки масштабованого та безпечного середовища, яке розвивається разом зі змінами організацій.

Опис SSL/ TLS

TLS – криптографічний протокол, що забезпечує захищену передачу даних між вузлами в мережі хмари Інтернет. TLS-протокол заснований на Netscape SSL-протоколі версії 3.0 і складається із двох частин – TLS Record Protocol і TLS Handshake Protocol. Розходження між SSL 3.0 і TLS 1.0 незначні, тому далі в тексті термін «SSL» буде відноситися до них обох. TLS Working Group, заснована в 1996 році, продовжує працювати над протоколом.

TLS надає можливості автентифікації й безпечної передачі даних через Інтернет з використанням криптографічних засобів. Часто відбувається лише автентифікація сервера, у той час як клієнт залишається неавтентифікованим. Для взаємної автентифікації кожна зі сторін повинна підтримувати інфраструктуру відкритого ключа (PKI), що дозволяє захистити клієнт-серверні додатки від перехоплення повідомлень, редагування існуючих повідомлень і створення підроблених.

SSL містить у собі три основних фази:

– Діалог між сторонами, метою якого є вибір алгоритму шифрування.

– Обмін ключами на основі криптосистем з відкритим ключем або автентифікація на основі сертифікатів.

– Передача даних, шифруємих за допомогою симетричних алгоритмів шифрування.

Опис протоколу обміну ключами Діффі-Хеллмана

Ціль алгоритму полягає в тому, щоб два учасники могли безпечно обмінятися ключем, що надалі може використовуватися в якому-небудь алгоритмі симетричного шифрування. Сам алгоритм Діффі-Хеллмана може застосовуватися тільки для обміну ключами.

Алгоритм заснований на труднощі обчислень дискретних логарифмів. Дискретний логарифм визначається в такий спосіб. Уводиться поняття примітивного кореня простого числа Q як числа, чиї ступені створюють всі цілі від 1 до $Q - 1$. Це означає, що якщо А є примітивним коренем простого числа Q, тоді числа: $A \bmod Q$, $A^2 \bmod Q$, . . . , $A^{Q-1} \bmod Q$, є різними й складаються із цілих від 1 до $Q - 1$ з деякими перестановками. У цьому випадку для будь-якого цілого $Y < Q$ і примітивного кореня A простого числа Q можна знайти єдину експоненту X, таку, що:

$$Y = A^X \bmod Q, \text{ де } 0 \le X \le (Q - 1).$$

Експонента X називається дискретним логарифмом, або індексом Y, по підставі $A \bmod Q$. Це позначається як:

$$\text{ind}_{A, Q}(Y).$$

3.2 Розробка структурної схеми

Найкращі методи захисту хмарних даних:

1. Забезпечте вбудовану безпеку на всіх етапах

Рішення захисту даних має гарантувати, що всі збережені дані зашифровані. Постачальники повинні мати SLA, які підтверджують заходи захисту даних. Навіть із гарантованим хмарним постачальником шифруванням можна створити додатковий рівень безпеки шляхом шифрування на рівні файлів. Це передбачає шифрування даних навіть перед надсиланням для зберігання в хмарі.

Іншим заходом безпеки є шардинг даних, які зберігатимуться в кількох

точках розташування. Через це хакерам важко отримати цілі фрагменти даних, маючи в своєму розпорядженні лише фрагменти. Передача даних «точка-точка» також може бути посилена додатковим шифруванням за допомогою традиційного SSL.

2. Створити централізований контроль

Узгодженість є ключем до захисту хмарних даних. Весь доступ користувачів і програм на будь-якому рівні системи має проходити через цей компонент. Це можна зробити за допомогою рішення IAM або PAM .

Потрібно встановити правила доступу. Хороший захист даних у хмарі також шукає контекстні зміни поведінки. Це означає, що він надає доступ відомому користувачеві на основі інших факторів, таких як час запиту, місцезнаходження запиту тощо. Необхідно встановити MFA, особливо якщо йдеться про конфіденційні та обмежені дані. Необхідно встановити політику даних із можливим впровадженням інструменту керування даними.

Централізований моніторинг також має дозволити IT-команді виявляти тіньові IT, які досить часто зустрічаються в хмарних розгортаннях. Хмари полегшують запуск, налаштування та запуск сторонніх служб, що дозволяє багатьом шахрайським системам використовувати хмарні ресурси.

3. Забезпечте стабільну та надійну роботу користувача

Легко переборщити з безпекою, призупиняючи та перенаправляючи кожен біт даних у трафіку системи. Це збільшує використання пропускної здатності Інтернету та може призвести до зниження продуктивності програм.

Фреймворк SASE, визначений Gartner, рекомендує розміщувати механізми захисту даних ближче до рівня користувача, щоб запобігти зворотному зв’язку. Хмарна модель захисту даних повинна забезпечити оптимальний баланс між зручністю використання та безпекою.

4. Автоматизуйте, де це можливо

Конкурентні постачальники хмарних технологій тепер надають вбудовані алгоритми для виявлення можливих вразливостей . Вони використовують AI,

щоб допомогти цьому процесу. Автоматизоване створення виправлень на основі виявлених ризиків є величезною перевагою для команди IT.

Автоматизоване створення журналів і звітів також дозволяє командам безпеки виявляти підозрілу поведінку щодо доступу до даних і маніпуляцій. Це також допомагає під час перевірок відповідності. Оновлення безпеки також мають бути автоматизованими та такими, що можна налаштувати.

5. Задокументуйте відповідальність організації

Існує місце для плутанини в моделі спільної відповідальності з точки зору обов'язків. Організації повинні ретельно пройти через свої SLA з постачальниками хмарних послуг. В ідеалі постачальник забезпечує безпеку апаратного та програмного забезпечення, тоді як організація несе відповідальність за власні активи даних.

Граничні випадки, як-от протокол у разі порушення даних, також необхідно узгодити з постачальником хмарних технологій. Усе це має бути задокументовано прозорим і доступним способом, гарантуючи, що зацікавлені працівники добре знають свої обов'язки. Це також стане в нагоді під час перевірок відповідності.

6. Забезпечте сумісність

Кожній організації потрібна єдина платформа захисту даних, яка підтримує як локальні, так і хмарні рішення. Це робить його більш масштабованим. Багатохмарне розгортання потребує плану захисту даних у хмарі, який повинен вирішувати проблеми з даними в кількох програмах і розгортаннях. Наприклад, безпечний веб-шлюз не може бути єдиним існуючим рішенням. Для захисту програм SaaS також може знадобитися брокер безпеки доступу до хмари.

Сама природа хмари передбачає інтеграцію та передачу даних між кількома службами, як внутрішніми, так і сторонніми. План захисту даних у хмарі має гарантувати відсутність прогалин у безпеці під час інтеграції. Організація повинна підібрати заходи безпеки, яких не виконує

постачальник послуг.

7. Перевірте резервне копіювання та відновлення

Після того як безпека та конфіденційність даних охоплені, захист даних вступає в дію. Дані повинні бути відтворені та збережені у вторинному місці, що дозволяє підприємствам відновити роботу в разі будь-якої перерви. Показники аварійного відновлення організації, такі як цільовий час відновлення (RTO) і прикметники точки відновлення (RPO), повинні узгоджуватися з тим, що надає постачальник хмарних технологій. Вони мають бути вбудовані безпосередньо в SLA.

Опитування IDG щодо хмарних обчислень у 2022 році показало, що більше 75% витрат на IT зараз спрямовано на хмару. Опитування також показало, що близько 92% організацій так чи інакше працюють у хмарі. Отже, настав час, щоб організації почали серйозно ставитися до захисту хмарних даних.

Однак важливо зазначити, що для того, щоб хмарний план захисту даних був справді ефективним, він має бути розроблений спеціально для організації, пам'ятаючи про масштабованість. Хоча спочатку це може здатися страшним, фіскальних, операційних і репутаційних переваг занадто багато, щоб їх ігнорувати. Якщо вашій організації бракує ресурсів для створення плану захисту даних у хмарі, вона може дослідити захист даних як послугу (DPaaS) для експертної підтримки.

Розроблене програмне забезпечення представляє із себе набір компонентів призначених для забезпечення політики безпеки як у вже існуючих, так і в створюваних мережних інформаційних системах.

Розроблене програмне забезпечення дозволяє забезпечити захист переданої по мережі хмари інформації, строгу взаємну автентифікацію користувачів і серверів, гнучке розмежування доступу. Для реалізації цих функцій у системі використовуються SSL/TLS протоколи й X.509 цифрові сертифікати, тобто універсальні, що стали стандартом де-факто, механізми, підтримувані практично всіма розповсюдженими Веб-агентами.

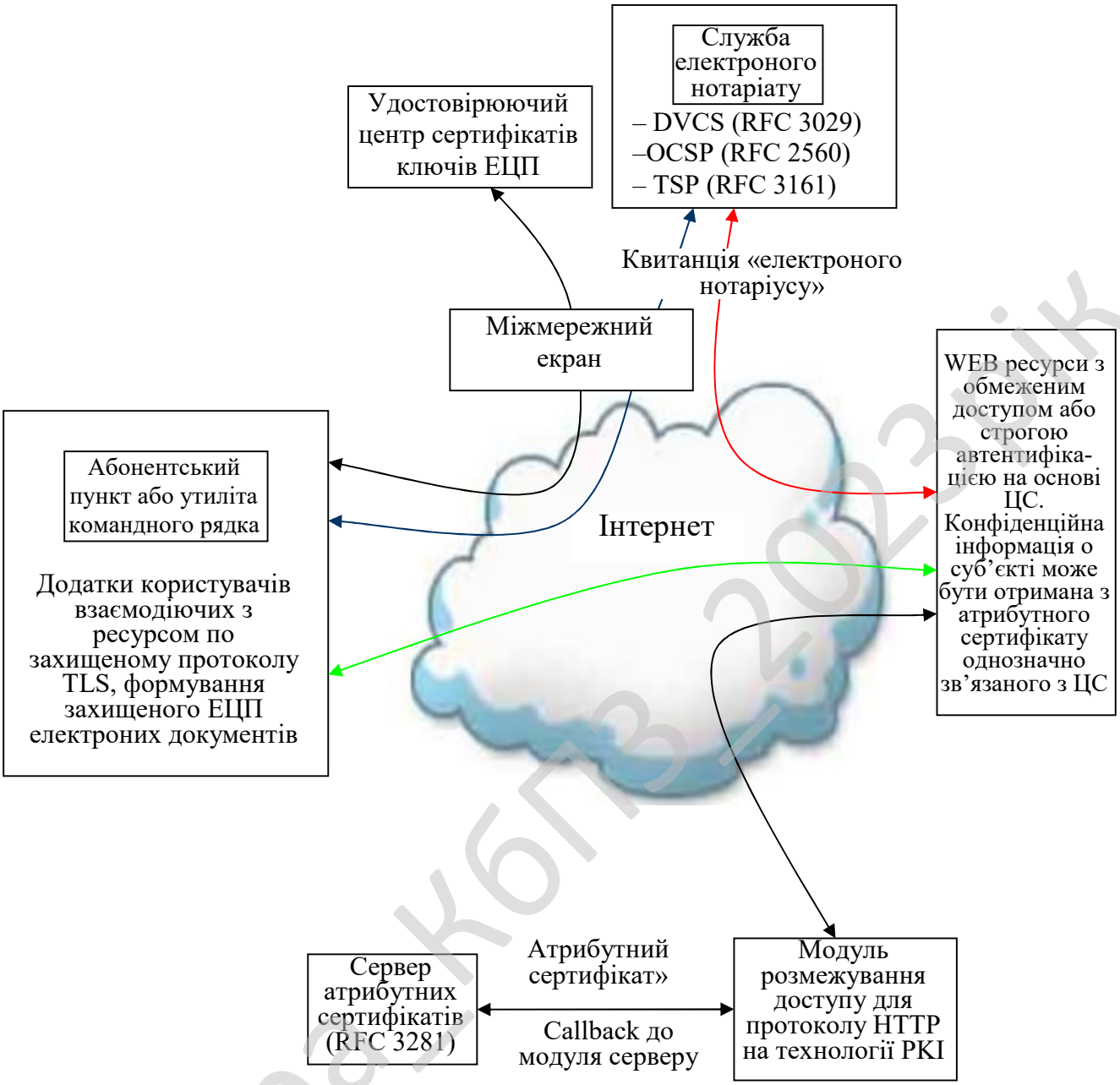

Рисунок 3.1 – Структурна схема

За допомогою розробленого програмного забезпечення легко забезпечуються вимоги по інформаційній безпеці, запропоновані різними Інтернет додатками, такими як сервера платіжних систем, інтернет-магазини, багатопрофільні корпоративні Веб-сервера, що містять інформацію з різним рівнем конфіденційності, B2B системи, системи захищеного документообігу, обміну електронною поштою й багато які інші.

На рисунку 3.1 представлена структурна схема розробленої системи. На цій схемі введені наступні позначення:

– ЕЦП – електронний цифровий підпис;

– ЦС – цифровий сертифікат;

– PKI – інфраструктура відкритих ключів;

– DVCS – Data Validation and Certification Server Protocols – протокол підтвердження даних та сертифікації серверу;

– OCSP – Online Certificate Status Protocol – онлайн протокол статусу сертфікату;

– TSP – Time-Stamp Protocol – протокол часових міток;

– TLS – криптографічний протокол, що забезпечує захищену передачу даних між вузлами в мережі хмари Інтернет;

– RFC – документ, у якому описується той або інший стандарт.

Розглянемо більш детально складові структурної схеми, та принцип роботи системи.

Служба "Електронного нотаріату"

"Електронний нотаріус" (ЕН) може бути як додатковим сервісом в комп'ютерній системі, що виконує функції Центра, що засвідчує (УЦ) сертифікатів ключів підпису в якості стандартизованогоRFC 3029 Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols (DVCS). RFC 2560 Online Certificate Status Protocol – OCSP. RFC 3161 Time-Stamp Protocol (TSP)) технічного рішення «Про Електронний цифровий підпис», так і як самостійний програмний комплекс, і реалізовувати разову або абонентську послугу з перевірки й сертифікації інформації, перевірки сертифікатів і виробленню квитанції, що містять «штамп» часу. В ЕН зведені служби, технічна реалізація яких стандартизована міжнародними рекомендаціями, по фактах усіляких перевірок, підтверджень і вироблення «штампа часу» для зовнішніх компонентів інфраструктури відкритих ключів.

"Електронний нотаріус" виконує наступні функції:

– Посвідчення факту володіння інформацією з або без її подання сервісу.

– Перевірка дійсності ЕЦП.

– Перевірка дійсності сертифіката відкритого ключа (для компонентів DVCS або OCSP).

– Вироблення квитанції, що містить «штамп» часу (TSP).

Задачі, у рішенні яких, може бути використана Служба "Електронного нотаріату":

1. Створення єдиного домена захищеного електронного документообігу, у тому числі побудованому на несумісних між собою засобах криптографічного захисту інформації, але маючих сертифікат ДСТЗІ СБУ на засоби криптографічного захисту інформації для гетерогенних програмно-апаратних платформ.

2. Одержання штампа «дійсного часу для даної PKI системи» на завіреному електронному документі. Досить важливо (для попередження шахрайських дій або колізій) при завіренні електронного документа коректно вказувати дату підписання, однак проставляння дійсної дати цілком є відповідальністю підписується стороны, що. ЕН у цьому випадку є «третьою» стороною – довіреному арбітром, що фіксує факт наявності дійсної ЕЦП на конкретний момент часу. Даний сервіс іноді називають Time Stamping. Наявність «третьої» незалежної сторони може виявитися корисною щоб зафіксувати певний етап у технологічному ланцюжку документообігу (якийсь документ пройшов яку те стадію свого формування), наприклад, на конкретний момент часу податкова декларація завірена й доставлена від платника податків в інспекцію. У більше широкому змісті ЕН може бути використаний абстракною прикладною системою як джерело TSP міток «еталонного часу».

3. Тривале архівне зберігання електронних документів. ЕЦП на електронному документі має «строк життя», що, зокрема визначається «строком життя» персонального сертифіката, який бере участь у формуванні ЕЦП. Через багато причин цей час досить обмежений, що не дозволяє будувати повноцінну

систему документообігу, включаючи такий важливий компонент як архівне зберігання. Наявність DVC квитанцій по перевірці ЕЦП дозволяє робити висновки про дійсність ЕЦП уже після витікання часу дійсності сертифіката, що брав участь у виробленні даної ЕЦП. Дана властивість пояснюється тим, що сертифікат ЕН (DVCS), яким завірена квитанція, більш тривала й існують механізми пролонгації квитанцій (випуск квитанції на квитанцію).

4. Організація перевірки ЕЦП «третьою» стороною для користувачів дозволяє перевести сам факт перевірки із площини криптографічних обчислень на сертифікованих серверах захисту інформації в площину організації довіреної доставки квитанцій із сервера ЕН, що в багатьох випадках значно технологичніше. Ступінь «доручення» доставки квитанцій не регламентується законом «Про ЕЦП» і цілком визначається специфікою комп'ютерної системи, у якій використовуються завірені документи. Способів доставки досить багато: від доставки квитанції кур'єрською службою, підтвердженням по телефоні, порівнянням самих файлів – квитанцій отриманих по мережі хмари й з репозиторія ЕН (DVCS), до організації захищених сегментів мережі хмари, на підтвердження що мають, наприклад, атестат ДСТЗІ СБУ.

5. Покладання на сервіс ЕН функцію перевірки дійсності якогось цифрового сертифіката істотно спрощує комп'ютерну систему, у якій циркулюють завірені електронні документи. Сама по собі процедура перевірки сертифіката досить трудомістка, необхідно побудувати ланцюжок перевірки кінцевого сертифіката з перевіркою всіх проміжних кореневих сертифікатів, визначити місце розповсюдження, одержати й обробити списки відкликаних сертифікатів і т.п.

6. Даний сервіс може бути досить корисний для комп'ютерних систем (КС), у яких використовується факт володіння користувачем якоюсь інформацією без її опублікування. Наприклад, КС проведення різних тендерів, регламент яких визначає, що до певного строку ніхто не повинен мати доступ до конкурсного матеріалу (за винятком коротких анотацій) і тільки по настанню часу початку

конкурсу «конверти» повинні бути розкриті. Для таких систем учасники представляють квитанції на істинність ЕЦП конкурсного матеріалу без фактичної передачі самого матеріалу до моменту настання конкурсу. Істинність представленого в наслідку матеріалу підтверджено ЕЦП зі складу DVC квитанції. У цьому випадку захист конкурсного матеріалу покладає на самих конкурсантів – самих зацікавлених у захисті даних осіб і повністю знімає ризик шахрайства в системі.

Служба атрибутування (реалізація заснована на RFC 3281) вирішує дві задачі:

– Дозволяє здійснити криптографічний зв'язок сертифіката ключа підпису з додатковою інформацією, захищеної ЕЦП, що визначає роль власника сертифіката в КС, наприклад, для цілей розмежування доступу, розміщення персональної інформації, розміщення інформації уточнюючого повноваження й т.п.

– Дозволяє здійснити криптографічний зв'язок між абстрактним блоком даних і додатковою інформацією (метаданими), наприклад, такий атрибутний контейнер можна асоціювати з електронним документом разом з метаданими при міжсистемному (міжвідомчому) інформаційному обміні. Додатково, використовувана технологія дозволяє (ЕЦП у вигляді CMS або PKCS#7, або «підпис із розширеними даними для перевірки» по ETSI TS 101 733 не може це забезпечити) ввести поняття строку дійсності документа, включаючи механізми екстреного визнання розміщеної в контейнері інформації недійсною. Дана унікальна властивість може бути використана при випуску електронних дозволів, наприклад, імпортно-карантинні дозволи, ліцензії (у тому числі й на програмне забезпечення) і т.п.

Центр, що засвідчує, сертифікатів ключів підпису

Центр, що засвідчує, сертифікатів ключів підпису (удостовірюючий центр – УЦ) є повністю вітчизняною розробкою:

– відповідає вимогам Доктрини інформаційної безпеки в плані заміщення імпортних технічних і програмних засобів у українських інформаційних системах;

– завдяки відомості всіх криптографічних процедур в ізольований криптографічний PKSC#11 токен, в УЦ можуть використовуватися як вітчизняні криптографічні алгоритми, так і закордонні;

– використовувані вітчизняні криптографічні механізми й протоколи опираються на інтернет драфти, розроблені вітчизняними компаніями, а отже, рішення сумісні з рішеннями інших вітчизняних виробників;

– технічна реалізація заснована на сучасних керівних документах, стандартах і міжнародних рекомендаціях, що дозволяє:

a) в одному технічному рішенні підтримувати невиразно велике число зовсім ізольованих, у тому числі з різними криптографічними алгоритмами, видавців;

b) підтримуються механізми кросування видавців (аж до рівня мостового УЦ), у тому числі й зовнішніх, для утворення єдиних зон обігу захищених документів;

c) для систем наближених до On-line передбачене поширення відновлень списків відкликаних сертифікатів (delta CRL);

d) компоненти самого УЦ для побудови ланцюжків сертифікації використовують внутрішній сервіс OCSP, що може бути оформлений як корпоративний зі складу служби "електронного нотаріату";

e) до складу системи входить власна служба роздачі міток часу, з механізмами підстроювання під зовнішні еталони;

f) відповідно до RFC 3039 Internet X.509 Public Key Infrastructure. Qualified Certificate Profile у сертифікат може бути введений серійний номер імені, тим самим вирішена "колізія імен" – проблема «однофамільців»;

g) реєстр крім самого сертифіката користувача може містити додаткову інформацію про суб'єкта, включаючи графічні елементи (фотографії, відбитки пальців і т.п.);

h) всі інформаційні блоки при транспортуванні й зберіганні захищені електронними цифровими підписами, що забезпечує цілісність, авторство й невідрікаємість і спрощує процедури розбору конфліктних ситуацій.

Центр, що засвідчує, сертифікатів ключів підпису (УЦ) є основою комп'ютерних систем захищеного документообігу на технології відкритого розподілу ключів (Public Key Infrastructure (PKI)). Технічна реалізація Центра, що засвідчує, відповідає вимогам Закону України "Про електронний цифровий підпис" за умови використання в ЦУ сертифікованих ДСТЗІ СБУ засобів електронного цифрового підпису. УЦ, може виступати як ключовий компонентом для різного типу прикладних захищених систем корпоративного рівня (захищений документообіг, Інтернет-банкінг, білінгові системи, електронна комерція (B2C, B2B), Інтернет процесінг і.т.п.).

Сертифікат X.509 v3

Сертифікат X.509 v3 визначається в такий спосіб. Для обчислення підпису дані, які повинні бути підписані, представляються з використанням ASN.1 однозначних правил подання (DER).

```
Certificate  ::=  SEQUENCE  {
    tbsCertificate     TBSCertificate,
    signatureAlgorithm AlgorithmIdentifier,
    signatureValue     BIT STRING
}
TBSCertificate  ::= SEQUENCE  {
    version [0] EXPLICIT Version DEFAULT v1,
    serialNumber CertificateSerialNumber,
    signature AlgorithmIdentifier,
    issuer Name,
    validity Validity,
    subject Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID [1] IMPLICIT
        UniqueIdentifier OPTIONAL,
```

```
    ---і якщо є присутнім, версія повинна
    ---і бути v2 або v3
    subjectUniqueID [2] IMPLICIT
        UniqueIdentifier OPTIONAL,
    ---і якщо є присутнім, версія повинна бути
    ---і v2 або v3
    extensions [3] EXPLICIT Extensions OPTIONAL
    ---і якщо є присутнім, версія повинна бути v3
}
Version ::= INTEGER { v1(0), v2(1), v3(2) }
CertificateSerialNumber ::= INTEGER
Validity ::= SEQUENCE {
    notBefore Time,
    notAfter  Time
}
Time ::= CHOICE {
    utcTime     UTCTime,
    generalTime GeneralizedTime
}
UniqueIdentifier ::= BIT STRING
SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm        AlgorithmIdentifier,
    subjectPublicKey BIT STRING
}
Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
Extension ::= SEQUENCE  {
      extnID    OBJECT IDENTIFIER,
      critical  BOOLEAN DEFAULT FALSE,
      extnValue OCTET STRING
}
```

Поля сертифіката. Сертифікат є послідовність трьох обов'язкових полів: tbsCertificate, signatureAlgorithm і signatureValue.

– tbsCertificate. Поле містить імена суб'єкта й випускаючого, відкритий ключ, пов'язаний із суб'єктом, період дійсності й іншу пов'язану із цим сертифікатом інформацію. Поля докладно описані далі; tbsCertificate звичайно включають розширення, які теж будуть описані нижче.

– signatureAlgorithm. Поле signatureAlgorithm містить ідентифікатор криптографічного алгоритму, використовуваного УЦ для підписування даного

сертифіката. Існують стандартні алгоритми, які повинні підтримуватися всіма реалізаціями, але конкретна реалізація може підтримувати й інші алгоритми.

Ідентифікатор алгоритму визначається наступної ASN.1-структурою:

```
AlgorithmIdentifier ::= SEQUENCE  {
  algorithm  OBJECT IDENTIFIER,
  parameters ANY DEFINED BY algorithm OPTIONAL
}
```

Ідентифікатор алгоритму використовується для визначення криптографічного алгоритму. Компонент OBJECT IDENTIFIER ідентифікує алгоритм (такий як DSA з SHA-1). Компоненти поля параметрів змінюються відповідно до зазначеного алгоритму. Поле повинне містити той же самий ідентифікатор алгоритму, що й поле підпису в tbsCertificate.

– signatureValue. Поле signatureValue містить цифровий підпис, обчислений для поля tbsCertificate, записаному в DER-поданні ASN.1. Це означає, що поле tbsCertificate, представлене як ASN.1 DER, використовується як вхід у функцію підпису. Отримане значення підпису представлене як BIT STRING і включено в поле підпису. Деталі даного процесу можуть відрізнятися для кожного конкретного алгоритму підпису. Створенням даного підпису УЦ підтверджує дійсність інформації в поле tbsCertificate. Зокрема, УЦ підтверджує зв'язок між матеріалом відкритого ключа й суб'єктом сертифіката.

– TBSCertificate. Послідовність TBSCertificate містить інформацію, пов'язану із суб'єктом сертифіката й УЦ, що випустив сертифікат. Кожний TBSCertificate містить імена суб'єкта й випускаючого, відкритий ключ, пов'язаний із суб'єктом, період дійсності, номер версії й серійний номер сертифіката; деякі поля можуть (але це не обов'язково) містити унікальний ідентифікатор. Розглянемо синтаксис і семантику таких полів. TBSCertificate звичайно включає розширення. Розглянемо також найбільше часто використовувані в Internet розширення.

– Version. Дане поле описує версію подання сертифіката. Якщо використовуються розширення, то версія повинна бути 3 (значення – 2). Якщо

розширення не зазначені, але UniqueIdentifier представлений, версія може бути 2 (значення – 1); але версія може бути й 3. Якщо представлені тільки базові поля, версія може бути 1 (значення в сертифікаті опущене як значення за замовчуванням); але версія може бути 2 або 3. Реалізації повинні бути готові приймати будь-яку версію сертифіката. Як мінімум конформні реалізації повинні розпізнавати версію 3 сертифікатів.

– Serial number. Серійний номер повинен бути позитивним цілим, призначуваним УЦ для кожного сертифіката. Він повинен бути унікальним для кожного сертифіката, випущеного даним УЦ. Таким чином, ім'я що выпустили й серійний номер однозначно визначають сертифікат. Cas повинні забезпечувати, щоб серійні номери були ненегативними цілими. Уважається, що серійні номери можуть мати довжину до 20 октетів.

– Signature. Дане поле містить ідентифікатор алгоритму, використовуваного УЦ для підписування сертифіката.

– Дане поле повинне містити той же самий ідентифікатор алгоритму, що й поле signatureAlgorithm в Certificate. Зміст необов'язкового поля параметрів залежить від конкретного алгоритму.

– Issuer. Поле issuer ідентифікує того, хто підписав і випустив сертифікат. Поле issuer повинне містити непусте унікальне ім'я (DN). Ім'я визначається у відповідності з наступної ASN.1 структурою:

```
Name ::= CHOICE { RDNSequence }
RDNSequence ::= SEQUENCE OF
    RelativeDistinguishedName
RelativeDistinguishedName ::=
    SET OF AttributeTypeAndValue
AttributeTypeAndValue ::= SEQUENCE {
    type   AttributeType,
    value  AttributeValue
}
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY DEFINED BY
    AttributeType
```

Ім'я описує ієрархічне ім'я, що складається з атрибутів, таких, наприклад, як назва країни, і відповідних значень, таких як RU. Тип компонента AttributeValue визначається значенням AttributeType. Стандарт X.509 не обмежує набір типів атрибутів, які можуть з'явитися в ім'ї. Проте, стандартом рекомендується підтримувати наступні типи атрибутів в іменах випускаючі й суб'єкта:

– Країна.

– Організація.

– Організаційна одиниця.

– Позначення унікального імені.

– Назва штату або регіону.

– Загальноприйняте ім'я (наприклад, Іванов Іван).

– Серійний номер.

Додатково можуть бути присутнім деякі інші типи атрибутів в іменах випускаючі й суб'єкта, наприклад:

– Локалізація.

– Заголовок.

– По батькові.

– Призначене ім'я.

– Ініціали.

– Псевдонім.

– Спеціальна назва (наприклад, "Jr.", " 3-ій" або "IV").

Також може бути присутнім атрибут domainComponent. DNS надає собою ієрархічну систему позначення ресурсів. Даний атрибут надає зручний механізм для організацій, які хочуть використовувати унікальні DN імена паралельно зі своїми DNS-іменами. Це не заміняє dNSName компонент альтернативного поля ім'я. Стандарт не вимагає конвертувати такі імена в DNS-імена.

Сторона, що перевіряє, повинна обробляти поля унікального ім'я випускаючого й унікального ім'я суб'єкта для одержання ланцюжка імен при перевірці дійсності сертифікаційного шляху. Ланцюжок імен виходить у випадку відповідності унікального ім'я випускаючого в першому сертифікаті ім'я суб'єкта в сертифікаті УЦ.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи.

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.

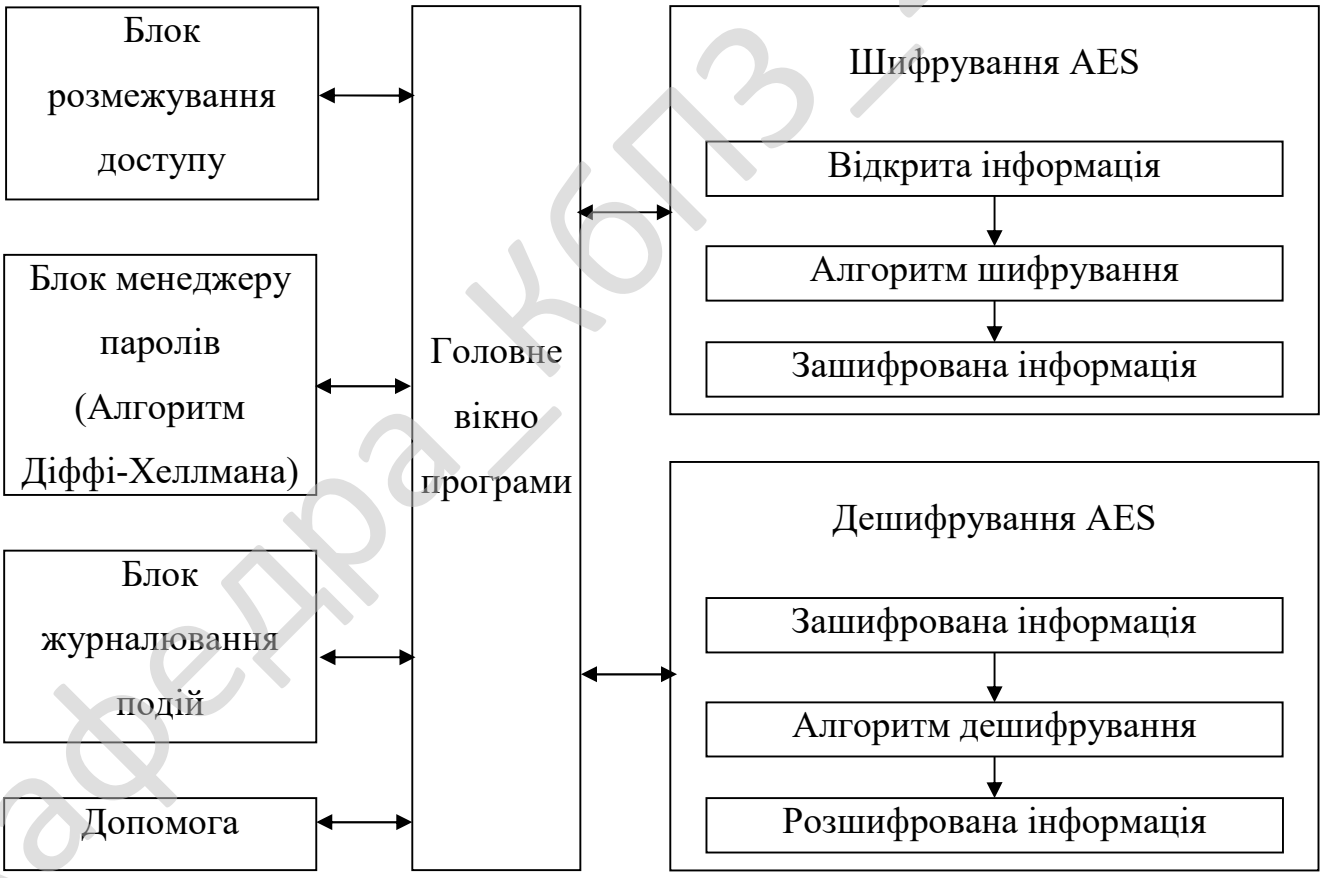

Рисунок 3.2 – Функціональна схема системи

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.

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. Призначень 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.

Блок розмежування доступу. Призначень 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.

Допомога. Блок призначень 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 AES. 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 бакалаврського проектування, наведена на рисунку 3.3. Процеси взаємодіють наступним чином. Спершу завантажується процес виведення головного вікна програми.

Він взаємодіє з процесом створення облікових записів.

Процес створення облікових записів взаємодіє з наступними процесами:

– Процес створення паролів, який, у свою чергу взаємодіє з процесом менеджеру паролів.

– Процес створення ключів та сертифікатів.

– Процес розмежування прав доступу.

Процес створення ключів та сертифікатів взаємодіє з наступними процесами:

– Процес управління ключами.

– Процес управління сертифікатами.

Процес розмежування прав доступу взаємодіє з процесом роботи з конфіденційними даними.

Процес роботи з конфіденційними даними взаємодіє з наступними процесами:

– Процес шифрування даних.

– Процес дешифрування даних.

Процес шифрування даних взаємодіє з процесом запису зашифрованих даних.

Процес дешифрування даних взаємодіє з процесом читання дешифрованих даних.

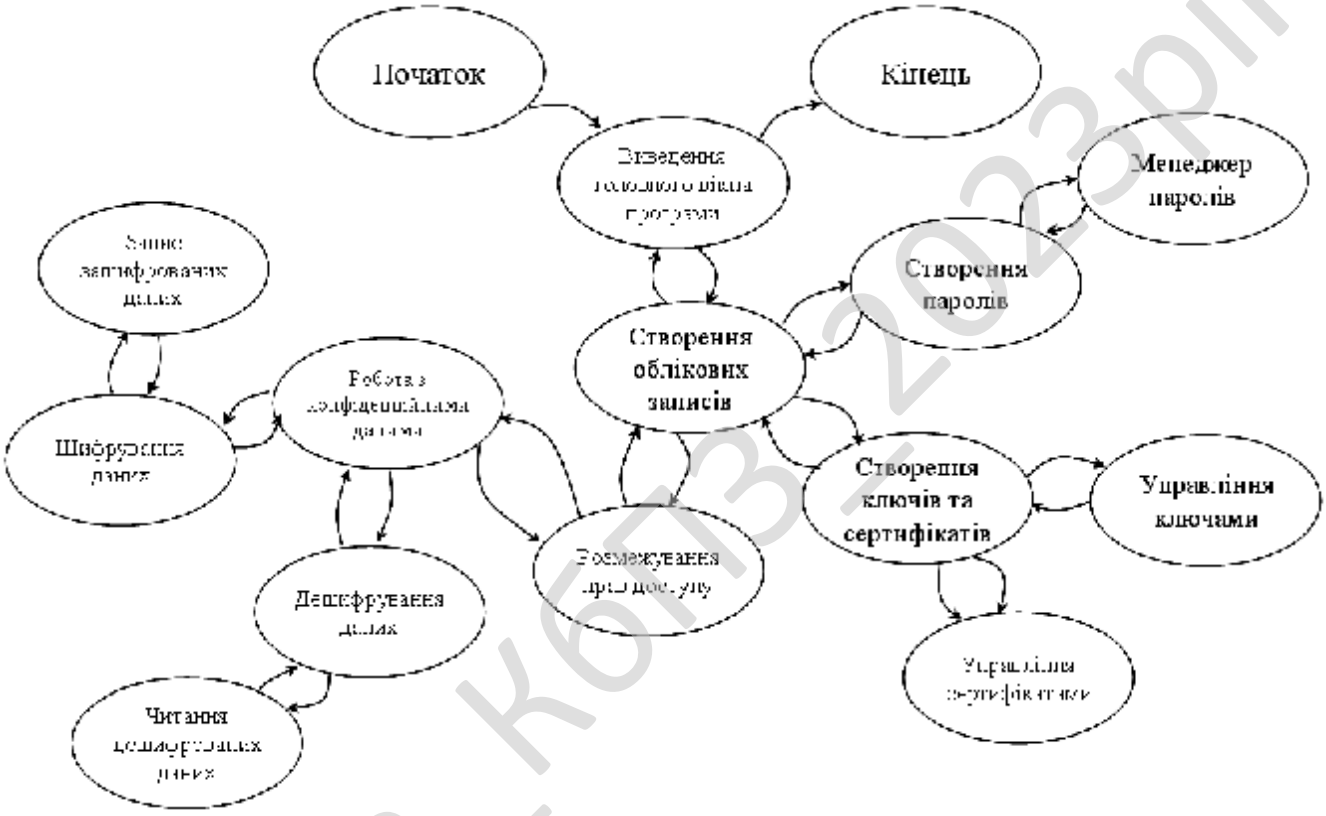

Рисунок 3.3 – Діаграма процесів системи

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 відбувається виведення основного вікна програми. Після цього користувач обирає одну з доступних дій:

– Створення нового облікового запису.

– Запис конфіденційних даних.

– Читання конфіденційних даних.

Якщо користувач обирає створення нового облікового запису, тоді виконуються наступні дії:

– Створення облікового запису та додавання його у БД.

– Створення паролів.

– Створення сертифікатів та ключів.

Якщо користувач обирає запис конфіденційних даних, тоді виконуються наступні дії:

– Введення даних.

– Введення паролю та ключа.

– Шифрування даних алгоритмом AES.

– Запис зашифрованих даних.

Якщо користувач обирає читання конфіденційних даних, тоді виконуються наступні дії:

– Введення паролю та ключа.

– Дешифрування даних алгоритмом AES.

– Виведення дешифрованих даних.

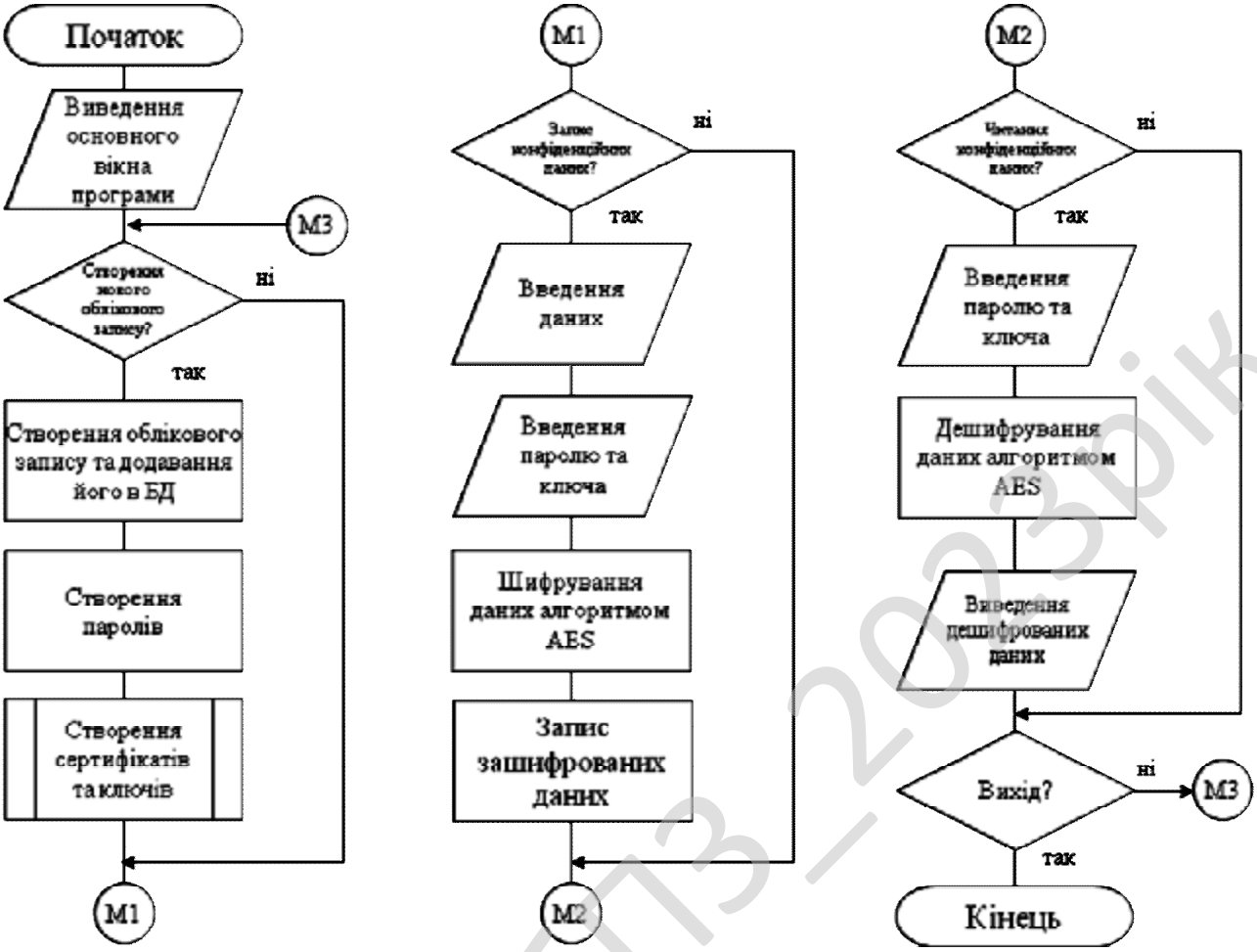

Рисунок 4.1 – Блок-схема роботи основної програми

На рисунку 4.2 зображено блок-схему алгоритму роботи підпрограми створення сертифікатів та ключів.

Спершу відбувається виведення діалогового вікна створення ключів та сертифікатів.

Після цього відбувається введення параметрів з’єднання з сервером.

Якщо потрібно здійснити з’єднання з сервером, тоді виконується підключення до сервера.

Якщо потрібно створити сертифікат, тоді виконуються наступні дії

– Генерація запиту на одержання сертифікату.

– Відправка запиту на одержання сертифікату.

– Отримання сертифікату та закритого ключа.

– Збереження сертифікату у базі даних.

– Збереження закритого ключа у окремому файлі.

– Шифрування файлу закритим ключем.

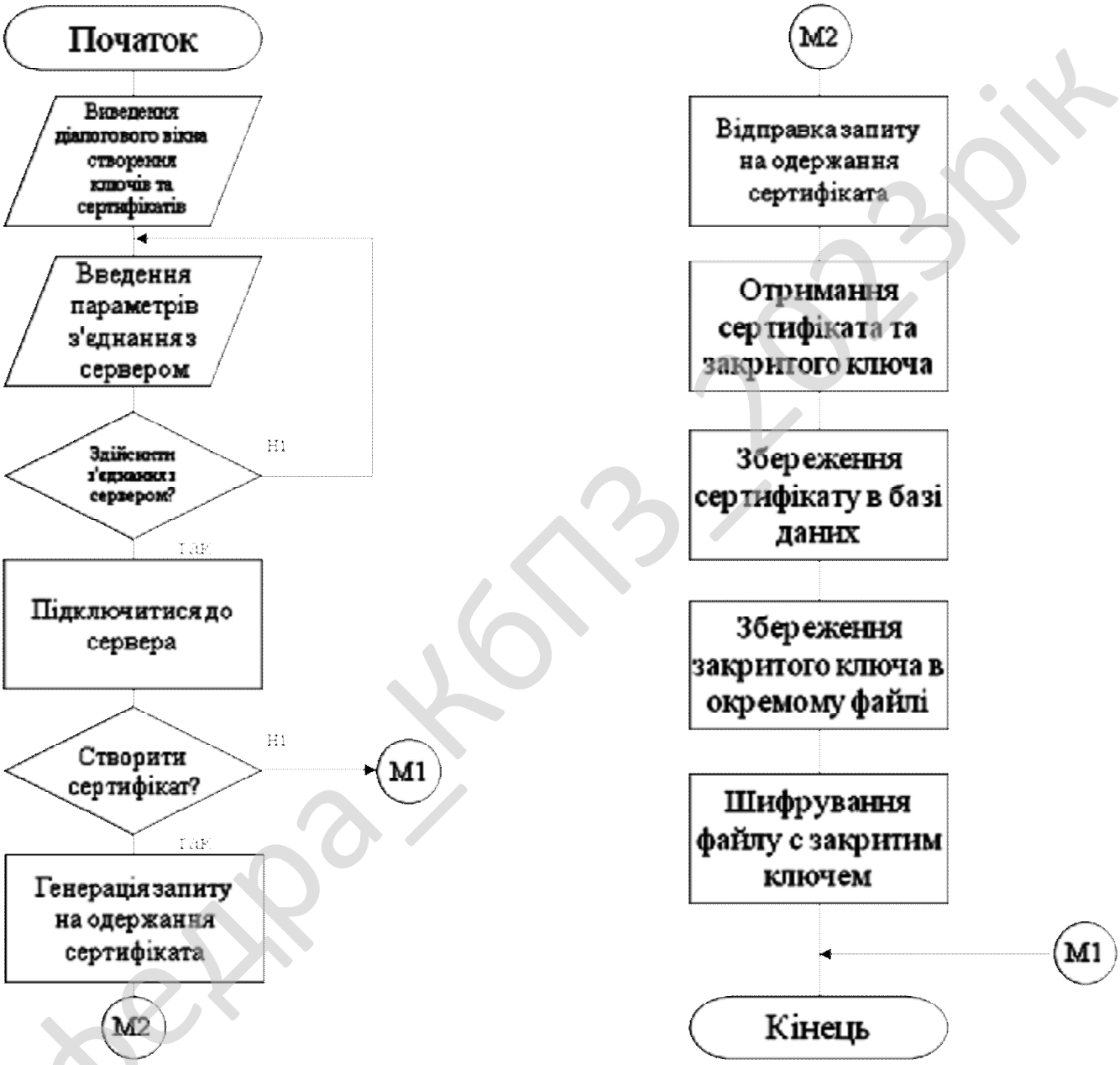

Рисунок 4.2 – Блок-схема алгоритму роботи підпрограми створення сертифікатів та ключів

Розглянемо, як програмно реалізуються деякі функції, які описані у блок-схемах.

Простір імен System.Security.Cryptography надає криптографічні служби, що включають безпечне кодування й декодування даних, а також цілий ряд інших функцій, таких як хешування, генерація випадкових чисел і перевірка дійсності повідомлень.

Класи:

– Aes – Абстрактний базовий клас, від якого повинні успадковуватися всі реалізації стандарту AES (Advanced Encryption Standard).

– AesCryptoServiceProvider – Виконує симетричне шифрування й дешифрування за допомогою реалізації CAPI алгоритму симетричного шифрування AES.

– AesManaged – Надає керовану реалізацію алгоритму симетричного шифрування AES.

– AsnEncodedData – Представляє дані в кодуванні ASN.1.

– AsnEncodedDataCollection – Представляє колекцію об'єктів AsnEncodedData. Цей клас не успадковується.

– AsnEncodedDataEnumerator – Надає можливість переміщення по об'єкті AsnEncodedDataCollection. Цей клас не успадковується.

– AsymmetricAlgorithm – Представляє абстрактний базовий клас, від якого успадковуються всі реалізації алгоритмів асиметричного шифрування.

– AsymmetricKeyExchangeDeformatter – Представляє базовий клас, з якого створюються всі об'єкти, що забезпечують дешифрування даних обміну ключами за допомогою алгоритму асиметричного шифрування.

– AsymmetricKeyExchangeFormatter – Представляє базовий клас, з якого створюються всі об'єкти, що забезпечують зашифрування даних обміну ключами за допомогою алгоритму асиметричного шифрування.

– AsymmetricSignatureDeformatter – Представляє абстрактний базовий клас, з якого створюються всі реалізації класів перевірки підпису.

– AsymmetricSignatureFormatter – Базовий клас, від якого успадковуються всі реалізації асиметричних об'єктів створення сигнатур.

– CngAlgorithm – Інкапсулює ім'я алгоритму шифрування.

– CngAlgorithmGroup – Інкапсулює ім'я групи алгоритмів шифрування.

– CngKey – Визначає базову функціональність для ключів, використовуваних об'єктами криптографії наступного покоління (CNG).

– CngKeyBlobFormat – Задає формат великого двійкового об'єкта ключа для використання з об'єктами Microsoft CNG.

– CngKeyCreationParameters – Містить додаткові властивості для створення ключа.

– CngPropertyCollection – Надає строго типізований набір властивостей CNG.

– CngProvider – Інкапсулює ім'я постачальника сховища ключів (KSP), використовуване з об'єктами криптографії наступного покоління (CNG).

– CngUIPolicy – Інкапсулює додаткові параметри конфігурації для користувальницького інтерфейсу, відображуваного механізмом CNG при доступі до захищеного ключа.

– CryptoAPITransform – Виконує криптографічне перетворення даних. Цей клас не успадковується.

– CryptoConfig – Звертається до відомостей про криптографічну конфігурацію.

– CryptographicAttributeObject – Містить тип і колекцію значень, пов'язаних із цим типом.

– CryptographicAttributeObjectCollection – Містить набір об'єктів CryptographicAttributeObject.

– CryptographicAttributeObjectEnumerator – Надає функціональність перерахування для колекції CryptographicAttributeObjectCollection. Цей клас не успадковується.

– CryptographicException – Виключення, що виникає у випадку помилки при виконанні криптографічної операції.

– CryptographicUnexpectedOperationException – Виключення виникає при виконанні непередбаченої операції під час криптографічної операції.

– CryptoStream – Визначає потік, що зв'язує потоки даних із криптографічними перетвореннями.

– CspKeyContainerInfo – Надає додаткові відомості про пару криптографічних ключів. Цей клас не успадковується.

– CspParameters – Містить параметри, передані постачальникові служб шифрування (CSP), що виконує криптографічні обчислення. Цей клас не успадковується.

– DeriveBytes – Абстрактний базовий клас, від якого успадковуються всі класи, що одержують послідовності байтів заданої довжини.

– DES – Представляє базовий клас для алгоритму DES, з якого створюються всі реалізації DES.

– DESCryptoServiceProvider – Визначає об'єкт-оболонку для доступу до версії алгоритму DES, надаваної постачальником служб шифрування (CSP). Цей клас не успадковується.

– DSA – Представляє абстрактний базовий клас, від якого успадковуються всі реалізації алгоритму DSA.

– DSACryptoServiceProvider – Визначає об'єкт-оболонку для доступу до реалізації алгоритму DSA, надаваної постачальником служб шифрування (CSP).Цей клас не успадковується.

– DSASignatureDeformatter – Перевіряє підпис, створений по алгоритму PKCS 1 v1.5 DSA (DSA).

– DSASignatureFormatter – Створює підпис DSA.

– ECDiffieHellman – Надає абстрактний базовий клас, з якого створюються реалізації алгоритму Діффі-Хеллмана на еліптичних кривих (ECDH). Цей клас надає базовий набір операцій, які повинні підтримувати всі реалізації алгоритму ECDH.

– ECDiffieHellmanCng – Надає реалізацію CNG алгоритму Діффі-Хеллмана на еліптичних кривих (ECDH). Цей клас використовується для виконання криптографічних операцій.

– ECDiffieHellmanCngPublicKey – Задає відкритий ключ алгоритму Діффі-Хеллмана на еліптичних кривих (ECDH) для використання із класом ECDiffieHellmanCng.

– ECDiffieHellmanPublicKey – Абстрактний базовий клас, від якого повинні успадковуватися всі реалізації ECDiffieHellmanCngPublicKey.

– ECDsa – Надає абстрактний базовий клас, інкапсулюючий алгоритм цифрового підпису DSA на еліптичних кривих (Elliptic Curve Digital Signature Algorithm, ECDSA).

– ECDsaCng – Забезпечує реалізацію CNG алгоритму цифрового підпису DSA на еліптичних кривих (Elliptic Curve Digital Signature Algorithm, ECDSA).

– FromBase64Transform – Перетворить потік CryptoStream з кодування base64.

– HashAlgorithm – Представляє базовий клас, з якого створюються всі реалізації криптографічних хеш-алгоритмів.

– HMAC – Абстрактний клас, від якого повинні успадковуватися всі реалізації хеш-коду перевірки дійсності повідомлення (HMAC).

– HMACMD5 – Обчислює хеш-код перевірки дійсності повідомлення (HMAC) за допомогою хеш-функції MD5.

– HMACRIPEMD160 – Обчислює хеш-код перевірки дійсності повідомлення (HMAC) за допомогою хеш-функції RIPEMD160.

– HMACSHA1 – Обчислює хеш-код перевірки дійсності повідомлення (HMAC) за допомогою хеш-функції SHA1.

– HMACSHA256 – Обчислює хеш-код перевірки дійсності повідомлення (HMAC) за допомогою хеш-функції SHA256.

– HMACSHA384 – Обчислює хеш-код перевірки дійсності повідомлення (HMAC) за допомогою хеш-функції SHA384.

– HMACSHA512 – Обчислює хеш-код перевірки дійсності повідомлення (HMAC) за допомогою хеш-функції SHA512.

– KeyedHashAlgorithm – Представляє абстрактний клас, з якого створюються всі реалізації хеш-алгоритмів із ключем.

– KeySizes – Визначає набір припустимих розмірів ключа для симетричних алгоритмів шифрування.

– MACTripleDES – Обчислює код перевірки дійсності повідомлення (MAC) за допомогою алгоритму TripleDES для вхідних даних CryptoStream.

– ManifestSignatureInformation – Надає інформацію про підпис маніфесту.

– ManifestSignatureInformationCollection – Представляє колекцію об'єктів ManifestSignatureInformation, доступну тільки для читання.

– MaskGenerationMethod – Абстрактний клас, від якого повинні успадковуватися всі алгоритми створення масок.

– MD5 – Представляє абстрактний клас, від якого успадковуються всі реалізації хеш-алгоритму MD5.

– MD5Cng – Надає реалізацію CNG алгоритму MD5 (Message Digest 5) для формування 128-розрядних хеш-значень.

– MD5CryptoServiceProvider – Обчислює значення хеша MD5 для вхідних даних за допомогою реалізації, надаваної постачальником служб шифрування (CSP).Цей клас не успадковується.

– Oid – Представляє ідентифікатор криптографічного об'єкта. Цей клас не успадковується.

– OidCollection – Представляє колекцію об'єктів Oid. Цей клас не успадковується.

– OidEnumerator – Надає можливість переміщення по об'єкті OidCollection. Цей клас не успадковується.

– PasswordDeriveBytes – Формує ключ із пароля за допомогою розширення алгоритму PBKDF1.

– PKCS1MaskGenerationMethod – Обчислює маски, що відповідають стандарту PKCS #1, для використання алгоритмами обміну ключами.

– ProtectedData – Надає методи для шифрування й розшифровки даних. Цей клас не успадковується.

– ProtectedMemory – Надає методи для захисту даних, що зберігаються в оперативній пам'яті, і зняття захисту з них. Цей клас не успадковується.

– RandomNumberGenerator – Представляє абстрактний клас, з якого створюються всі реалізації криптографічних генераторів випадкових чисел.

– RC2 – Представляє базовий клас, з якого створюються всі реалізації алгоритму RC2.

– RC2CryptoServiceProvider – Визначає об'єкт-оболонку для доступу до реалізації алгоритму RC2, надаваної постачальником служб шифрування (CSP).Цей клас не успадковується.

– Rfc2898DeriveBytes – Реалізує функцію формування ключа на основі пароля (PBKDF2) з використанням генератора псевдовипадкових чисел HMACSHA1.

– Rijndael – Представляє базовий клас, від якого успадковуються всі реалізації алгоритму симетричного шифрування Rijndael.

– RijndaelManaged – Здійснює доступ до керованої версії алгоритму Rijndael. Цей клас не успадковується.

– RijndaelManagedTransform – Виконує криптографічне перетворення даних за допомогою алгоритму Rijndael. Цей клас не успадковується.

– RIPEMD160 – Представляє абстрактний клас, від якого успадковуються всі реалізації хеш-алгоритму MD160.

– RIPEMD160Managed – Обчислює хеш RIPEMD160 для вхідних даних за допомогою керованої бібліотеки.

– RNGCryptoServiceProvider – Реалізує криптографічний генератор випадкових чисел, використовуючи реалізацію, надавану постачальником служб шифрування (CSP). Цей клас не успадковується.

– RSA – Базовий клас, від якого успадковуються всі реалізації алгоритму RSA.

– RSACryptoServiceProvider – Виконує зашифрування й дешифрування за допомогою реалізації асиметричного алгоритму RSA, надаваного постачальником служб шифрування (CSP). Цей клас не успадковується.

– RSAOAEPKeyExchangeDeformatter – Виконує дешифрування даних обміну ключами, зашифрованих з використанням заповнення OAEP.

– RSAOAEPKeyExchangeFormatter – Створює дані обміну ключами, використовуючи заповнення OAEP і алгоритм RSA.

– RSAPKCS1KeyExchangeDeformatter – Виконує дешифрування даних обміну ключами по стандарті PKCS 1.

– RSAPKCS1KeyExchangeFormatter – Створює дані обміну ключами по стандарті PKCS 1 за допомогою RSA.

– RSAPKCS1SignatureDeformatter – Перевіряє підпис RSA PKCS 1 версії 1.5.

– RSAPKCS1SignatureFormatter – Створює підпис RSA PKCS 1 версії 1.5.

– SHA1 – Обчислює хеш SHA1 для вхідних даних.

– SHA1Cng – Надає реалізацію CNG алгоритму SHA.

– SHA1CryptoServiceProvider – Обчислює значення хеша SHA1 для вхідних даних за допомогою реалізації, надаваної постачальником служб шифрування (CSP). Цей клас не успадковується.

– SHA1Managed – Обчислює хеш SHA1 для вхідних даних за допомогою керованої бібліотеки.

– SHA256 – Обчислює хеш SHA256 для вхідних даних.

– SHA256Cng – Надає реалізацію CNG алгоритму SHA для формування 256-розрядних хеш-значень.

– SHA256CryptoServiceProvider – Визначає об'єкт-оболонку для доступу до реалізації алгоритму SHA256, надаваної постачальником служб шифрування (CSP).

– SHA256Managed – Обчислює хеш SHA256 для вхідних даних за допомогою керованої бібліотеки.

– SHA384 – Обчислює хеш SHA384 для вхідних даних.

– SHA384Cng – Надає реалізацію CNG алгоритму SHA для формування 384-розрядних хеш-значень.

– SHA384CryptoServiceProvider – Визначає об'єкт-оболонку для доступу до реалізації алгоритму SHA384, надаваної постачальником служб шифрування (CSP).

– SHA384Managed – Обчислює хеш SHA384 для вхідних даних за допомогою керованої бібліотеки.

– SHA512 – Обчислює хеш SHA512 для вхідних даних.

– SHA512Cng – Надає реалізацію CNG алгоритму SHA для формування 512-розрядних значень хеша.

– SHA512CryptoServiceProvider – Визначає об'єкт-оболонку для доступу до реалізації алгоритму SHA512, надаваної постачальником служб шифрування (CSP).

– SHA512Managed – Обчислює хеш SHA512 для вхідних даних за допомогою керованої бібліотеки.

– SignatureDescription – Містить відомості про властивості цифрового підпису.

– StrongNameSignatureInformation – Містить відомості про підпис строгого ім'я для маніфесту.

– SymmetricAlgorithm – Представляє абстрактний базовий клас, від якого успадковуються всі реалізації алгоритмів симетричного шифрування.

– ToBase64Transform – Перетворить CryptoStream у кодування base64.

– TripleDES – Представляє базовий клас для алгоритмів Triple DES, з якого створюються всі реалізації TripleDES.

– TripleDESCryptoServiceProvider – Визначає об'єкт-оболонку для доступу до реалізації алгоритму TripleDES, надаваної постачальником служб шифрування (CSP). Цей клас не успадковується.

Структури:

– CngProperty – Інкапсулює властивість ключа або постачальника CNG.

– DSAParameters – Містить типові параметри для алгоритму DSA.

– RSAParameters – Представляє стандартні параметри для алгоритму RSA.

Інтерфейси:

– ICryptoTransform – Визначає основні операції криптографічних перетворень.

– ICspAsymmetricAlgorithm – Визначає методи, що дозволяють класу AsymmetricAlgorithm одержувати інформацію про контейнер ключів, а також імпортувати й експортувати більші двійкові об'єкти ключа, сумісні з інтерфейсом Microsoft Cryptographic API (CAPI).

Перерахування:

– CipherMode – Задає режим блокового шифру для використання при шифруванні.

– CngExportPolicies – Задає політики експорту для ключа.

– CngKeyCreationOptions – Задає параметри створення ключа.

– CngKeyHandleOpenOptions – Задає параметри відкриття дескрипторів ключа.

– CngKeyOpenOptions – Задає параметри відкриття ключа.

– CngKeyUsages – Задає криптографічні операції, у яких може використовуватися ключ CNG.

– CngPropertyOptions – Задає параметри властивостей ключа CNG.

– CngUIProtectionLevels – Задає рівень захисту ключа в скриптах запиту користувальницького інтерфейсу.

– CryptoStreamMode – Задає режим доступу до криптографічного потоку.

– CspProviderFlags – Задає прапори, які змінюють режим роботи постачальників служб шифрування.

– DataProtectionScope – Задає область захисту даних, що встановлюється за допомогою методу Protect.

– ECDiffieHellmanKeyDerivationFunction – Задає функцію формування ключа, використовувану класом ECDiffieHellmanCng для перетворення секретної угоди в ключовий матеріал.

– ECKeyXmlFormat – Визначає формати серіалізації ключів на еліптичних кривих в XML.

– FromBase64TransformMode – Визначає, потрібно чи ігнорувати порожній простір при перетворенні в кодування base64.

– KeyNumber – Задає тип ключа для створення – асиметричний ключ підпису або асиметричний ключ обміну.

– MemoryProtectionScope – Задає область захисту пам'яті, що встановлюється за допомогою методу Protect.

– PaddingMode – Задає тип заповнення, використовуваного у випадку, коли блок дані повідомлення коротше повного числа байтів, необхідного для криптографічної операції.

– SignatureVerificationResult – Задає більшість кодів результату для перевірки підпису.

Надамо реалізацію CNG алгоритму Діффі-Хеллмана на еліптичних кривих (ECDH). Цей клас використовується для виконання криптографічних операцій.

Ієрархія спадкування

System.Object

System.Security.Cryptography.AsymmetricAlgorithm

System.Security.Cryptography.ECDiffieHellman

System.Security.Cryptography.ECDiffieHellmanCng

Простір імен: System.Security.Cryptography

Складання: System.Core (в System.Core.dll)

[HostProtectionAttribute(SecurityAction.LinkDemand, MayLeakOnAbort = true)]

public sealed class ECDiffieHellmanCng : ECDiffieHellman

Тип ECDiffieHellmanCng надає наступні члени.

Конструктори:

– ECDiffieHellmanCng – Ініціалізує новий екземпляр класу ECDiffieHellmanCng парою випадкових ключів.

– ECDiffieHellmanCng(CngKey) – Ініціалізує новий екземпляр класу ECDiffieHellmanCng, використовуючи зазначень об'єкт CngKey.

– ECDiffieHellmanCng(Int32) – Ініціалізує новий екземпляр класу ECDiffieHellmanCng парою випадкових ключів, використовуючи заданий розмір ключа.

Властивості:

– HashAlgorithm – Одержує або задає хеш-алгоритм, використовуваний при генерації ключового матеріалу.

– HmacKey – Одержує або задає ключ HMAC, використовуваний при формуванні ключового матеріалу.

– Key – Задає об'єкт CngKey, що буде використовуватися поточним об'єктом для виконання криптографічних операцій.

– KeyDerivationFunction – Одержує або задає функцію формування ключа для класу ECDiffieHellmanCng.

– KeyExchangeAlgorithm – Одержує ім'я алгоритму обміну ключами. (Успадковано від ECDiffieHellman.)

– KeySize – Одержує або задає розмір модуля ключа (у бітах), використовуваного алгоритмом асиметричного шифрування. (Успадковано від AsymmetricAlgorithm.)

– Label – Одержує або задає значення мітки, використовуване для формування ключа.

– LegalKeySizes – Одержує розміри ключа, які підтримуються алгоритмом асиметричного шифрування. (Успадковано від AsymmetricAlgorithm.)

– PublicKey – Одержує відкритий ключ, що може використовуватися іншим об'єктом ECDiffieHellmanCng для генерації секретної угоди. (Перевизначає ECDiffieHellman.PublicKey.)

– SecretAppend – Одержує або задає значення, що додається до кінця секретної угоди при генерації ключового матеріалу.

– SecretPrepend – Одержує або задає значення, що додається до початку секретної угоди при формуванні ключового матеріалу.

– Seed – Одержує або задає початкове значення, використовуване при формуванні ключового матеріалу.

– SignatureAlgorithm – Одержує ім'я алгоритму підпису. (Успадковано від ECDiffieHellman.)

– UseSecretAgreementAsHmacKey – Одержує значення, що визначає, чи використовується секретна угода як ключ HMAC для формування ключового матеріалу.

Методи:

– Clear – Звільняє всі ресурси, використовувані об'єктом AsymmetricAlgorithm. (Успадковано від AsymmetricAlgorithm.)

– DeriveKeyMaterial(CngKey) – Формує ключовий матеріал із секретної угоди, укладеного між двома сторонами, використовуючи заданий об'єкт CngKey, у якому втримується відкритий ключ другої сторони.

– DeriveKeyMaterial(ECDiffieHellmanPublicKey) – Формує ключовий матеріал із секретної угоди, укладеного між двома сторонами, використовуючи заданий об'єкт ECDiffieHellmanPublicKey, у якому втримується відкритий ключ другої сторони. (Перевизначає ECDiffieHellman.DeriveKeyMaterial(ECDiffieHellmanPublicKey).)

– DeriveSecretAgreementHandle(CngKey) – Одержує дескриптор секретної угоди, укладеного між двома сторонами, використовуючи заданий об'єкт CngKey, у якому втримується відкритий ключ другої сторони.

– DeriveSecretAgreementHandle(ECDiffieHellmanPublicKey) – Одержує дескриптор секретної угоди, погодженого між двома сторонами, використовуючи заданий об'єкт ECDiffieHellmanPublicKey, у якому втримується відкритий ключ другої сторони.

– Dispose – Звільняє всі ресурси, використовувані поточним екземпляром класу AsymmetricAlgorithm. (Успадковано від AsymmetricAlgorithm.)

– Dispose(Boolean) – Звільняє некеровані ресурси, використовувані класом AsymmetricAlgorithm (при необхідності звільняє й керовані ресурси). (Успадковано від AsymmetricAlgorithm.)

– Equals(Object) – Визначає, чи рівний заданий об'єкт Object поточному об'єкту Object. (Успадковано від Object.)

– Finalize – Дозволяє об'єкту спробувати звільнити ресурси й виконати інші операції очищення, перед тим як об'єкт буде утилізований у процесі складання сміття. (Успадковано від Object.)

– FromXmlString(String) – Цей метод не реалізований. (Перевизначає AsymmetricAlgorithm.FromXmlString(String).)

– FromXmlString(String, ECKeyXmlFormat) – Виконує десеріалізацію даних ключа з XML-Рядка, використовуючи зазначень формат.

– GetHashCode – Відіграє роль хеш-функції для певного типу. (Успадковано від Object.)

– GetType – Повертає об'єкт Type для поточного екземпляра. (Успадковано від Object.)

– MemberwiseClone – Створює неповну копію поточного об'єкта Object. (Успадковано від Object.)

– ToString – Повернення рядка, що представляє поточний об'єкт. (Успадковано від Object.)

– ToXmlString(Boolean) – Цей метод не реалізований. (Перевизначає AsymmetricAlgorithm.ToXmlString(Boolean).)

– ToXmlString(ECKeyXmlFormat) – Виконує серіалізацію даних ключа в XML-рядок, використовуючи зазначень формат.

Поля:

– KeySizeValue – Представляє розмір модуля ключа (у бітах), використовуваного алгоритмом асиметричного шифрування. (Успадковано від AsymmetricAlgorithm.)

– LegalKeySizesValue – Задає розміри ключа, які підтримуються алгоритмом асиметричного шифрування. (Успадковано від AsymmetricAlgorithm.)

Клас ECDiffieHellmanCng дозволяє двом сторонам обмінюватися матеріалом закритих ключів, навіть якщо взаємодія здійснюється по відкритому каналі. Обидві сторони можуть обчислити те саме таємне значення, що називається секретна угода в керованих класах Діффі-Хелмана. Секретна угода може потім використовуватися для різних цілей, у тому числі як симетричний ключ. Однак, замість прямого подання таємної угоди клас ECDiffieHellmanCng клас робить деяку обробку угоди перед наданням значення. Ця постобработка називається функцією формування ключа (key derivation function, KDF); можна вибрати, яку функція формування ключа використовувати, і задати її параметри через набір властивостей в екземплярі об'єкта Діффі-Хеллмана.

Hash:

– HashAlgorithm – алгоритм хеширования, використовуваний для обробки секретної угоди.

– SecretPrepend – додатковий байтовий масив для вставки в секретну угоду перед хешуванням.

– SecretAppend – додатковий байтовий масив для прикріплення секретної угоди перед хешуванням.

HMac:

– HashAlgorithm – алгоритм хеширования, використовуваний для обробки секретної угоди.

– SecretPrepend – додатковий байтовий масив для вставки в секретну угоду перед хешуванням.

– SecretAppend – додатковий байтовий масив для прикріплення секретної угоди перед хешуванням.

TLS:

– Label – мітка для формування ключа.

– Seed – початкове значення для формування ключа.

Результат передачі секретної угоди через функцію створення ключа являє собою масив байтів, якому можна використовувати як матеріал ключа для відповідного додатка. Кількість байтів створеного матеріалу ключа залежить від функції формування ключа; наприклад SHA-256 створить 256 бітів матеріалу ключа, а SHA-512 – 512 бітів матеріалу ключа. Обмін ключами ECDH відбувається в такий спосіб:

1. А й Б створюють пару ключів для операції обміну ключами Діффі-Хелмана.

2. А й Б набудовують функцію формування ключа з використанням застережених параметрів.

3. А відправляє свій відкритий ключ Б.

4. Б відправляє свій відкритий ключ А.

5. А й Б використовують відкриті ключі один одного для створення секретної угоди й застосовують функцію формування ключа для створення матеріалу ключа.

Застосований до даного типу або члена атрибут HostProtectionAttribute має наступне значення властивості Resources: MayLeakOnAbort. Атрибут HostProtectionAttribute не робить впливу на настільні додатки (запускаються звичайно подвійним клацанням значка, уведенням команди або URL-адреси в оглядачі).

У наступному прикладі показане використання класу ECDiffieHellmanCng для встановлення обміну ключами, а також використання цього ключа для шифрування повідомлення, яке можна відправляти по відкритому каналу й розшифровувати одержувачем.

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Text;
class Alice
```

```
        {
            public static byte[] alicePublicKey;
            public static void Main(string[] args)
            {
                using (ECDiffieHellmanCng alice = new ECDiffieHellmanCng())
                {
                    alice.KeyDerivationFunction =
ECDiffieHellmanKeyDerivationFunction.Hash;
                    alice.HashAlgorithm = CngAlgorithm.Sha256;
                    alicePublicKey = alice.PublicKey.ToByteArray();
                    Bob bob = new Bob();
                    CngKey k = CngKey.Import(bob.bobPublicKey,
CngKeyBlobFormat.EccPublicBlob);
                    byte[] aliceKey =
alice.DeriveKeyMaterial(CngKey.Import(bob.bobPublicKey,
CngKeyBlobFormat.EccPublicBlob));
                    byte[] encryptedMessage = null;
                    byte[] iv = null;
                    Send(aliceKey, "Секретне повідомлення", out encryptedMessage,
out iv);
                    bob.Receive(encryptedMessage, iv);
                }
            }
            private static void Send(byte[] key, string secretMessage, out byte[]
encryptedMessage, out byte[] iv)
            {
                using (Aes aes = new AesCryptoServiceProvider())
                {
                    aes.Key = key;
                    iv = aes.IV;
                    // Шифрування повідомлення
                    using (MemoryStream ciphertext = new MemoryStream())
                    using (CryptoStream cs = new CryptoStream(ciphertext,
aes.CreateEncryptor(), CryptoStreamMode.Write))
                    {
                        byte[] plaintextMessage =
Encoding.UTF8.GetBytes(secretMessage);
                        cs.Write(plaintextMessage, 0, plaintextMessage.Length);
                        cs.Close();
                        encryptedMessage = ciphertext.ToArray();
                    }
                }
            }
```

```
        }
        public class Bob
        {
            public byte[] bobPublicKey;
            private byte[] bobKey;
            public Bob()
            {
                using (ECDiffieHellmanCng bob = new ECDiffieHellmanCng())
                {
                    bob.KeyDerivationFunction =
ECDiffieHellmanKeyDerivationFunction.Hash;
                    bob.HashAlgorithm = CngAlgorithm.Sha256;
                    bobPublicKey = bob.PublicKey.ToByteArray();
                    bobKey =
bob.DeriveKeyMaterial(CngKey.Import(Alice.alicePublicKey,
CngKeyBlobFormat.EccPublicBlob));
                }
            }
            public void Receive(byte[] encryptedMessage, byte[] iv)
            {
                using (Aes aes = new AesCryptoServiceProvider())
                {
                    aes.Key = bobKey;
                    aes.IV = iv;
                    // Дешифрування повідомлення
                    using (MemoryStream plaintext = new MemoryStream())
                    {
                        using (CryptoStream cs = new CryptoStream(plaintext,
aes.CreateDecryptor(), CryptoStreamMode.Write))
                        {
                            cs.Write(encryptedMessage, 0, encryptedMessage.Length);
                            cs.Close();
                            string message =
Encoding.UTF8.GetString(plaintext.ToArray());
                            Console.WriteLine(message);
                        }
                    }
                }
            }
        }
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Khufu. Khufu – це 64-бітовий блоковий шифр. 64-бітовий відкритий тест спочатку розщеплюється на дві 32-бітові половини, *L* і *R.* Над обома половинами й певними частинами ключа виконується операція XOR. Потім, аналогічно DES, результати проходять деяку послідовність раундів. У кожному раунді молодший значущий байт *L* використовується як вхід S-блоку. У кожного S-блоку 8 вхідних біт і 32 вихідних біта. Далі обраний в S-блоці 32-бітовий елемент піддається операції XOR з *R.* Потім *L* циклічно зрушується на число, кратним восьми биткам, *L* і *R* міняються місцями, і раунд завершується. Сам S-блок не статичний, він міняється кожні вісім раундів. Нарешті, по закінченні останнього раунду, над *L* і *R* виконується операція XOR з іншими частинами ключа, і половини поєднуються, утворюючи блок шифртексту.

Хоча частини ключа використовуються для операції XOR із блоком шифрування на початку й кінці виконання алгоритму, головне призначення ключа – генерація S-блоків. Ці S-блоки секретні, по суті, це частина ключа. Повний розмір ключа алгоритму Khufu дорівнює 512 біт (64 байт), алгоритм надає спосіб генерації S-блоків по ключу.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс головного вікна програми.

Він складається з наступних блоків:

– Блок меню.

– Вікно вибору файлу для шифрування/дешифрування.

– Вікна виведення змісту зашифрованого та дешифрованого файлів.

– Кнопки швидкого доступу до елементів програми.

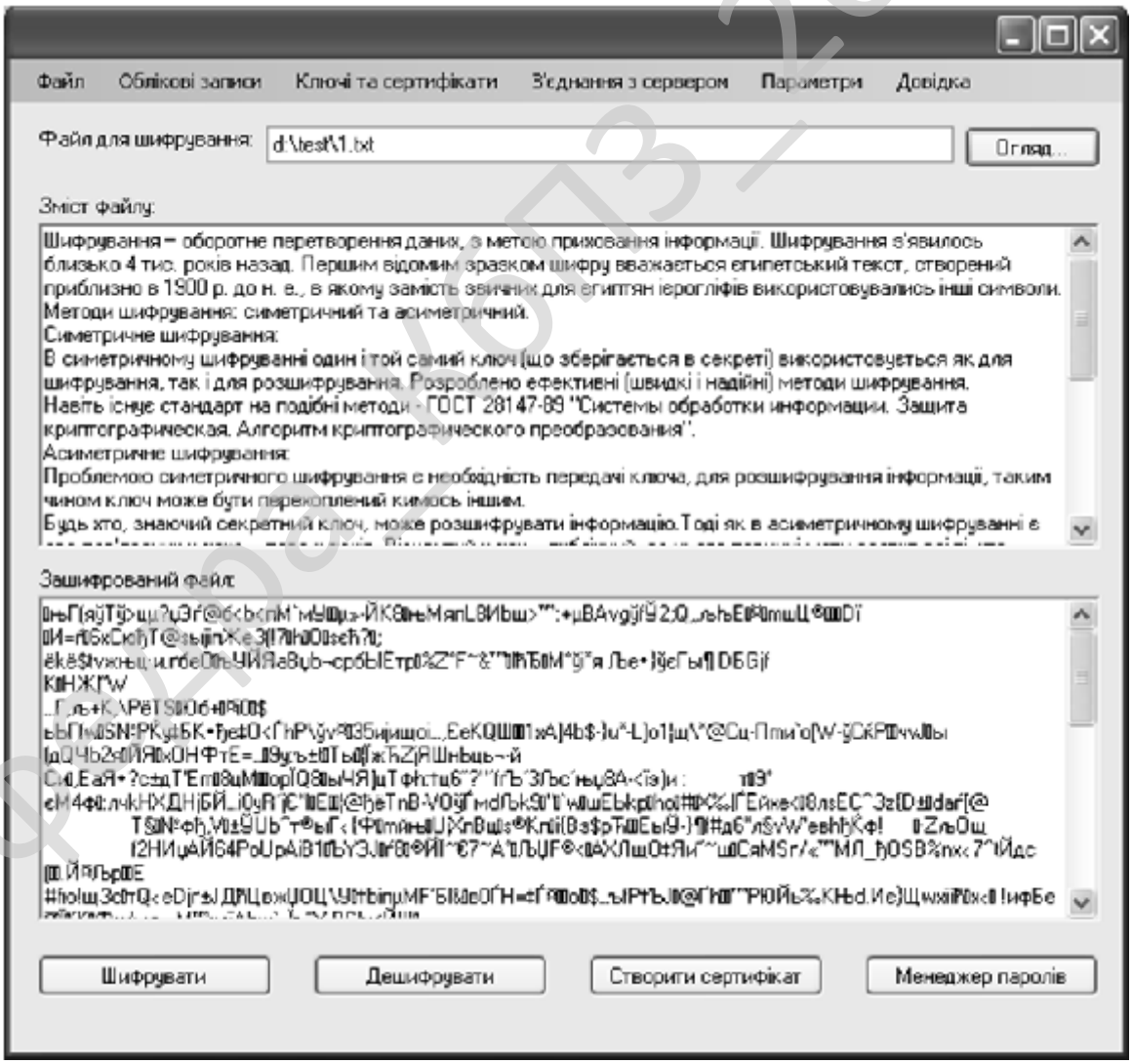

Рисунок 5.1 – Головне вікно програми

Блок меню складається з наступних елементів:

– Файл.

– Облікові записи.

– Ключі та сертифікати.

– З’єднання з сервером.

– Параметри.

– Довідка.

Реалізовані наступні кнопки швидкого доступу до елементів програми:

– Шифрувати.

– Дешифрувати.

– Створити сертифікат.

– Менеджер паролів.

На рисунку 5.2 зображено вікно довідки, у якій наведено, хто виконував бакалаврське проектування, тема бакалаврського проектування, керівник та місце виконання.

Рисунок 5.2 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для захисту інформації у хмарі з розмежуванням доступу.

– Досліджена система для захисту інформації у хмарі з розмежуванням доступу.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для захисту інформації у хмарі з розмежуванням доступу.

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 призначені для системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу.

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 кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Khufu.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 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).**

2. 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**).

3. 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).**

4. 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).**

5. 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).**

6. 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).**

7. 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).**

8. 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)**.

9. 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)**.

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

11. 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)**.

12. 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)**.

13. 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)**.

14. 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)**.

15. 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)**.

16. 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)**.

17. 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)**.

18. 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)**.

19. 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)**.

20. 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)**.

21. 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)**.

22. 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)**.

23. 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)**.

24. 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)**.

25. 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)**.

26. 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).

27. 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).

28. 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).

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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

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

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 13-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) 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 кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу;

– 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 забезпеченням.

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 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 87 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 8.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнова Т.В.

Програмне забезпечення системи кібербезпеки для захисту інформації у хмарі з розмежуванням доступу

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 18

Літера: РП

Кропивницький – 2023 року

Файл Program.cs - файл проекту

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

namespace My_AES
{
    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.Designer.cs - інтерфейс користувача

```
namespace My_AES
{
    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 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 My_AES
{
    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 _AESCryptoServiceProvider об’єкт
                // для генерування вектора ініціалізації (IV).
                _AESCryptoServiceProvider tDESalg = new
_AESCryptoServiceProvider();

                // 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 _AESCryptoServiceProvider об’єкт
                // для генерування вектора ініціалізації (IV).
                _AESCryptoServiceProvider tDESalg = new
_AESCryptoServiceProvider();

                // 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 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 My_AES
{
    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 _AESCryptoServiceProvider().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 _AESCryptoServiceProvider().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("My_AES")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("My_AES")]
[assembly: AssemblyCopyright("Copyright ©  2012")]
[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 My_AES.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("My_AES.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 My_AES.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[] {
            "БАКАЛАВРСЬКИЙ ПРОЕКТ",
            "",
            "На тему:",
            "",
            "Програмне забезпечення системи кібербезпеки для захисту інформації
у хмарі з розмежуванням доступу",
            "",
            "",
            "Керівник: Смірнова Т.В.",
            "",
            "Розробив: студент Ковальов Євген Валерійович",
            "                  гр. КБ-20-3СК                                    ",
            "",
            "м. Кропивницький 2023"});
            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;

    }
}
```