ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки,
молоді та спорту України
29.03.2012 № 384

Форма № Н-9.02

Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)
Навчально-науковий інститут інформаційних технологій і робототехніки
(повна назва факультету)
Кафедра комп'ютерних та інформаційних технологій і систем
(повна назва кафедри)

Пояснювальна записка
до дипломного проекту (роботи)

_____магістра_____
(освітньо-кваліфікаційний рівень)
на тему
_____Програмна реалізація дешифратора на основі алгоритму ММВ_____
_____із імплементацією модуля штучного інтелекту_____
_____

Виконав: студент 6 курсу, групи 601-ТН
напряму
____122 Комп'ютерні науки____
(шифр і назва напряму)
_____Походун В.Р_____
(прізвище та ініціали)
Керівник _____Головко Г.В_____
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Полтава – 2021 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І РОБОТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Спеціальність 122 «Комп'ютерні науки»

на тему

«Програмна реалізація дешифратора на основі алгоритму ММВ із імплементацією модуля штучного інтелекту»

Студента групи 601-ТН Походуна Валентина Руслановича

Керівник роботи
кандидат технічних наук,
доцент Головко Г.В.

Завідувач кафедри
кандидаттехнічних наук,
доцент Головко Г.В.

Полтава – 2021

РЕФЕРАТ

Пояснювальна записка містить: 87сторінок, 16 малюнків, 2 додатки, 31 джерело.

Об’єкт дослідження: розробка та аналіз систем комплексного захисту інформації на підприємстві ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ.

Мета роботи: проаналізувати стан та розробити рекомендації щодо систем комплексного захисту інформації на підприємстві ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ.

Методи: Аналіз стану та розробка комплексного захисту інформації на підприємстві ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ.

Ключові слова: захист інформації, інформаційна безпека, безпека інформації, криптографія, комплексні методи захисту інформації, методи захисту інформації, штучний інтелект, генетичний алгоритм.

ANNOTATION

Explanatorynote contains: 87pages, 16 figures, 2 appendices, 31sources.

Object of research: development and analysis of complex information protection systems for the enterprise ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ.

Thepurposeofthework: analyze current state and develop recommendations for integrated information security systems at the enterprise ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ.

Methods: analysis of the state and development of comprehensive information protection at the enterprise ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ.

Keywords: information protection, information security, cryptography, complex methods of information protection, methods of information protection, artificial intelligence, genetic algorithm.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІБ – Інформаційна безпека.

ЗІ – Захист інформації.

СЗІ – Системи захисту інформації.

ІС– Інформаційна система.

ДТ – Державна таємниця.

СУБД – Система управління базами даних.

ПЗ – Програмне забезпечення.

C# (C Sharp) – об’єктно орієнтована мова програмування.

ММВ – Модульний блочний шифр на основі множення.

ПК – персональний комп’ютер.

ШІ – штучний інтелект.

ГА – генетичний алгоритм.

ВСТУП

Інформація є активом, котра, як і інші важливі бізнес активи, становить цінність для організацій та підприємств, отже, потребує відповідного захисту.

Окрім того, що інформацію можуть створювати, зберігати, змінювати, передавати, використовувати вона також може бути: ушкодженою, втраченою, викраденою або знищеною. Задля запобігання несанкціонованих операцій над інформацією дедалі компанії впроваджують методи захисту інформації.

Актуальністю теми дипломної є забезпечення інформаційної безпеки підприємства ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ з використанням криптографічних методів захисту інформації в програмно-технічних засобах, а також вдосконалення самих методів криптографії.

Метою дипломної роботи є дослідження та аналіз технологій безпеки інформаційних мереж для отримання вимог та рекомендацій до вибору та застосування заходів і технологій ІБ на підприємстві, а також вивчення механізмів та алгоритмів для захисту персональних даних з подальшою розробкою програмного продукту, що надає зручний інтерфейс для надійного шифрування конфіденційної інформації.

При аналізі проблематики, пов'язаної з інформаційною безпекою, необхідно враховувати специфіку даного аспекту безпеки полягає в тому, що інформаційна безпека є складова частина інформаційних технологій – області, що розвивається безпрецедентно високими темпами. Тут важливі не стільки окремі рішення (закони, навчальні курси, програмно-технічні засоби), що знаходяться на сучасному рівні, скільки механізми генерації нових рішень, що дозволяють жити в темпі технічного прогресу[1].

Збільшення числа атак – ще не найбільша неприємність. Гірше те, що постійно виявляються нові вразливі місця в програмному забезпеченні і, як наслідок, виникають нові види атак.

В таких умовах системи інформаційної безпеки повинні вміти протистояти різноманітним атакам, як зовнішнім, так і внутрішнім, атакам автоматизованим і скоординованим. Іноді напад триває частки секунди; інколи знаходження вразливих місць ведеться повільно і продовжується годинами, так що підозріла активність практично непомітна. Метою зловмисників можливо порушення всіх складових ІБ – доступності, цілісності або конфіденційності.

Незалежний дослідник в галузі інформаційної безпеки Ponemon Institute спільно з IBM Security опитав 383 компанії з 12 країн і з'ясував, що за останній рік розмір збитку від витоків конфіденційних даних зросла з $ 3,8 млн до $ 4 млн.

Офіційні звіти повідомляють, що середній збиток від втрати кожного окремого запису виріс з $ 154 до $ 158. Експерти також встановили, що використання засобів інформаційного захисту та швидка реакція на інциденти допомагає компаніям знизити витрати на ліквідацію наслідків. Так, якщо витоку стоденної давності обходяться компаніям приблизно в $ 4,3 млн, то більш раннє виявлення мінімізує витрати до $ 3,23 млн.[2].

У першому півріччі 2017 року світі стався витік 7,78 млрд записів з персональної і платіжною інформацією. Про це йдеться в звіті компанії InfoWatch, що спеціалізується на інформаційній безпеці (документ є у РБК). Цей показник майже у вісім разів вище, ніж за аналогічний період 2016 роки (1,06 млрд записів), і більш ніж удвічі перевищує кількість даних, які потрапили в руки третіх осіб за весь 2016 рік (3 млрд записів).

Велика частина витоку даних (98%) сталася в результаті великих інцидентів, які InfoWatch називає «мегавитоками». Всього в компанії зафіксували 20 подібних випадків, коли в розпорядженні третіх осіб виявлялося більше 10 млн записів конфіденційних даних.

«Загальносвітовий тренд на збільшення числа витоків і обсягів скомпрометованих даних, на нашу думку, задають не особливості окремих регіонів, а нові можливості, які пов'язані з використанням інформації в

цифровому світі, такі як переклад послуг в електронний вигляд, електронні гроші, об'єкти виняткових прав (інтелектуальна власність) в цифровому вигляді», – йдеться в звіті InfoWatch.

Всього за перше півріччя 2017 року відбулася 925 інцидентів, пов'язаних з втратою конфіденційної інформації, що на 10% більше, ніж за той же період 2016 року. Через зовнішнього втручання, наприклад з вини хакерів, відбулися 384 таких інциденту. Через внутрішні порушень (наприклад, з вини співробітників компаній, випадково або навмисно оприлюднили конфіденційну інформацію) – 520 витоків. Причини залишилася 21 витоку з'ясувати не вдалося.

Обсяг скомпрометованих даних в результаті атак ззовні виявився значніше (5,23 млрд записів), ніж через внутрішні порушень (2,32 млрд). У той же час, на думку InfoWatch, звіт охоплює не більше 1% всіх випадків витоку даних, так як заснований на аналізі публічної інформації, а випадки компрометації конфіденційних даних часто ховаються компаніями.

Збільшення кількості атак в останні два роки, є показником того що ставлення до максимального забезпечення безпеки інформації є неналежним, тому що більшість атак було здійснено через те, що зловмисники використали вади операційних систем застарілих версій. Використані недоліки були присутні лише на версіях ОС які були розроблені в кращому випадку 10 років тому, в останніх версіях ОС цього виробника дані недоліки були відсутні. Постраждали компаній та організації всіх рівнів, від аеропортів та банківських систем та Національної системи охорони здоров’я Великобританії, до малих компаній. Збитки за різними даними були більше ніж $1 млрд.[3].

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Поняття захисту інформації

1.1.1 Визначення захисту інформації та інформаційної безпеки.Захист інформації– сукупність методів та заходів, які впливають на збереження конфіденційності, цілісності у обміні та взаємодії з інформацією. Інформаційний захист і підтримка його належної інфраструктури для запобігання від випадкових або зловмисних дій, які можуть нанести збитки суб'єктам обміну інформацією (їх власникам і користувачам інфраструктури).

Якість комерційної інформації забезпечує необхідний економічний ефект для компанії, тому важливо охороняти критично важливі дані від неправомірних дій. Це дозволить компанії успішно конкурувати на ринку.

Інформаційна безпека – це стан, який визначає захищеність інформації у системах де вона зберігається, при якому головним є впровадження одного або комплексу методів задля забезпечення доступності, конфіденційності, цілісності. Слід враховувати, що інформаційна безпека є частиною інформаційних технологій галузі, що розвивається надто високими темпами. Важливі не стільки окремі рішення (закони, державні стандарти, програмні засоби), котрі знаходяться на високому сучасному технологічному рівні, скільки механізми генерації нових рішень та усвідомлення важливості їх впровадження на національному рівні, окремих галузей, корпоративних або персональних рівнів.

Безпека інформації, яка обробляється в організації, – це комплекс дій, спрямованих на вирішення проблеми захисту інформаційного середовища в рамках компанії. При цьому інформація не повинна бути обмежена у використанні і динамічний розвиток для уповноважених осіб.

Захист інформаційних ресурсів повинен бути:

- постійним – зловмисник в будь-який момент може спробувати обійти модулі захисту даних, які його цікавлять;
- цільовим – інформація повинна захищатися в рамках певної мети, яку ставить організація або власник даних;
- плановим – всі методи захисту повинні відповідати державним стандартам, законам і підзаконним актам, які регулюють питання захисту конфіденційних даних;
- активним – заходи для підтримки роботи та вдосконалення системи захисту повинні проводитися регулярно;
- комплексним – використання тільки окремих модулів захисту або технічних засобів неприпустимо. Необхідно застосовувати всі види захисту в повній мірі, інакше розроблена система буде позбавлена сенсу і економічного підґрунтя;
- універсальним –засоби захисту повинні бути обрані відповідно до існуючих в компанії каналами витоку;
- надійним – всі прийоми захисту повинні надійно перекривати можливі шляхи до охоронюваної інформації з боку зловмисника, незалежно від форми представлення даних.

Інформація вважається захищеною, якщо дотримуються три головних властивості[1].

Перша – цілісність – передбачає забезпечення достовірності і коректного відображення даних, що охороняються, незалежно від того, які системи безпеки і прийоми захисту використовуються в компанії. Обробка даних не повинна порушуватися, а користувачі системи, які працюють з захищеними файлами, не повинні стикатися з несанкціонованою модифікацією або знищенням ресурсів, збоями в роботі програмного забезпечення.

Друга – конфіденційність – означає, що доступ до перегляду та редагування даних надається виключно авторизованим користувачам системи захисту.

Третя – доступність – має на увазі, що всі авторизовані користувачі повинні мати доступ до конфіденційної інформації.

Досить порушити одне з властивостей захищеної інформації, щоб використання система стало безглуздим.

Виділяють наступні комплексні методи захисту інформації:

- фізичні засоби;
- апаратні засоби;
- програмні засоби;
- криптографічний та організаційні методи.

Більшість методів спрямована на взаємодію з комп’ютерними засобами управління інформацією тому, що на сьогоднішній день, майже, всі організації мають мережу комп’ютерів, за допомогою якої здійснюють більшість операцій з інформацією.

Фізичні засоби захисту – це засоби, необхідні для зовнішнього захисту засобів обчислювальної техніки, території та об'єктів. Вони реалізуються на базі ЕОМ, які спеціально призначені для створення фізичних перешкод на можливих шляхах проникнення і несанкціонованого доступу до компонентів інформаційних систем, що захищаються.

Апаратні засоби захисту – це різні електронні, електронно-механічні та інші пристрої, які вмонтовуються в серійні блоки електронних систем обробки і передачі даних для внутрішнього захисту засобів обчислювальної техніки: терміналів, пристроїв введення та виведення даних, процесорів, ліній зв'язку тощо.

Програмні засоби захисту, які вмонтовані до складу програмного забезпечення системи, необхідні для виконання логічних та інтелектуальних функцій захисту.

Апаратно-програмні засоби захисту – це засоби, які основані на синтезі програмних та апаратних засобів.

Криптографічні засоби захисту – це засоби, які за допомогою програмного забезпечення або приладів що здійснюють шифрування або

передають інформацію у зашифрованому вигляді, щоб навіть при втраті зловмисник не зміг використати інформацію у своїх цілях.

Організаційні заходи захисту інформації складають сукупність заходів щодо підбору, перевірки та навчання персоналу, який бере участь у всіх стадіях інформаційного процесу.

Якщо у роботі підприємства хоча б одна із цих складових порушена, то підприємство матиме збитки, зниження продуктивності праці або нестиме репутаційні втрати[4].

Положення про забезпечення захисту інформації у Україні визначені на законодавчому рівні низкою Законів України, постанов Кабінету міністрів України, нормативних документів у галузі технічного захисту інформації:

- Закон України «Про інформацію» від 02.10.1992 №2657-XII;
- Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» від 05.07.1994 № 80/94-ВР;
- Закон України «Про державну таємницю» від 21.01.1994 № 3855-XII;
- Закон України «Про захист персональних даних» від 01.06.2010 № 2297-VI;
- Постанова Кабінету міністрів України «Про затвердження Правил забезпечення захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах» від 29.03.2006 №373;
- Постанова Кабінету міністрів України «Про затвердження Інструкції про порядок обліку, зберігання і використання документів, справ, видань та інших матеріальних носіїв інформації, які містять службову інформацію» від 27 листопада 1998 р. №1893;
- НД ТЗІ 3.7-003-05;
- ДСТУ 3396.1-96;
- НД ТЗІ 1.4-001-2000;
- НД ТЗІ 2.5-004-99;

- НД ТЗІ 2.5-005-99;
- НД ТЗІ 2.5-008-02;
- НД ТЗІ 2.5-010-03;
- НД ТЗІ 3.7-001-99;
- НД ТЗІ 3.6-001-2000;
- НД ТЗІ 1.1-002-99[5].

Належний захист інформації одна з головних задач сучасних компаній, які вирішують свою потребу у захисті кваліфікованим ІТ відділом. Між тим важливим є не тільки запобігання втрати або витоку інформації, зниження і протистояння кількості атак на їх ресурси та атак комп'ютерних вірусів.

Найти загальне рішення одночасно і назавжди неможливо, у цьому питанні потрібно постійно оновлювати методи захисту інформації та використовувати нові розробки та рішення у захисті інформації. Тому що з розвитком методів захисту інформації, постійне зростає кількість зловмисників, вдосконалюються методи що вони використовують, збільшуються обчислювальних можливості їх апаратних засобів.

Крім інформації що зберігається на електронних носіях. Для комерційних організацій, що мають різні відділи, велику територію слід використовувати фізичні методи захисту, рівні доступи та розмежування доступу для різних посад, відділів тощо.

1.1.2 Фактори, що обумовлюють впровадження методів захисту інформації.Основними факторами, які впливають на впровадження захисту інформації, є потреби підприємств та користувачів, а також наявність відповідних засобів для їх формування. Потреби підприємства поділяються на:

Розробку правового забезпечення захисту інформації. Фактично це система нормативно-правових документів, актуальних для діяльності підприємства. З її допомогою, з одного боку, визначаються правила забезпечення інформаційної безпеки на підприємстві (наприклад, обов'язки

співробітників), а з іншого – встановлюється відповідальність за їх порушення. До складу правового забезпечення включаються державні закони та акти (наприклад, закон про державну таємницю), внутрішні нормативні та організаційні документи підприємства.

Визначення потенційних загроз безпеки інформації. Їх можна розділити на три групи – це загрози, що виникають:

- внаслідок дій людини – це можуть бути як випадкові помилки фахівців підприємства при роботі з інформаційною системою (неправильне введення даних або їх видалення), так і навмисні дії (крадіжка документів або носіїв інформації);
- внаслідок некоректної роботи або відмови технічних або програмних засобів (наприклад, збій в роботі операційної системи, викликаний вірусом);
- через стихійних лих, природних катаклізмів, форс-мажорних обставин (повені, пожежі, смерчі, військові дії тощо).

Список потенційних загроз для інформаційної безпеки підприємства може бути дуже великий. Рекомендується оцінити кожну з них з позиції здорового глузду або даних статистики, а потім виконати ранжування за ступенем ймовірності виникнення та обсягу потенційного збитку.

Складання переліку даних, що підлягають захисту. Інформація, яка використовується на підприємстві, може бути відкритою (доступна для всіх) або закритою (доступна для обмеженого кола осіб). До першого типу відносяться відомості, що не становлять державної або комерційної таємниці, не належать до категорії конфіденційної інформації (згідно із законодавством або внутрішніми документами підприємства). Збиток від втрати подібного роду відомостей не є значним, тому їх захист не пріоритетна.

До другого типу відносяться:

- дані, що становлять державну таємницю – їх перелік визначається законодавством;

- комерційні або службові відомості – будь-яка інформація, пов'язана з виробництвом, фінансами, що використовуються технологіями, витік або втрата якої може завдати шкоди інтересам підприємства;
- персональні дані співробітників.

Ця інформація повинна бути захищена в першу чергу. Для кожного типу такого роду даних вказується, як і де вони виникають, за допомогою яких програмних або технічних засобів ведеться їх обробка, які підрозділи (співробітники) з ними працюють і т.д.

Створення підрозділу, відповідального за питання захисту інформації. Як правило, на російських підприємствах існує поділ функцій, пов'язаних із забезпеченням інформаційної безпеки. Це має на увазі, що за розробку політики захисту даних, виконання організаційних заходів відповідає служба безпеки компанії, а питання, пов'язані із застосуванням будь-яких програмних і апаратних засобів, включаються до компетенції IT-підрозділу. Нерідко виникають ситуації, коли прагнення якомога надійніше захистити дані вступає в протиріччя з потребами бізнесу підприємства. У тому числі це відбувається, якщо заходи безпеки розробляються без урахування можливостей сучасних IT-засобів.

Правильним підходом є створення єдиної точки прийняття рішень, а саме створення підрозділу, завданням якого буде вирішення всього спектру питань щодо захисту інформації на підприємстві. До його складу необхідно включити як співробітників служби безпеки, так і IT-фахівців.

Визначення основних напрямків забезпечення інформаційної безпеки. В рамках вирішення цього завдання, зокрема, позначаються компоненти АСУ, які потребують захисту, визначаються необхідні програмні та технічні засоби, формуються організаційні заходи, спрямовані на захист інформації.

Таким чином, можна сказати, що в разі нехтуванням методами інформації компанії будуть нести великих економічних та репутаційних втрат. В іншому випадку цим можуть скористатися конкуренти і зловмисники[1].

1.1.3 Сучасний стан розвитку систем захисту інформації. Будь-яка локальна мережа є окремим випадком розподіленої системи. Головними відмінними рисами є кількість об'єднаних мережею автоматизованих робочих місць і їх територіальне розташування. Найчастіше, розподілені системи включають в себе значну кількість комп'ютерів, а також, окремі її сегменти можуть перебувати на великій відстані один від одного. Основним вразливим місцем даної технології є канал передачі інформації між сегментами. Оскільки, у міру переміщення по загальним комунікаційним лініям, дані можуть бути перехоплені, що веде до порушення мінімум одного з трьох основних властивостей інформації – конфіденційності (інформація стає відомою третім особам), а також, можливо, і інших: цілісності (дані можуть бути модифіковані) і доступності (дані можуть не дійти до адресата або дійти з затримкою). Крім того, існує можливість інформаційних атак на систему котра знаходиться під захистом ззовні, через глобальну мережу. Вони можуть здійснюватися за різними сценаріями і різними методами. Варто виділити наступні: вірусні атаки різних видів, мережеві атаки спрямовані на проникнення і мережеві атаки спрямовані на відмову роботи всієї системи, її структурних елементів або конкретного обладнання. Для створення системи захисту від вищеназваних загроз необхідно застосування комплексу програмних, апаратних або програмно-апаратних засобів захисту. Так, одним з ефективних інструментів для захисту каналу передачі інформації є створення і використання віртуальної приватної мережі (VPN – virtual private network), через яку проходить мережа загального доступу і шифрування даних. Для захисту від мережевих і вірусних атак застосовуються системи антивірусного захисту, системи виявлення та запобігання вторгнень і міжмережеві екрани.

Також для підвищення захищеності, інформація може бути попередньо зашифрована окремими засобами. Обидві ці функції об'єднують в собі програмні і апаратні комплекси, звані криптопровайдерами.

Системами виявлення та запобігання вторгнень є апаратні і програмні засоби, що служать для виявлення атак на мережеві вузли розподіленої системи і протидії їм. Принцип роботи даних систем базується на аналізі всього трафіку, що проходить як в одну, так і в іншу сторону. Найчастіше застосовуються технології сигнатурного аналізу та метод заснований на виявленні аномалій. Найбільш поширеними засобами захисту в даному сегменті є: «Cisco» «IPS Sensor» серії 4500 і «Network Security Platform» від «McAfee».

Програми антивірусного захисту застосовуються для запобігання і ліквідації наслідків зараження комп'ютерів шкідливим ПЗ. Головною негативною рисою шкідливого ПО є велика динаміка змін форми та синтаксису тіла вірусу, що дозволяє їм в деяких випадках обходити детектори антивірусних програм. Блок-аналізатор може використовувати метод статистичного, евристичного аналізу або їх комбінацію для підвищення ефективності. Також одним з важливих чинників застосування антивірусів полягає в підтримці вірусних баз в актуальному стані і їх регулярне оновлення.

Найбільш популярними являються: Norton 360, NOD 32, Avast, Dr. Web Security, 360 Security, Avira, MS Essential.

З вищесказаного можна зробити висновок, що сучасні методи захисту інформацію базуються на наступних методах та принципах:

Системний підхід до побудови систем захисту, який обумовлює оптимальне поєднання програмних, організаційних, фізичних і апаратних засобів, які застосовують на всіх етапах обробки інформації[6].

Принцип постійного вдосконалення системи, сучасний захист інформації передбачає постійне вдосконалення системи у відповідності зі збільшенням ризиків витоку інформації. Даний процес безперервний і полягає в реалізації сучасних методів і шляхів вдосконалення систем інформаційної безпеки, постійному контролюванні, виявленні її вразливих

місць і потенційних каналів витоку інформації. Безперервне вдосконалення системи обумовлено появою нових способів доступу до інформації ззовні.

Забезпечення надійності систем інформаційного захисту, тобто контролю рівня надійності при відмові системи, появи збоїв, зломі та помилках.

Контроль функціонування системи захисту. Безперервне вдосконалення засобів і методів контролю над працездатністю механізмів захисту.

Удосконалення методів боротьби з шкідливими програмами і вірусами.

Оптимізація витрат на створення і експлуатацію систем контролю, що виражені в економічній доцільності застосування систем інформаційної безпеки.

1.1.4 Аналіз причин неефективної експлуатації систем захисту інформації.Головним принципом захисту інформації є комплексний підхід, тобто використання всіх доступних методів або засобів систем захисту інформації на підприємстві. Придбання і встановлення найсучасніших та найдорожчих технічних засобів захисту інформації не захистить інформацію, якщо не було проведено комплексний аналіз, організаційні заходи: не було інструктажу персоналу, не проведено навчання персоналу, не був створений контролюючий відділ спрямований на забезпечення захисту інформації, кожен з працівників не ознайомлений з відповідальністю за порушення режиму секретності і т. ін.

За відсутності законного користувача, контролю та розмежування доступу до термінала кваліфікований порушник легко використовує його функціональні можливості для несанкціонованого доступу до інформації, що підлягає захисту, шляхом уведення відповідних запитів або команд.

За наявності вільного доступу до приміщення можна візуально спостерігати інформацію на засобах відбиття і документування, викрасти

паперовий носій, зняти зайву копію, а також викрасти інші носії з інформацією: лістинги, магнітні носії та ін.

Особливу загрозу становить безконтрольне завантаження програмного забезпечення, в якому можуть бути змінені установки, властивості, дані, алгоритми, введено "троянську" програму або вкорінено комп'ютерний вірус, що виконують деструктивні несанкціоновані дії. Наприклад, записування інформації на сторонній носій, незаконне передавання у канали зв'язку, несанкціоноване друкування документів, порушення їх цілісності, несанкціоноване копіювання важливої інформації, вагомість якої визначається та обмежується на дуже короткий або, навпаки, тривалий час.

Загрозливою є ситуація, коли порушник – санкціонований користувач інформаційної системи, який у зв'язку зі своїми функціональними обов'язками має доступ до однієї частини інформації, а користується іншою за межами своїх повноважень. З боку санкціонованого користувача є багато способів порушення роботи інформаційної системи й одержання, модифікування, поширювання або знищення інформації, що підлягає захисту. Для цього можна використовувати, насамперед, привілейовані команди введення-виведення, неконтрольованість санкціонованості або законності запиту і звернень до баз та банків даних, серверів тощо. Вільний доступ дає порушникові можливість звертатись до чужих файлів і баз даних та змінювати їх випадково або умисно.

Під час технічного обслуговування апаратури можуть бути виявлені залишки інформації на її носіях (поверхні твердих дисків, магнітні стрічки та інші носії). Стирання інформації звичайними методами (засобами операційних систем, спеціальних програмних утиліт) неефективне з погляду технічного захисту інформації. Порушник може поновити і прочитати її залишки, саме тому потрібні тільки спеціальні засоби стирання інформації, що підлягає захисту.

Під час транспортування носіїв територією, яка не охороняється, виникає загроза перехоплення інформації, що підлягає захисту, і подальшого ознайомлення з нею сторонніх осіб.

Зловмисник може стати санкціонованим користувачем інформаційної системи у режимі розподілу часу, якщо він попередньо якось визначив порядок роботи санкціонованого користувача або якщо він працює з ним на одних лініях зв'язку. Він може здійснити підключення до лінії зв'язку між терміналом та процесором ЕОМ. Крім того, без переривання роботи санкціонованого користувача порушник може продовжити її від його імені, анулювавши сигнали відключення санкціонованого користувача.

Обробка, передавання та зберігання інформації апаратними засобами інформаційної системи забезпечуються спрацюванням логічних елементів на базі напівпровідникових приладів. Спрацювання логічних елементів зумовлено високочастотним зміщенням рівнів напруг і струмів, що призводить до виникнення в ефірі, ланках живлення та заземлення, а також у паралельно розміщених ланках й індуктивностях сторонньої апаратури електромагнітних полів, які несуть в амплітуді, фазі й частоті своїх коливань ознаки оброблюваної інформації. Використання порушником різних приймачів може призвести до несанкціонованого витоку та перехоплення дуже важливої інформації, що зберігається в інформаційній системі. Зі зменшенням відстані між приймачем порушника й апаратними засобами інформаційної системи ймовірність приймання таких інформаційних сигналів збільшується[6].

Як демонструє світовий досвід абсолютна більшість випадків втрати інформації, у тому числі в результаті атак, пов’язано з помилковими діями і неналежним ставленням до своїх обов’язків персоналу самих підприємств. Цілком зрозуміло, що проблема с помилковими рішеннями (діями) персоналу по захисту інформації особливо актуальна для підприємств, в яких не має штатних спеціалістів з захисту інформації. В своїй більшості це невеликі підприємства, офіси або відокремлені підрозділи державних установ, з

невеликим штатом працівників або низькою заробітною платнею, що не дозволяє належним чином забезпечувати свою інформаційну безпеку силами штатних спеціалістів. У відповідності зі загальноприйнятою практикою, у випадку відсутності штатного спеціаліста або підрозділу по захисту інформації, керівник назначає відповідального спеціаліста по захисту інформації. Як правило, це спеціаліст у галузі інформаційних технологій, а іноді і робітник яких ніколи раніше не мав справи з захистом інформації, на котрого додають окрім своїх основних функціональних обов’язків додаються також обов’язки з проведення заходів по захисту інформації. Слід зауважити, що не існує ніяких нормативних або інших документів які визначають вимоги до кваліфікації такого спеціаліста. Безумовно, що вузько спеціалізовані рішення з інформаційної безпеки, які буде приймати спеціаліст відповідальний за захист інформації будуть характерні для його досвіду з минулого місця роботи або місця навчання. З вищенаведеного, можливо зробити висновок, що однією з головних умов у комплексному підході є вибір кваліфікованих працівників з захисту інформації, та належний інструктаж працівників, які працюють з конфіденціальною інформацією.

1.1.5 Аналіз вірусних загроз у діяльності підприємства.Комп'ютерний вірус – різновид шкідливого ПО, яке має властивість розмножуватися шляхом створення копій самого себе, а також впроваджуватися в код інших програм, в розділи системної пам'яті, завантажувальні сектори. Це спеціально написана програма, найчастіше невелика за розмірами, яка поширює свої копії за допомогою різних каналів зв'язку. Зазвичай вірус запрограмований для порушення роботи програм, блокування роботи користувачів, знищення файлів, приведення в неробочий стан апаратних комплексів ПК.

Комп'ютерні віруси можна розділити на певні типи, в залежності від їх місця існування:

- завантажувальні віруси – віруси, які проникають в завантажувальний сектор пристроїв зберігання даних, таких як жорсткі диски, флешки, дискети і т.д., і здатні порушити доступність файлів;
- файлові віруси – тип вірусів, які впроваджуються у виконувані файли (файли з розширенням COM і EXE) і негативно впливають на їх функціональність;
- файлово-завантажувальні віруси – віруси, які об'єднують в собі функції двох попередніх типів вірусів;
- віруси документи – З огляду на вірусів, які заражають файли офісних систем. Такий вид ще називають «макровірусами», оскільки зараження системи відбувається за допомогою зараження макросів програми;
- мережеві віруси – тип вірусів, які поширюються за рахунок використання комп'ютерної мережі, тобто мережевих служб і протоколів.

Також віруси діляться на типи за принципом свого функціонування – віруси-паразити, віруси-станції, трояни, віруси-невидимки, віруси-шифратори, мутують, «відпочиваючі» віруси (запрограмовані на те, щоб активізуватися в певний час).

Для того, щоб виявити, видалити або захистити комп'ютер від вірусів, розробляються спеціальні програми. Ці програми називаються антивірусними і являють собою багатофункціональний продукт, який поєднує в собі такі засоби як: превентивні, профілактичні, кошти «лікування» або видалення, а також відновлення порушених чи втрачених даних.

Антивірусні програми діляться на певні типи:

- програми-детектори – ті, які допомагають знайти віруси в оперативній пам'яті або ж на носіях інформації, при цьому програми-детектори знайдені віруси не лікують;
- програми-доктора – програми, які на відміну від попереднього виду, не тільки знаходять вірус, але і лікують заражений файл, повертаючи його в початковий стан;

- програми-ревізори – такі програми мають властивість запам'ятовувати файл або системну область диска в його початковому стані, і пізніше порівнювати поточний стан з вихідним. При порівнянні файлу враховуються багато параметрів файлу, тому сховатися вірусу такі програми не залишають шансу;
- програми-фільтри – призначені для виявлення підозрілих дій в роботі комп'ютера. При спробі активізації вірусу програма може блокувати його роботу;
- вакцини – такі програми, які відразу запобігають зараженню різних файлів. Варто застосовувати такі програми, якщо програми-доктора відсутні. Але варто врахувати, що «вакцинація» можлива тільки проти вже відомих вірусів.

Проте, наявності одного лише антивірусного ПО на робочій станції є недостатнім для захисту загальносистемного і прикладного ПО і має підкріплюватися раціональними діями і обережністю при роботі в мережі самих користувачів[7].

За даними Брендона Гейлі у власному виданні. Статистика комп'ютерних вірусів. Відомо, що комп'ютерні віруси ставлять під загрозу користувачам конфіденційну інформацію, знищують дані та завдають пошкоджень обладнання вашої системи. Деякі з цих вірусів можуть призвести до збитків більше мільярдів.

Перелік типів вірусів та відсоток, які найчастіше вражають користувачів.

1. Віруси – 57%.
2. Замасковані трояни – 21%.
3. Троянські завантажувачі – 7%.
4. Небажане програмне забезпечення – 4%.
5. Рекламне програмне забезпечення – 3%.
6. Віруси-експлойти – 3%.
7. Віруси хробаки – 2%.

8. Викрадачі паролів та інструменти моніторингу – 2%.

9. Віруси-бекдори(потаємний хід) – 1%.

10. Шпигунське програмне забезпечення – 0,01%.

Віруси, які завдали найбільших грошових збитків. Найдорожчий комп'ютерний вірус всіх часів був віднесений до "MyDoom", що призвело до відшкодування 38 мільярдів доларів. Він швидко рухався, заражаючи відкриті мережі та кожен комп'ютер, що має доступ до нього. У 2004 році цей вірус, за оцінками, вплинув на 25% всіх листів. Нижче наведено список 10 найдорожчих програм вірусів, а також загальна сума збитків, що сталися.

1. MyDoom – $ 38 млрд;

2. So Big – $ 37,1 млрд;

3. ILOVEYOUE – $ 15 млрд;

4. Conficker – $9,1 млрд;

5. Code Red – $ 2 млрд;

6. Melissa – $ 1,2 млрд;

7. WannaCry – $ 1 млрд;

8. SirCam – $1 млрд;

9. SQL Slammer – $ 750 млн;

10. Nimda – $ 635 млн.[8].

Враховуючи економічне становище в деяких компаній, задля економії використовують попередні версії програмного забезпечення. Враховуючи той факт, що наприклад деякі версії операційних систем перестають обслуговуватись з годом, це вдало використовують зловмисники. Так 12 травня 2017 року велика кількість комп’ютерів з операційною системою Windows знаходились під вірусною атакою вірусом-вимагачем WannaCry. Вірус шифрував файли користувачів, щоб їх не можна було використовувати; за розшифрування зловмисники вимагали гроші. Уражено було близько 300 тисяч комп’ютерів у щонайменше 150 країнах світу. Збитки оцінювались більше ніж 1 мільярд доларів. Від атаки постраждала велика кількість державних компаній у різних країнах та більше ніж 70% від постраждалих

компаній, знаходились на території Росії та України. Після атаки Microsoft випустила оновлення пакетів безпеки для операційних систем, що вже не підтримувались компанією а саме: Windows XP, Windows Server 2003 та Windows 8. Тому спеціалісти радять, комп'ютери, що мають доступ до важливої інформації використовувати без доступу до мережі Інтернет або належно обслуговати їх.

1.2 Застосування штучного інтелекту в криптографії

1.2.1 Поняття штучного інтелекту. Поняття штучного інтелекту охоплює значний ряд методів та застосунків, що призначені для виконання комп'ютером завдань, які зазвичай вимагають уваги людського інтелекту. Іншими словами, штучний інтелект – це "здатність цифрового комп'ютера або керованого комп'ютером робота виконувати завдання, зазвичай асоційовані з інтелектуальними істотами" [9].

Основний тест для штучного інтелекту – тест Тюрінга, що був створений в 1950 Аланом Тюрінгом. Це тест на здатність машини проявляти інтелектуальну поведінку, що є еквівалентною, або такою що неможливо відрізнити від поведінки людини. [10]. Деякими із прикладів сфер застосування штучного інтелекту в наш час є розпізнавання мовлення, медицина, фінанси, авіоніка, навігація, робототехніка. Подальше дослідження в даній роботі ставить за мету огляд таких застосувань у криптографії.

1.2.2 Машинне навчання. Важливою категорією штучного інтелекту є машинне навчання. Машинне навчання частіше за все асоціюють із проблемами, що пов'язані з розпізнаванням образів. Складні набори потенційно неоднорідних даних у будь-якому сигналі, наприклад у зображенні чи спектральному представленні деякого звуку, вимагають категоризації на набір однорідних ознак, якими можна надалі описати та

класифікувати об’єкти заздалегідь відомим чином. Такі класифікації зазвичай асоційовані з статистичними вимірами, розрахованими з сигналу та статистичними чи геометричними характеристиками зображення. Якщо кластер таких характеристик в певному числовому векторі у заданих межах достатньо відрізняється щоб бути корельованим із якоюсь іншою відомою ознакою в наборі, то досягнення порогу прийняття рішення дозволяє прийняти рішення по класифікації з дотриманням обраного критерія.

Проблема часто полягає у знаходженні такого оптимального порогу прийняття рішень, при якому точність прийнятого рішення була б оптимальною, а оптимальне порогове значення підлягало б довірчому інтервалу. Зробивши поріг адаптованим до розмежування певних вхідних показників з точки зору їх відомої точності, кількості та іншої попередньої інформації, логіку процесу прийняття рішень можна зробити «м’якшою» з точки зору її толерантності до даних. Це є основою для впровадження так званих «систем нечіткої логіки», які створюють основи для розробки та впровадження штучних нейронних мереж. Такі системи зазвичай підвищують точність рішень, пов'язаних з класифікацією шаблону, ніж це можна отримати за допомогою звичайної категоризації даних на основі логічного та/або нечіткого логічного кількісного визначення.

1.2.3 Штучні нейронні мережі. Проста штучна нейронна мережа базується на введенні класу значень, що взяті за композитне представлення даних, вимірів, які обчислюються шляхом обробки даних для створення так званого «вектору ознак». Вага *w*застосовується для визначення впливу кожного компоненту на цей вектор (шляхом множення кожного компоненту вектору на його вагу) та суму всіх компонентів доданих разом для створення єдиного вихідного значення. Змінюючи значення цих *w*штучна нейронна мережа забезпечує результативність процесу прийняття рішень. Це приклад процесу передачі даних який перетворює множину входів у єдиний вихід. Він може бути відтворений для створення ідентичних вихідних сигналів

шляхом введення ідентичного набору вектору ознак з вагами, або створення різних вихідних сигналів шляхом зміни компонентів введеного вектору. Це створює єдиний шар вихідних сигналів, що можуть бути використані для вводу у певний «прихований шар», в якому ваги *w* можуть змінюватися. Відтворюючи цей процес *n* разів ми можемо створити багатошарову мережу, що складається з *n* шарів. На рис. 1.1 зображено приклад простої штучної нейронної мережі з одним прихованим шаром [11]. У цьому прикладі прихований шар складається з чотирьох вузлів. Обчислення, застосовані у трансфері інформації із прихованого шару в вихідний сигнал, враховують зміну числових значень у всіх чи деяких застосованих вагах. Це є прикладом нейронної мережі прямого поширення, в якій інформація передається лише в одному напрямку, тобто вперед, із вхідних вузлів, через прихований шар та у вихідні вузли. У цій мережі немає зацикленості. Таким чином, якщо три вхідні вузли мають числові значення (x_1, x_2, x_3),тоді входи до чотирьох вузлів прихованого шару за елементом вектору: ($w_{11}x_1 + w_{12}x2 + w_{13}x_3$, $w_{21}x_1 + w_{22}x_2 + w_{23}x_3$, $w_{31}x_1 + w_{32}x_2 + w_{33}x_3$, $w_{41}x_1 + w_{42}x_2 + w_{43}x_3$), де w_{ij}, i= 1, 2, 3, 4; j = 1, 2, 3 – ваги необхідні для модифікації трьох вихідних сигналів із вхідного шару, оскільки вони передаються в чотири вузли прихованого шару.

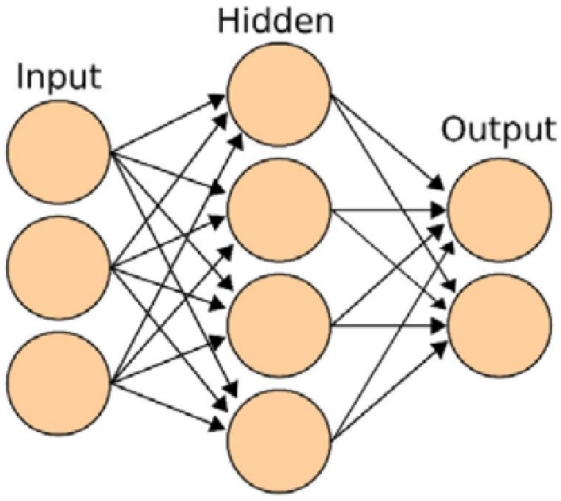

Рисунок 1.1 – Приклад простої штучної нейронної мережі

Із вищесказаного виникають чотири проблеми, що необхідно прийняти до уваги при створенні будь-якої подібної мережі входів-виходів:

1. яким повинен бути розмір вхідного вектору?
2. Скільки виходів повинно бути у мережі?
3. Скільки необхідно прихованих шарів та зі скількох вузлів вони повинні складатися?
4. Яким чином можна автоматизувати процес у якому вага компонентів вхідного вектору задається та потім змінюється?

Існує ряд різних, але тісно пов'язаних між собою методів для тренування штучних нейронних мереж шляхом налаштування ваги компонентів вхідного вектору допоки вихідний сигнал для певного взятого набору вхідних даних не буде таким, що відповідав би результатам тренувальної вибірки у межах визначеної похибки. Основний та можливо найширше застосовуваний метод – це застосування алгоритму зворотного поширення помилки [12]. Він базується на методі градієнтного спуску, ітеративний алгоритм оптимізації для знаходження локального оптимуму диференційовної функції. Обчислюючи градієнт функції відносно всіх ваг заданих дискретним вектором, ваги можна підігнати для генерації результату, близького до навчальної вибірки.

Існує широкий вибір програмного забезпечення та систем доступних для розробки додатків штучного інтелекту. MathWorksMATLAB, наприклад, надає спеціалізовані набори інструментів, включаючи можливості, які інтегрують штучний інтелект у робочий процес для розробки повністю спроектованих систем та машинного навчання з контрольованим та неконтрольованим навчанням. Розвиток систем штучного інтелекту з використанням програмування на Python стає дедалі поширенішим [13]. Однак незалежно від системи та мови програмування або набору інструментів, дизайн мережі є фундаментальним і має бути адаптований та оптимізованим для застосування штучного інтелекту за допомогою штучної

нейронної мережі для конкретного рішення. Особливе значення мають якість та кількість навчальних даних. Це визначає точність ваг, які поряд з дизайном мережі можна розглядати як «ключі» до майбутньої роботи мережі. Таким чином існує можливе застосування у криптографії, де вхідний вектор та ваги аналогічні криптографічним ключам та конструкції системи.

1.2.4 Обробка даних та глибинне навчання. Як сказано в попередньому розділі, типовим підходом для обробки даних є створення вектору ознак, що містить значення які є задовільним представленням та описом визначальних характеристик даних, наприклад цифрового сигналу. Таким чином кількість вузлів у вхідному шарі стає відносно невеликою відносно до початкових даних, тобто довжини цифрового сигналу. Це важливо для застосування неминуче обмежених обчислювальних можливостей, доступних для обслуговування штучної нейронної мережі з метою виконання ефективного процесу прийняття рішень, тобто задля економії часу та ресурсів необхідних для обчислення. Але, в деяких випадках, дуже складно визначити у повністю кількісному вираженні елементи вектору ознак які є унікальними та однозначними характеристиками даних. Проблема часто вирішується за допомогою дослідження нових властивостей даних. Наприклад, текстура зображення може бути кількісно виражена за допомогою принципів фрактальної геометрії та обчислювальних метрик. Це дозволяє більш визначальним ознакам зображення, або зображенню в цілому, наприклад, описаними доволі неоднозначним поняттям текстури, бути кількісно вираженими через поняття фрактальної геометрії у цифровій фотографії[14][15].

Визначення того, який вектор ознак слід використовувати, та елементи, які він містить, визначає розмір вектора ознак, який в ідеалі повинен бути набагато меншим за вихідні дані, з яких він був побудований за допомогою кількох різних алгоритмів обробки сигналів або зображень, щоб вивести конкретні показники. Тут є проблема оптимальності що вимагає розглядання,

а саме відношення часу, необхідного для обробки обчислень вектору ознак до часу, що необхідний для навчання штучної нейронної системи. Проте останнім часом обчислювальна здатність доступна завдяки розвитку апаратного забезпечення збільшилась настільки, що замість переробки сигналів на композитні вектори ознак цей процес часто ігнорується і сирі дані, тобто початковий цифровий сигнал, використовуються прямо для введення у штучну нейронну мережу. Разом із багатьма шарами великої кількості вузлів, цей підхід істотно підвищую складність штучної нейронної мережі, але зменшує необхідність вирішувати які сигнали алгоритм обробки вимагає для генерування меншого вектору ознак.

Цей принцип пов’язаний із більш сучасним підходом до штучного інтелекту, відомим як глибинне навчання, в якому по суті відмовляються від попередньої обробки сигналів, а штучні нейронні мережі з високою пропускною здатністю даних призначені виключно для обробки вхідних даних. Рис. 1.2 демонструє різницю між стандартною мережею із одним прихованим шаром та глибокою мережею, що містить три приховані шари [16].

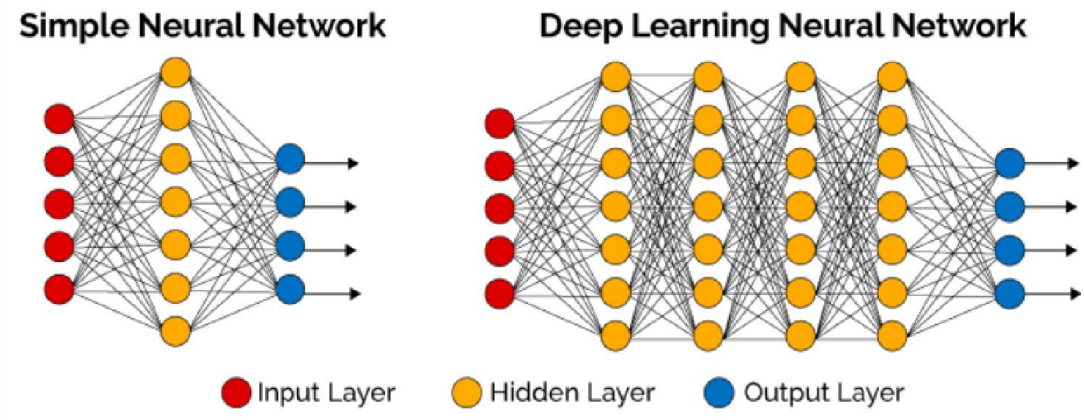

Рисунок 1.2 – Порівняння простої та глибокої нейронних мереж

Незалежно від доступної обчислювальної потужності, штучні нейронні мережі вимагають для роботи в реальному середовищі дані із реального світу Хоча ця необхідність і догоджується розвитком доступності до «великих даних», в деяких випадках певні дані можуть бути не доступними або

неіснуючими. В таких випадках штучні нейронні мережі також можуть бути корисними завдяки їх навчанню для створення або доповнення недостатніх даних, використовуючи інші пов’язані реальні дані, які є доступними. Таким чином штучний інтелект може використовуватися для створення нових даних на основі уже існуючих.

Разом із навчальними вибірками, призначення штучних нейронних мереж може значно відрізнятися, наприклад, в кількості шарів, кількості вузлів та гомогенності. Це включає розсіювання значень ваги по шарах мережі та розподіл по певних глибинах шляхом застосування різних просторово-інваріантних фільтрів за допомогою дискретної операції згортання. Ці згорткові нейронні мережі були вперше представлені на початку 1980-х років. Згорткова нейронна мережа – це клас штучних нейронних мереж глибинного навчання, розроблений в основному в сфері обробки цифрових зображень та комп’ютерного зору [17]. В цьому контексті, вони замінюють собою багато алгоритмів обробки цифрових зображень, створених для розпізнавання образів використовуючи, серед інших методів, згорткові фільтри для сегментації зображень на області подібності та розриви [18]. Цей підхід був і залишається вживаним для розробки рядку визначених, геометричних та текстурних вимірів, які традиційно використовуються для створення систем прийняття рішень на основі нечіткої логіки або для навчання штучних нейронних мереж із низьким числом вхідних вузлів та прихованих шарів.

Застосування рішень для глибинного навчання для вирішення складних проблем розпізнавання шаблонів є ключем до майбутнього розвитку багатьох технологій та середовищ програмування, функцій та систем, усі з яких знаходяться у процесі швидкого розвитку. Наприклад, нещодавно MATLAB розробили новий набір інструментів для вирішення проблем з використанням глибокого навчання [19].

Застосування глибокого навчання до криптографії виходить за рамки цієї роботи. Однак деякі підходи природно піддаються поліпшенню

продуктивності використовуючи методи глибокого навчання. Особливо це стосується заміни традиційних методів криптоаналізу, які з огляду на деякі розробки стають все більш зайвими [20].

1.2.5 Штучні нейронні мережі в криптографії. Однією з головних особливостей штучного інтелекту є його здатність розпізнавати закономірності в складних даних. Криптографія спирається на максимізацію поширення та заплутаності, пов'язаної з перетворенням відкритого тексту у зашифрований. В ідеалі зашифровані дані повинні бути повністю позбавлені будь-якого виду шаблонності. Потрібно, щоб шифрований текст був складним, рандомізованим поданням відкритого тексту, який зазвичай асоціюється із застосуванням односторонньої функції, яка не має зворотного рішення. Це дає можливість застосування штучного ітелекту щодо генерації шифрів, їх класифікації (тобто їх криптографічної міцності) та аналізу чи криптоаналізу зашифрованих даних.

Оскільки криптографія спирається на перетворення відкритого тексту в зашифрований текст на основі генерації поля випадкових чисел (шифру), припустимо, ми вводимо кілька початкових випадкових чисел для створення мережі, а потім тренуємо її за допомогою реального джерела шуму, дозволяючи щоб ваги могли бути відрегульованими, щоб мережа виводила досить точне моделювання цільового поля випадкових даних. У цьому випадку, за умови, що дизайн мережі відомий двом об’єктам, що спілкуються, Алісі та Бобу, початкові випадкові числа та ваги представляють ключі, необхідні для відтворення поля випадкових чисел, яке можна використовувати для шифрування, а потім розшифрування даних (за умови, що Алісі та Бобу відомі як початкові випадкові числа, так і ваги через застосування протоколу обміну ключами).

Дизайн мережі (включаючи обчислені ваги) еквівалентний алгоритму шифрування/дешифрування. Таким чином, мала кількість випадкових елементів у вхідному векторі ознак може генерувати поле випадкових чисел,

що становить шифр. Це базується на застосуванні техніки машинного навчання під назвою Evolutionary Computing, яка може бути використана для створення нелінійного рівняння, яке після ітерації може створити поле випадкового числа, початковою умовою ітератора є звичайний ключ.

Так само, як штучний інтелект можна використовувати для створення шифру, здатність штучного інтелекту розпізнавати складні шаблони підпорядковується виконанню завдання розпізнавання шаблонів у зашифрованому тексті і тим самим шукає підписи, які визначають близькість або "точку атаки" там, де є слабкі місця у рандомізованому символі зашифрованих даних. Враховуючи, що потужний механізм шифрування не повинен генерувати таких слабкостей, штучний інтелект пропонує додатковий підхід до перевірки міцності методу шифрування, включаючи ті методи, які ґрунтуються на використанні штучного інтелекту для генерації зашифрованих даних. Таким чином, штучний інтелект відіграє роль як у шифруванні даних, так і в криптоаналізі. В останньому випадку це можна поширити на проблеми, пов'язані з приховуванням інформації, коли потрібно виявити підпис відкритого тексту, прихованого в іншому відкритому тексті (стеганаліз), і підпис зашифрованого тексту, прихованого у відкритому тексті (стеганокриптоаналіз).

Кібербезпека передбачає цілий ряд методів, призначених для захисту створення, обробки, зберігання та передачі даних через відкриту мережу, таку як Інтернет. Конкретний і відомий приклад – це зараження файлів вірусом, призначеним для таємного проникнення на окремий комп’ютер або мережу комп’ютерів. Комп'ютерні віруси стали складними і працюють у прихованому режимі, щоб уникнути виявлення.

Кожен день створюються нові віруси. Однак більшість цих нібито «нових» вірусів не є абсолютно новими, але часто є варіаціями (деякі з них відносно незначні) їхніх попередників. Багато з цих вірусів просто змінюють свою форму та підписи, щоб уникнути виявлення, але їх робота та спосіб зараження файлів та систем залишається незмінним. Це дає можливість

навчати мережеві мережі щодо низки відомих вірусів за припущення, що їх цифрові підписи можуть бути розпізнані шляхом перевірки майбутніх потоків даних та нових файлів [21][22].

1.2.6 Генетичні алгоритми. Генетичні алгоритми[23] – адаптивні евристичні алгоритми, що головним чином застосовуються у задачах оптимізації та пошуку. Вони базуються на принципах природньої селекції та генетики, належать до класу еволюційних алгоритмів, а також використовують біологічні еволюційні методи (оператори мутації, кросинговер, та селекція).

Селекція – це числова операція, де хромосоми із популяції обираються для репродукції на основі значення, розрахованого фітнес-функцією для зазначеної хромосоми.

У операції кросинговеру беруться дві хромосоми і генеруються нова, беручи деякі признаки із однієї батьківської хромосоми, а інші признаки із другої хромосоми. Наприклад, два рядки 11110010 та 01001111 можуть бути схрещені для отримання двох нових хромосом: 11110111 та 01001010. Кросинговер може проводитися більш ніж у одній точці.

Мутація використовується для підтримки генетичної різноманітності популяції із покоління у покоління, подібно біологічним мутаціям. ГА використовують побітові операції для рядків значень у хромосомах-кандидатах, включаючи біт-реверс. Ці операції випадково змінюють значення кількох бітів у хромосомі. Наприклад, рядок 000000111 може бути мутований у його п’ятій позиції на 00001111. Базовий цикл ГА зображений на рис. 1.3.

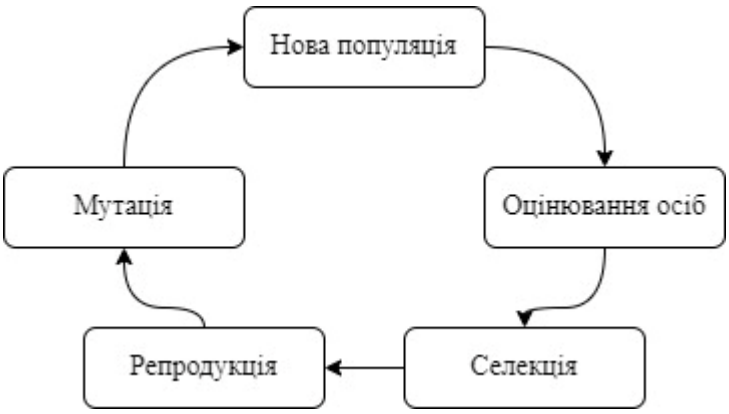

Рисунок 1.3 – базова модель циклу ГА

Головною ідеєю, закладеною в генетичний алгоритм є реплікація природньої випадковості, де популяція індивідів адаптується до навколишнього середовища за законами натуральної селекції. Це означає, що виживання та репродукція кожного окремого представника популяціїдосягається шляхом елімінації слабкостей. ГА створює популяцію таким чином, що признаки, які є домінантними та мають вище значення для фітнес-функції відтворюються частіше ніж решта популяції. Це робить ГА хорошим кандидатом для процесу генерації криптографічних ключів.

Основною задачею використання ГА у даній роботі є створення ключа та набору під-ключів, які будуть використані для забезпечення надійності да збереження інформації шляхом покращення криптографічних властивостей алгоритму ММВ.

1.2.7 Генерація криптографічних ключів. Будь-яка криптосистема складається із певної кількості алгоритмів та ключів, де ключ – це структура із впорядкованого ланцюга бітових значень, що визначаються за секретним значенням. Ключ погоджується авторизованими користувачами та інкорпорується в алгоритм для подальшого шифрування да дешифрування. Безпека зашифрованих даних повністю залежить від двох важливих чинників: ключового простору та чутливості, а також від сили криптографічного алгоритму.

Безумовно, чутливість криптографічного ключа є суттєвою характеристикою для запобігання статистичних та диференціальних атак на криптосистему. Визнано, що із збільшенням значення довжини ключа також пропорційно збільшується і чутливість до його змін у системі [24]. Загалом, цю чутливість можна описати як вимір мінливості вихідного результату системи у залежності до певних змін вхідних значень. Для генерації під-ключів, чутливість означає відношення відсотку зміни значень у кожному з під-ключів до зміни значення одного біту у початковому ключі.

Будь-яка спроба криптографічної атаки відбувається або способом вичерпного пошуку (метод грубої сили), або за допомогою диференціального криптоаналізу. Параметри, які використовуються для визначення вірогідності успішності таких атак базуються на довжині ключа та складності алгоритму, що використовується для його генерації. Ключ може бути ускладненим використанням більш складних методів в процесі його генерування. У цій роботі пропонується використання генетичного алгоритму у процесі генерації криптографічного ключа для його ускладнення, а також зменшення значення чутливості під-ключів без помітних жертв у загальній швидкодії системи.

1.3 Постановка задачі

Цілі: проведення аналізу стану, розробка та надання рекомендацій щодо впровадження комплексних систем захисту інформації на підприємстві ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ.

Об’єкт: підприємство ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ.

Предмет: аналіз стану та комплексний захист інформації на підприємстві ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ було поставлене завдання на проведення аналізу стану та комплексного захисту інформації на підприємстві.

Виходячи з аналізу проблеми сформовано завдання та критерії проведення комплексного аналізу.

Основною задачею при реалізації проекту є забезпечення функційоцінкистану систем захисту інформації:

- проведення аналізу мережі;
- проведення аналізу програмного забезпечення, що встановлене на робочі комп’ютери;
- проведення аналізу фізичних засобів захисту інформації;
- проведення аналізу роботи з конфіденційною інформацією.

А також створення системи інформаційного захисту, програми передачі даних та повідомлень з використанням алгоритму шифрування ММВ на базі операційної системи Windows із імплементацією модуля генерації криптографічних ключів за допомогою генетичного алгоритму.

РОЗДІЛ 2
ПРОВЕДЕННЯ АНАЛІЗУ СТАНУ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ НА ПІДПРИЄМСТВІ ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ

2.1 Загальні відомості про підприємство

Рисунок 2.1 – Структура управління підприємством

Основні види діяльності підприємства:

- вантажний автомобільний транспорт;
- діяльність посередників у торгівлі сільськогосподарською сировиною, живими тваринами, текстильною сировиною та напівфабрикатами;
- оптова торгівля молочними продуктами, яйцями, харчовими оліями та жирами;
- вантажний автомобільний транспорт (основний);
- надання в оренду й експлуатацію власного чи орендованого нерухомого майна;
- надання в оренду автомобілів і легкових транспортних засобів;
- надання в оренду вантажних автомобілів.

2.2 Характеристика комп'ютерної мережі у відділах підприємства

Кількість комп'ютерів та вид використаної топології зображено у таблиці 2.1.

Таблиця 2.1 – Характеристика відділів з комп'ютерною мережею

Назва	Посади	Кількість комп'ютерів	Топологія
Адміністративно-господарський відділ	директор, головний юрист, помічник юриста, секретар	5	деревоподібна змішана (шина та зірка).
Відділ прийому та зберігання	завідуючий складом, старший комірник	2	деревоподібна зірка
Відділ закупівель та збуту	маркетолог, менеджер зі збуту, менеджер по закупках	3	деревоподібна зірка
Бухгалтерія	головний бухгалтер, бухгалтери	5	деревоподібна шина
Відділ кадрів	начальник відділу кадрів, старший інспектор відділу кадрів	2	деревоподібна зірка
Лабораторія	головний технолог, старший лаборант, лаборанти	4	деревоподібна зірка
Комунальне забезпечення та транспорт	начальник відділу, старший інспектор, диспетчер	3	деревоподібна зірка

2.3 Управління доступом на підприємстві

Для управління доступом, потрібно розмежувати доступ до інформації на сервері для кожної посади. Для того щоб отримати доступ до різних видів інформації працівники використовують паролі, що надають певні права залежно від посади. Умовно за рівнем важливості інформація поділена на такі категорії:

- життєво важлива інформація – незамінна інформація, наявність якої необхідна для функціонування системи (інформація про банківські рахунки, особисті дані працівників і т. ін.);
- важлива інформація – інформація, що може бути замінена чи відновлена, але процес її відновлення важкий і зв’язаний з великими витратами (кількість продукції та сировини, бази даних постачальників і т. ін.);
- корисна інформація – інформація, яку важко відновити, однак система може досить ефективно функціонувати без неї;
- несуттєва інформація – інформація, без якої система продовжує існувати.

Для більш зручного та наочного представлення моделі керування та розмежування доступом на підприємстві використано матрицю доступу. Матриця доступу являє собою таблицю, у які зображено список робіт або процеси та користувачів, що беруть участь у робочому процесі. Елементами матриці є рівні доступу для кожного користувача враховуючи його повноваження відносно кожного списку робіт, процесу або об’єкту.

Керуючись постановою Кабінету міністрів України № 373. Потрібно виконувати наступні вимоги до забезпечення захисту інформації в системі: Відкрита інформація під час обробки в системі повинна зберігати цілісність, що забезпечується шляхом захисту від несанкціонованих дій, які можуть призвести до її випадкової або умисної модифікації чи знищення. Усім користувачам повинен бути забезпечений доступ до ознайомлення з відкритою інформацією. Модифікувати або знищувати відкриту інформацію можуть лише ідентифіковані та автентифіковані користувачі, яким надано відповідні повноваження. Спроби модифікації чи знищення відкритої інформації користувачами, які не мають на це повноважень, неідентифікованими користувачами або користувачами з не підтвердженою під час автентифікації відповідністю пред'явленого ідентифікатора повинні блокуватися. Під час обробки службової і таємної інформації повинен

забезпечуватися її захист від несанкціонованого та неконтрольованого ознайомлення, модифікації, знищення, копіювання, поширення. Доступ до службової інформації надається тільки ідентифікованим та автентифікованим користувачам[25]. Спроби доступу до такої інформації неідентифікованих осіб чи користувачів з не підтвердженою під час автентифікації відповідністю пред'явленого ідентифікатора повинні блокуватися. Згідно вимоги до забезпечення захисту інформації в системі було створено матрицю доступу (Табл. 2.2) для наочного зображення доступу до інформації у кожного з користувачів.

Таблиця 2.2 – Матриця доступу на підприємстві

Умовні позначення: + – повний доступ; Ч – лише для читання; Р – редагування О – обмежений доступ.	Сховище керівництва	Сховище бухгалтерії	Сховище персональних даних	Сховище технічної документації	Сховище транспорту	Сховище лабораторії	Інтернет без соц.мереж	Інтернет без обмежень	Склад	Комерційна інформація
Директор	+	+	+	Ч	Ч	Ч		+	Ч	+
Головний юрист		Ч	Ч					+		+
Помічник юриста		Ч	Ч				+			Ч
Секретар	Ч		Ч				+	О		
Завідуючий складом					Ч		+		+	
Старший комірник							О		+	
Маркетолог				Ч			+		Ч	+
Менеджер зі збуту				Ч	Ч		+		Р	+
Менеджер по закупках				Ч	Ч		+		Р	+
Головний бухгалтер	+	+	+					+	Р	Ч
Бухгалтери		+	Ч				+		Ч	Ч

Продовження таблиці 2.2

Начальник відділу кадрів	+		+					+		
Старший інспектор відділу кадрів	Ч		+					+		
Головний технолог				Р		Ч	+		Ч	Ч
Старший лаборант				Ч		+	+		Ч	
Лаборанти				Ч		Р	+		Ч	
Начальник відділу комунального забезпечення				Ч	+	Ч	+		Ч	
Старший інспектор					Р	Ч	+		Р	
Диспетчер					Р		+		Ч	Ч
Начальник охорони			О						О	
Охоронці									О	

Згідно другої частини пункту сьомого постанови Кабінету міністрів України: У системі забезпечується можливість надання користувачеві права на виконання однієї або кількох операцій з обробки службової інформації або позбавлення його такого права. Для наочного зображення прав доступу до службової інформації створено таблицю рівнів секретності (Табл 2.3) та зображено рівні доступу за допомогою методів мандатного керування доступом.

Методи мандатного керування доступом застосовуються до тих баз даних, у яких збережена інформація має досить статичну і тверду структуру, що властиво, наприклад, деяким військовим або урядовим організаціям. Основна ідея полягає в тому, що кожному об'єктові даних привласнюється деякий класифікаційний рівень (classification level) (або необхідний гриф таємності, наприклад "Абсолютно секретно", "Секретно", "Для службового користування" і т.д.), а кожному користувачеві надається рівень допуску (clearance level) із градаціями, аналогічними існуючим класифікаційним

рівням. Передбачається, що ці рівні утворять строгу ієрархічну систему (наприклад, "Абсолютно секретно" > "Секретно" > "Для службового користування" і т.д.).

Таблиця 2.3 – Таблиця рівнів секретності інформації

Гриф секретності	**Інформація**
Несекретно	Довідкова технічна інформація
Для службового користування	Комерційна інформація
Інформація з обмеженим доступом	Сховище керівництва, сховище бухгалтерії, персональні дані працівників
Секретно	Угоди, банківські рахунки, фонд заробітної плати
Абсолютно секретно	Розробки конструкторського бюро, банківські рахунки, патенти

Задаємо такі рівні доступу (згідно таблиці 2.3):

- директор – 1;
- головний юрист, головний бухгалтер, начальник відділу кадрів, головний технолог – 2;
- бухгалтери, юристи, маркетолог, менеджер по закупках, менеджер зі збуту, секретар, системний адміністратор, завідуючий складом, фахівець з хімічної лабораторії, директор з виробництва, старший інспектор – 3;
- менеджери з продажу, начальник охорони, начальник робочих підрозділів, охоронці – 4;
- робітник складу, робочі, лаборанти – 5.

2.4 Серверний захист

На підприємстві ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ сервер розташований у приміщенні ІТ-відділу, відповідальним за обслуговування серверу є системний адміністратор. На сервері існує рівень доступу до інформації згідно матриці доступу (Табл 2.2)

дані для авторизації працівник отримає при проведенні організаційних робіт з забезпечення захисту інформації. Надано рекомендацію, щодо переміщення серверу або створення окремої шафи для серверу з додатковим фізичним захистом. Операційна система, що використовується на сервері – Ubuntu 14.04 LTS. Операційна система Ubuntu є надійною, безкоштовною та стабільною, не потребує частих перезавантажень, не потребує антивірусів, та має відносно легкий рівень у налаштуванні, реліз нової версії відбувається кожні півроку. Тому було надано рекомендацію, щодо оновлення операційної системи на сервері до версії Ubuntu 18.04 LTS, у цілях отримання останніх налаштувань та підтримки, оскільки обслуговування версії Ubuntu 14.04 закінчується у 2019 році в квітні місяці, а остання версія Ubuntu 18.04 LTS матиме, підтримки до квітня 2023 року. Головною перевагою Ubuntu є її безкоштовність та надійність відносно інших серверних ОС.

2.5 Ативірусний захист

На підприємстві ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ встановлено антивірусне програмне забезпечення NOD 32. Оскільки антивірусна база значний час не оновлювалась, було проведено аналіз антивірусного програмного забезпечення. Для аналізу було обрано Panda Antivirus Pro, 360 Total Security, ESET NOD 32, Avira Free Antivirus, Dr. Web Antivirus, Microsoft Essential.

Оскільки одним з найважливішим критеріїв є часті оновлення, та безкоштовність і ефективність у роботі. Було обрано Microsoft Essential. Головними перевагами, є його абсолютна безкоштовність, евристичний алгоритм здатний виявляти будь-які загрози, навіть ті яких не має в базах визначених вірусів, кожного дня бази оновлюються по декілька разів, технологія сканування за часом дозволяє сканувати комп’ютер відповідно до режиму роботи підприємства. Недоліками є лише, відсутність вбудованого брандмауеру, проте у умовах підприємства, системний адміністратор

здійснює налаштування брандмауеру у ручному порядку, робота антивірусу не може бути завершеною завчасно. Враховуючи малу значимість недоліків та велику низку переваг, MS Essential було рекомендовано до встановлення для захисту від вірусних загроз на підприємстві.

2.6 Організація та фізичний захист

Відповідальним за проведення організаційних робіт з забезпечення захисту інформації, наказом підприємства було уповноважено виконавчого директора. Інструктаж на підприємстві проводиться згідно Законів України, та постанов Кабінету міністрів України, щодо захисту інформації. Кожен працівник освідомлений, освідомлений рівнем доступу та має дані для авторизації для доступу до інформації на сервері згідно (Табл 2.2) матриці доступу. Також кожен працівник освідомлений за відповідальність у разі порушення Закону України або нормативно правових-актів чи наказів підприємства, щодо збереження цілісності інформації.

2.7 Підсумки аналізу та надані рекомендації

Проведення аналізу заходів захисту інформації можна розділити на декілька груп.

Технічні (апаратні засоби) – це різні за типом пристрої (механічні, електромеханічні, електронні та ін), які апаратними засобами вирішують завдання захисту інформації. Вони або перешкоджають фізичній проникненню, або, якщо проникнення все ж таки відбулося, доступу до інформації, у тому числі за допомогою її маскування. На постах охорони проведено зв’язок для повідомлення проти спроб проникнення або інших загроз, на першому поверсі та також у відділах дирекції та бухгалтерії встановлено грати на вікна, кожен з відділів захищений замками, інформація про винаходи та випробування конструкторського бюро зберігається у

сейфах, було надано рекомендацію для поміщення серверу до приміщення що знаходиться під додатковим фізичним захистом.

Програмні засоби включають програми для ідентифікації користувачів, контролю доступу, шифрування інформації, видалення залишкової (робочої) інформації типу тимчасових файлів, тестового контролю системи захисту та ін. Переваги програмних засобів – універсальність, гнучкість, надійність, простота установки, здатність до модифікації та розвитку. Недоліки – обмежена функціональність мережі, використання частини ресурсів файл-сервера і робочих станцій, висока чутливість до випадкових або навмисним змін, можлива залежність від типів комп'ютерів (їх апаратних засобів). На даному підприємстві включають блокування портів проти розподілених атак спрямованих на відмову в обслуговуванні, а також доступ до інформації на сервері здійснювався завдяки паролю відповідно рівня доступу користувача, було надано рекомендацію по встановленню паролю для доступу до комп’ютерів, а також встановлення антивірусного програмного забезпечення.

Криптографічні методи захисту інформації – це методи захисту даних із використанням шифрування. Криптографічні засоби відсутні.

З вищенаведеного, можна зробити висновок, що рівень захищеності підприємства є недостатнім у криптографічному та програмному методах. Для забезпечення програмного методу, було надано рекомендації, а для забезпечення криптографічного захисту було прийнято рішення розробити програмний засіб з криптографічним алгоритмом шифрування інформації та повідомлень.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ШИФРУВАННЯ ПОВІДОМЛЕНЬ ТА ФАЙЛІВ

3.1 Вибір алгоритму шифрування

Криптографія – це усюдисущий інструмент у світі інформаційної безпеки. Він необхідний, коли потрібно зберегти секретність комунікацій або довести автентичність повідомлення, його можна використовувати для створення різних багатопартійних протоколів таким чином, що робить їх обхід, злом чи обман важким та непомірно дорогим [29]. Насправді сфера застосування криптографії неймовірно широка, і неможливо було б скомпілювати повний перелік функціональних можливостей, яких можна досягти за рахунок її використання.

З історичної точки зору, найстаріша та найголовніша криптографічна мета – це конфіденційність інформації, та існує багато методів та алгоритмів що можна використати для її досягнення. Один з них – це модульний блочний алгоритм шифрування на основі множення MMB.

Блочний алгоритм шифрування MMB[27] був розроблений Йоаном Дайменом в 1993 році, та запропонований як альтернатива шифру IDEA[28]. MMBбув розроблений спеціально для протистояння диференціальному криптоаналізу [30]. Його головною інновацією стало використання циклічного множення в групі $Z_2^n{}_{-1}$, де n – це довжина одного слова, з якими будуть проводитися операції. Всі внутрішні операції MMB проводяться з n-бітовими словами. Творці шифру запропонували n= 32, таким чином множення відбувається в групі $Z_2^{32}{}_{-1}$. Також можна зауважити, що $2^{32} - 1 = (2^{16}+1) \cdot (2^8+1) \cdot (2^4+1) \cdot (2^{12}+1) \cdot (2^3+1) = 65537 \cdot 257 \cdot 17 \cdot 5 \cdot 3 = 4294967295$, добуток усіх відомих простих чисел Ферма. MMB – це ітеративний шифр, що складається з шести раундів. MMB оперує 128-

бітовими текстовими блоками та використовує 128-бітовий ключ. Схема алгоритму описана в Рис. 3.1.

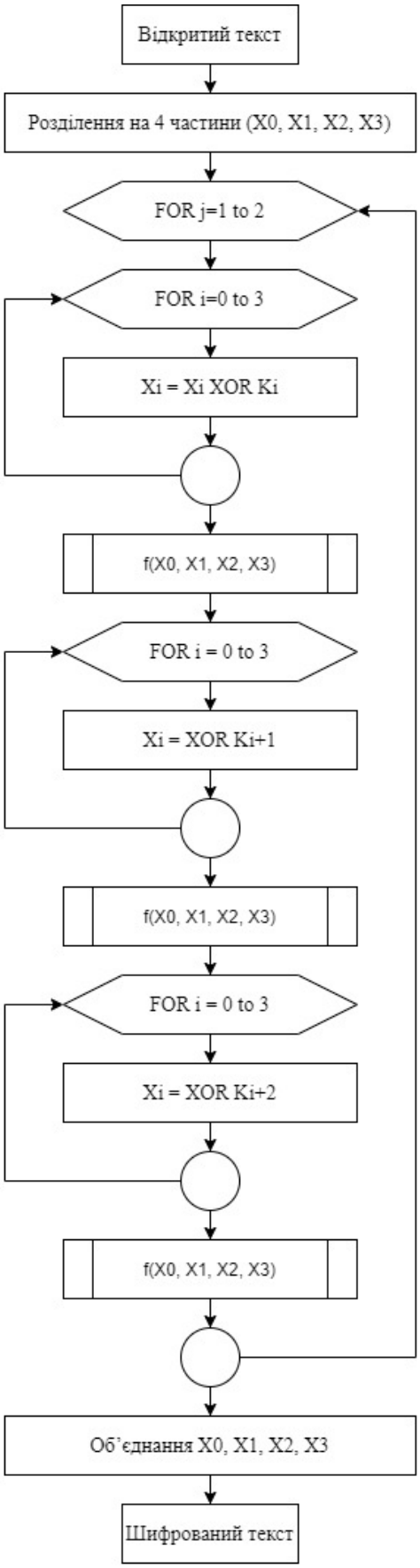

Рисунок 3.1 – Блок-схема алгоритмуMMB

ММВ оперує 32-бітовими підблоками тексту (x_0, x_1, x_2, x_3)і 32-бітовими підблоками ключа (k_0, k_1, k_2, k_3). Це робить зручним реалізацію алгоритму на 32-бітних процесорах. Чергуючись з XOR, шість разів використовується нелінійна функція *f*.

Алгоритм:

$x_i = x_i \oplus k_i$, для I = 0 до 3

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

$x_i = x_i \oplus k_{i+1}$, для I = 0 до 3

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

$x_i = x_i \oplus k_{i+2}$, для I = 0 до 3

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

$x_i = x_i \oplus k_i$, для I = 0 до 3

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

$x_i = x_i \oplus k_{i+1}$, для I = 0 до 3

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

$x_i = x_i \oplus k_{i+2}$, для I = 0 до 3

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

У функції f три етапи:

1. $x_i = c_i$, для I = 0 до 3 (Якщо на вході множення одні одиниці, то на виході – також одні одиниці).

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

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

Всі операції з індексами виконуються по модулю 3. Операція множення на етапі (1) виконується по модулю 2^{32}-1. В даному алгоритмі якщо другий операнд - 2^{32}-1, то результат також 2^{32}-1. В алгоритмі використовується константи:

C=2aaaaaaa

c_0=025flcdb

$c_1 = 2 * c_0$

$c_2=2^3 * c_0$

$c_3=2^7 * c_0$

Константа C – це «найпростіша» константа з високою трійковою вагою, нульовим молодшим бітом і без кругової симетрії. У константи c_0 дещо інші характеристики. Інші константи є зміщеними версіями c_0, і використовуються для попередження розкриття на основі симетрії.

Дешифрування є оберненим процесом.

3.2 Вибір мови програмування та середовища розробки

Для реалізації шифру MMBбуло обрано мову програмування C#. Тому, що вона має необхідний набір інструментів для роботи з шифрами, та на основі досвіду з інших навчальних дисциплін та набутих навичок.

C # (C Sharp) – об'єктно-орієнтована мова програмування. Розроблена в 1998-2001 роках групою інженерів під керівництвом Андерса Хейлсберг в компанії Microsoft як основна мова розробки додатків для платформи Microsoft .NET. Компілятор з C # входить в стандартну установку самої .NET, тому програми на ньому можна створювати і компілювати навіть без інструментальних засобів, на кшталт Visual Studio. C # відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C ++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML. Перейнявши багато від своїх попередників – мов C ++, Java, Delphi, Модула і Smalltalk – C #, спираючись на практику їх використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем: так, C # не підтримує множинне успадкування класів (на відміну від C ++).

C # розроблявся як мова програмування прикладного рівня для CLR і, як такий, залежить, перш за все, від можливостей самої CLR. Це стосується, перш за все, системи типів C #, яка відображає BCL.

Присутність або відсутність тих чи інших виразних особливостей мови диктується тим, чи може конкретна мовна особливість бути трансльований в відповідні конструкції CLR. Так, з розвитком CLR від версії 1.1 до 2.0 значно збагатився і сам C #; подібної взаємодії слід очікувати і в подальшому. (Однак ця закономірність була порушена з виходом C # 3.0, що представляє собою розширення мови, що не спираються на розширення платформи .NET.) CLR надає C #, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування.

Наприклад, прибирання сміття не реалізована в самому C #, а проводиться CLR для програм, написаних на C # точно так же, як це робиться для програм на VB.NET, J # та ін.[31].

Завдяки достатньому та зручному набору інструментів і безкоштовності було, обрано середовище розробки VisualStudioCommunity 2019.

3.3 Опис алгоритму шифрування.

Блок шифру MMB має структуру мережі замін-перестановок (SP-мережа) і працює в 128-бітних текстових блоках, використовує 128-бітний ключ та має шість ітерацій. Один раунд MMB складається з чотирьох трансформацийj:

- σ [kj]: виключна диз'юнкція застосовується для кожного слова данихз підблоком ключа kj, де j – індекс раунду. Формальне представлення зображене в формулі 3.1, де $\oplus$ означає побітову виключну диз’юзкцію, $a_i, k_i^j \in Z_{2^{32}}$ для 0≤ i≤ 3.Операція σ [kj] є інволюцією і є єдиною операцією, що залежить від ключа в раунді.

$$\sigma[k^j](a_0, a_1, a_2, a_3) = \left(a_0 \oplus k_0^j, a_1 \oplus k_1^j, a_2 \oplus k_2^j, a_3 \oplus k_3^j\right) \quad (3.1)$$

- γ: модульне множення кожного слова даних із фіксованими 32-бітовими константами Gi (Формула 3.2), де a ⊗ b= a ∗ b mod (2^{32} - 1), G0 = $025F1CDB_x$, G1 = 2 ⊗ G0 = $04BE39B6_x$, G2 = 8⊗G0 = $12F8E6D8_x$, а G3 = 128⊗G0 = $2F8E6D81_x$,які можна ефективно обчислити, оскільки (A ∗ 2^x) mod (2^{32} - 1) = (A ≪ x) mod (2^{32} - 1). У множення по модулю 2^{32} - 1 спостерігається ефект обтікання, оскільки $2^{32} \equiv 1$ mod (2^{32} - 1), що означає, що біти на (32 + i)-ій LSB (молодший біт) позиції зміщені в і-ту позицію LSB. Цей ефект схожий на операцію множення по модулю 2^{16} +1 в IDEA. Вся операція також може бути описана формулою 3.3.

$$\gamma(a_0, a_1, a_2, a_3) = (a_0 \oplus G_0, a_1 \oplus G_1, a_2 \oplus 2, a_3 \oplus G_3) \quad (3.2)$$

$$a \otimes b = a * b \bmod (2^{32} - 1) = \left(a * b \bmod 2^{32} + \left\lfloor \frac{a*b}{2^{32}} \right\rfloor\right) \bmod (2^{32} - 1) \quad (3.3)$$

Зауважимо, що γ є зворотним, але не є інволюцією. Кожне 32-бітове множення можна інтерпретувати як величезний 32 × 32-розрядний S-блок, оскільки один з операндів в множенні завжди фіксований. Для будь-якого Gi є дві фіксовані точки:0 ⊗ G_i = 0, і (2^{32} - 1) ⊗ G_i = 2^{32} - 1.

- - η: перетворення, залежне від даних, яке працює на двох із чотирьох входів слова (a_0, a_1, a_2, a_3) (Формула 3.4), де 'lsb' позначає найменш значущий (молодший) біт, а δ = $2aaaaaaa_x$; η – це інволюція та нелінійна операція.

$$\eta(a_0, a_1, a_2, a_3) = \left(a_0 \otimes (\mathrm{lsb}(a_0) * \delta), a_1, a_2, a_3 \otimes (1 \otimes \mathrm{lsb}(a_3) * \delta)\right) \quad (3.4)$$

- - θ: єдина дифузійна операція в MMB (Формула 3.5), де $a_i \in Z_{2^{32}}$, при 0 ≤ i ≤ 3. θ – інволюція.

$$\theta(a_0, a_1, a_2, a_3) = (a_3 \otimes a_0 \otimes a_1, a_0 \otimes a_1 \otimes a_2, a_1 \otimes a_2 \otimes a_3, a_2 \otimes a_3 \otimes a_0) \quad (3.5)$$

Є дві пари операцій, які можуть бути взаємозамінними: (θ, σ[k^j]) і (η, σ[k^j]). У кожному випадку ключ k^j перетворюється в еквівалентний ключ θ(k^j) абоη(k^j) відповідно.

J-тий (повний) раунд трасформації ММВ можна представити формулою 3.6.

$$\rho[k^j](X) = \theta \quad \eta \quad \gamma \quad \sigma[k^j](X) = \theta\left(\eta\left(\gamma\left(\sigma[k^j](X)\right)\right)\right) \quad (3.6)$$

Повний процес шифрування відкритого тексту P алгоритмом ММВ представлений в формулі 3.7.

$$\text{ММВ}(P) = \sigma[k^6] \quad \rho[k^5] \quad \rho[k^4] \quad \rho[k^3] \quad \rho[k^2] \quad \rho[k^1] \quad \sigma[k^0](P) \quad (3.7)$$

3.4 Методологія генерації ключів з використанням ГА

Початкова популяція хромосом є набором випадкових шістнадцятковихчисел. Ця початкова популяція має довжину в 128 бітів, аналогічно довжині ключа, що використовується у алгоритмі ММВ. Після цього створюється *n*поколінь. Усі ці особи потім обробляються фітнес-функцією. Особи, що мають максимальне обчислене значення відбираються для подальшого процесу. Після цього обираються дві найкращі особи, на них виконується операція кросинговеру на позиції, що визначається випадковим значенням. Після проведення схрещування отримуємо два нащадки обраних хромосом. Тепер знову фітнес-функція застосовується до отриманих нащадків і якщо їх значення краще ніж у батьківських хромосом, то батьківські хромосоми замінюються на нові, інакше нічого не змінюється. Далі вихідний результат попереднього етапу використовується як вхідні дані

для операції мутації. Після мутації ми отримуємо ключ, який буде використовуватися у процесі шифрування. Процес генерації ключа має декілька кроків.

Розширення початкової популяції.

Традиційні генетичні алгоритми зберігають інформацію в наборах генотипів. Кожен генотип є бітовим рядком. На базі згенерованого набору під-ключів створюється уся початкова популяція. Кожен генотип представляє один ключ.

Операція кросинговеру – це, по суті, основний інструмент генетичного алгоритму. Вона поєднує собою характеристики двох існуючих хромосом для створення нових нащадків. Після цього обрані хромосоми використовуються для створення нової популяції аналогічного розміру, тобто кожна пара хромосом продукує дві нові хромосоми.

Вхідними даними для наступного кроку, диверсифікації, буде початкова популяція, створена на цьому етапі. Спочатку відбираються два генотипи для операції схрещування. Після цього оператор кросинговеру проводиться над двома батьківськими генотипами і створює два нових генотипи нащадків. Єдина точка схрещування встановлюється після випадкової позиції в генотипах батьків. Ця операція повторюється для всієї популяції. Нові створені генотипи додаються до початкової популяції. Ця процедура повторюється до тих пір, поки не буде досягнуто остаточного розміру популяції (наприклад, 80 генотипів).

Диверсифікація популяції.

Цей крок включає у себе виконання власне генетичного алгоритму.

1. Створення початкової популяції.

Початкова популяція хромосом отримується із результатів попереднього кроку.

2. Обчислення фітнес-функції.

Визначається значення допасованості для кожної особи.Значення фітнес-функції розраховується на основі повторюваності символів, які

повторюються із максимальною частотою. Фітнес-функцію можна описати наступним чином:$F = n + (P/m)$.

Де F –значення допасованості,

n – загальне число символів, задіяних у формації ключа,

m – відсоток символу, що зустрічається найчастіше,

P – бажаний відсоток кожного символу.

3. Кросинговер.

На двох випадково обраних хромосомах проводиться кросинговер в одній позиції, що обирається на основі випадкового значення. Після операції кросинговеру отримуємо два нові нащадки, генеровані з батьківських хромосом.

4. Перевірка фітнес-функції.

Проводиться обчислення фітнес-функції для нових хромосом і якщо вони мають кращий результат то попередні замінюються на нащадків.

5. Мутація.

Мутація відбувається у випадково обраній хромосомі і розраховується її нове значення фітнес-функції.

6. Перевірка фітнес-функції.

Знову допасованість перевіряється для усієї популяції конкретного покоління і найбільше значення передається до початкової популяції наступного покоління.

Повний процес виконується певну кількість разів, зі строгим обмеженням у 100 поколінь і умовним обмеженням у 5 поколінь без збільшення максимального значення допасованості. Після досягнення бажаної умови популяція із максимальним значенням фітнес-функції обирається ключем для подальшого шифрування. Рис 3.2 ілюструє процес генерації ключа.

Рисунок 3.2 – процес генерації ключа використовуючи ГА

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ

4.1 Тестування програмного забезпечення

Скомпільований виконуючий файл займає 25 кілобайт.

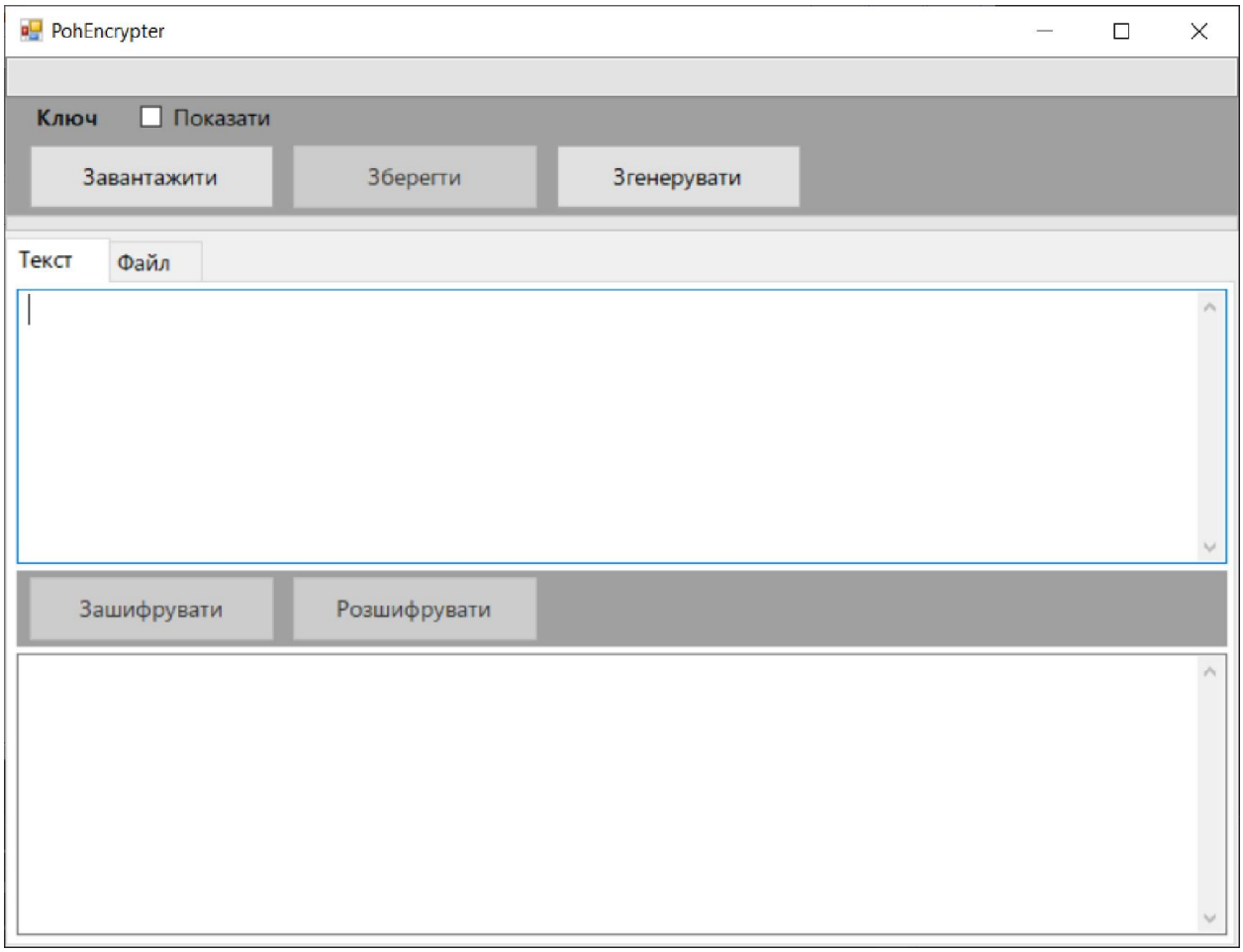

Рисунок 4.1 – Інтерфейс програми при запуску

Додаток працює в двох режимах: шифрування тексту, що відображується за замовчуванням при запуску програми (Рис. 4.1), та шифрування файлів (Рис. 4.2).

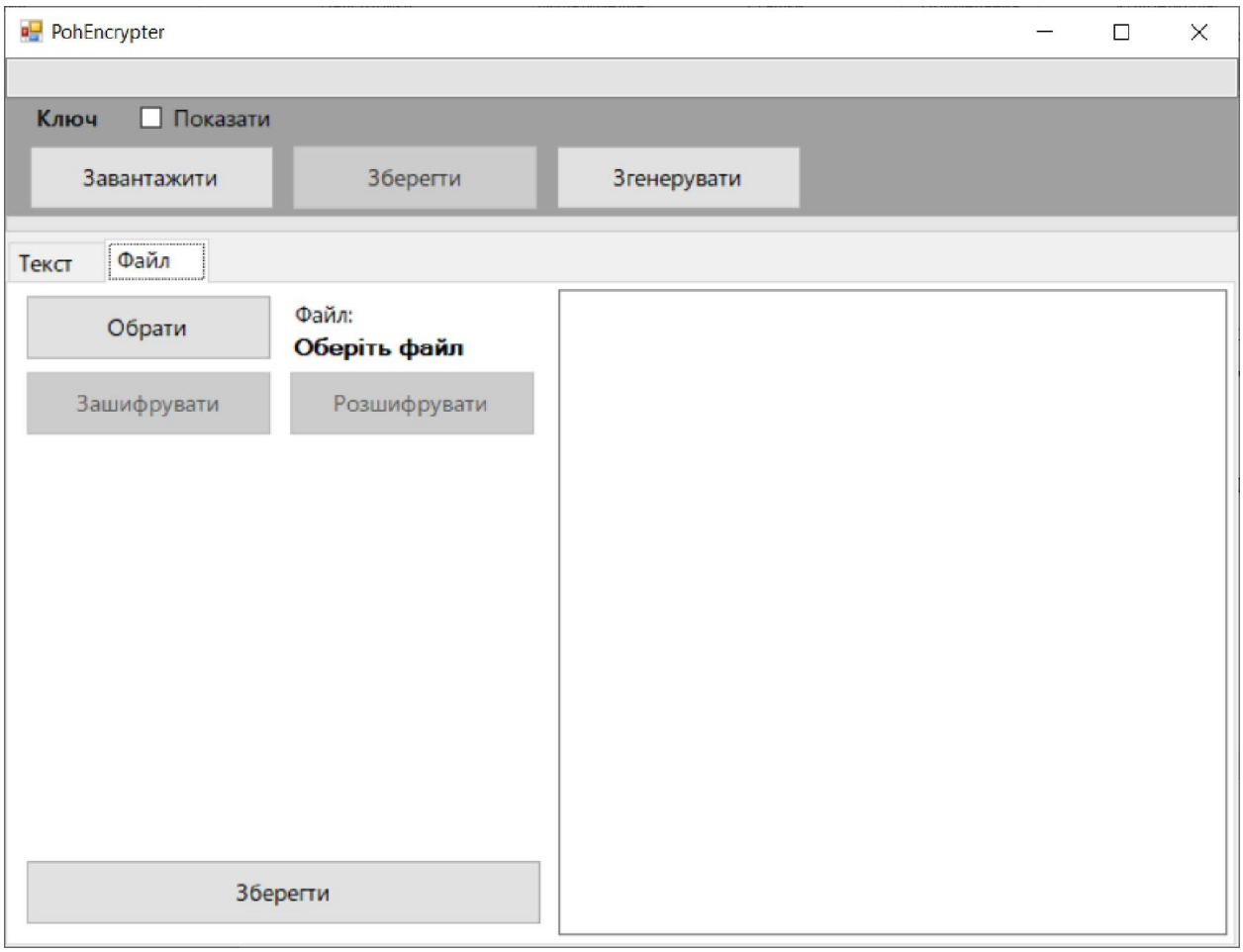

Рисунок 4.3 – Режим шифрування файлів

Для початку роботи необхідно завантажити існуючий (Рис. 4.3) або згенерувати новий файл криптографічного ключа. Можливо переключити відображення вмісту ключа, що за замовчуванням приховане (Рис. 4.4).Звернемо увагу на те, як при генерації ключа кнопки для шифрування та дешифрування стають активними, що дозволяє подальшу роботу з системою.

Ключ рекомендується зберегти окремим файлом для подальшого використання у процесі дешифровки у форматі *.pkey, адже при втраті ключа зашифрований текст буде практично неможливо розшифрувати.

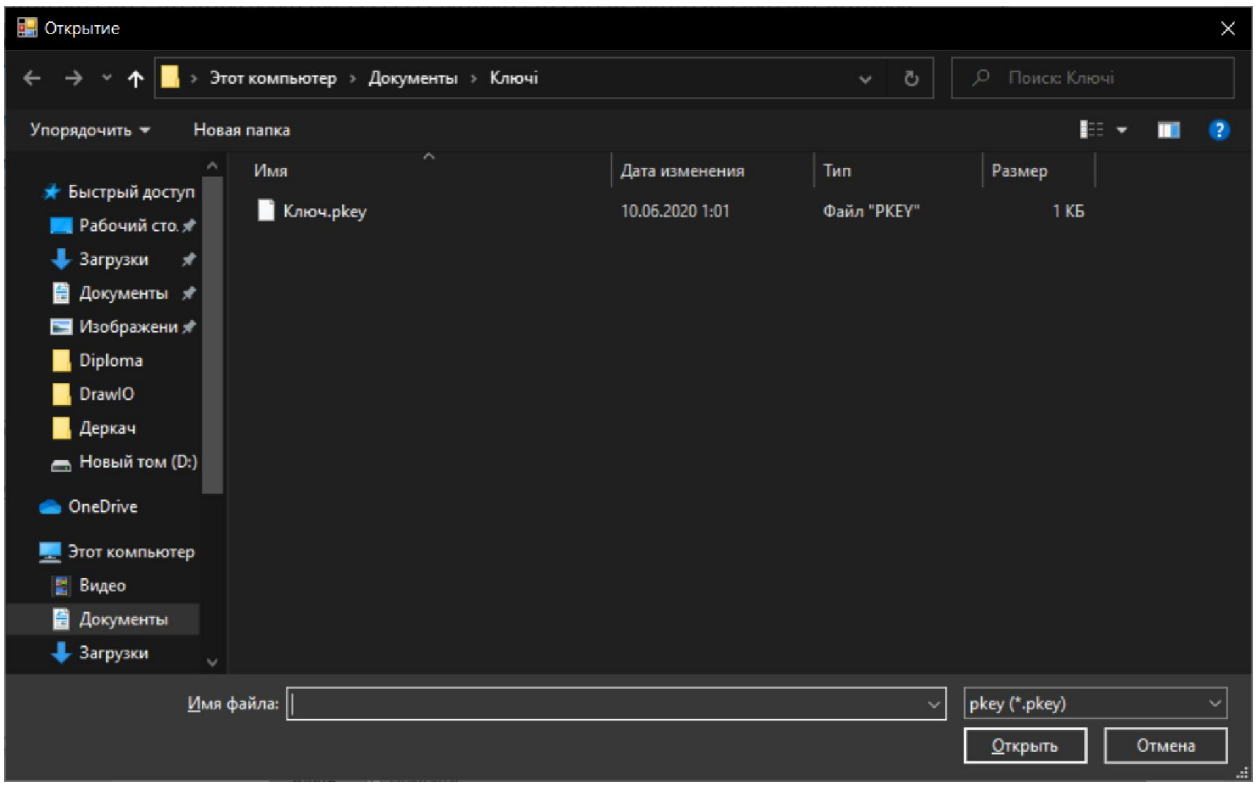

Рисунок 4.3 – Діалогове вікно вибору існуючого ключа зі сховища ПК

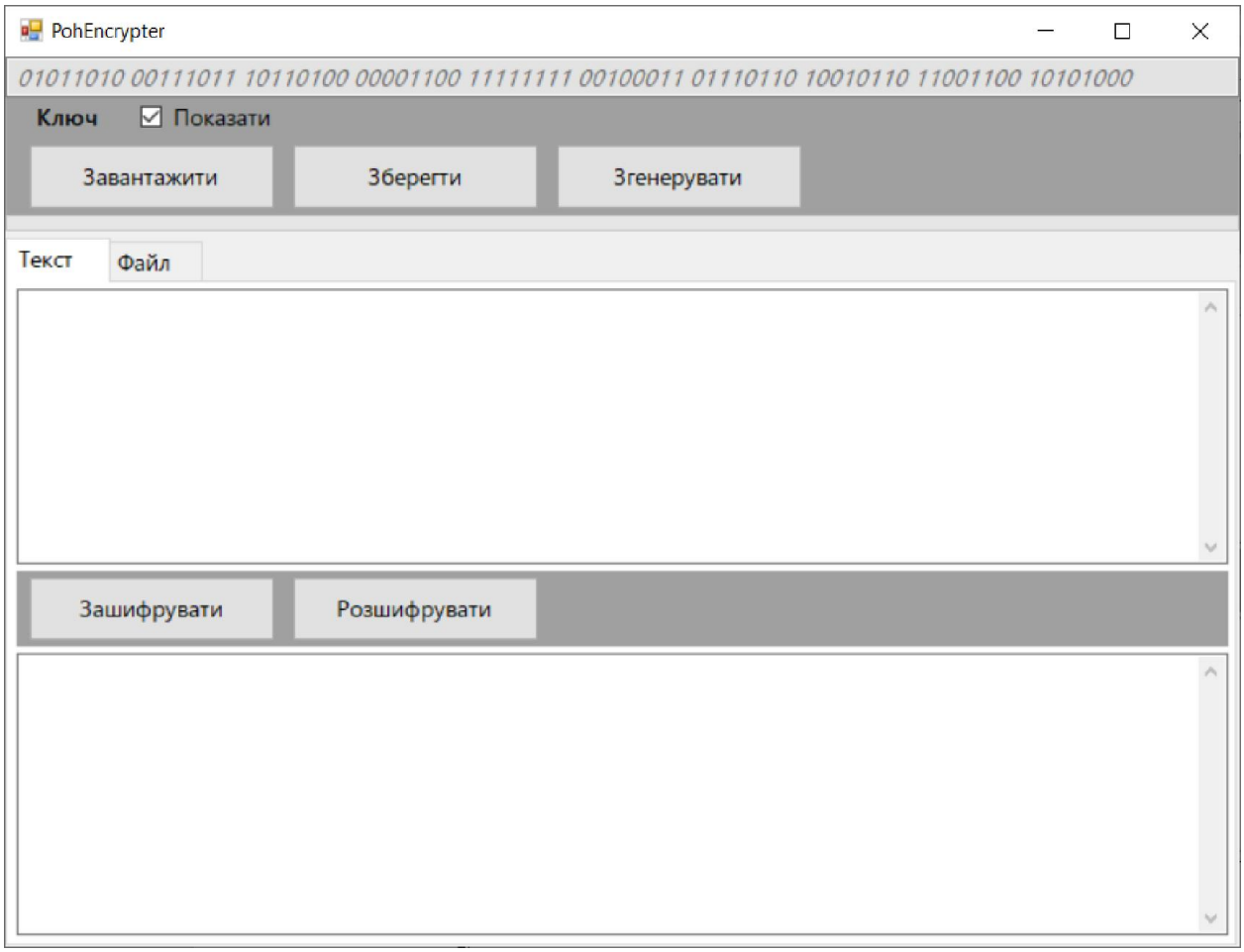

Рисунок 4.4 – Вид вікна зі згенерованим ключем

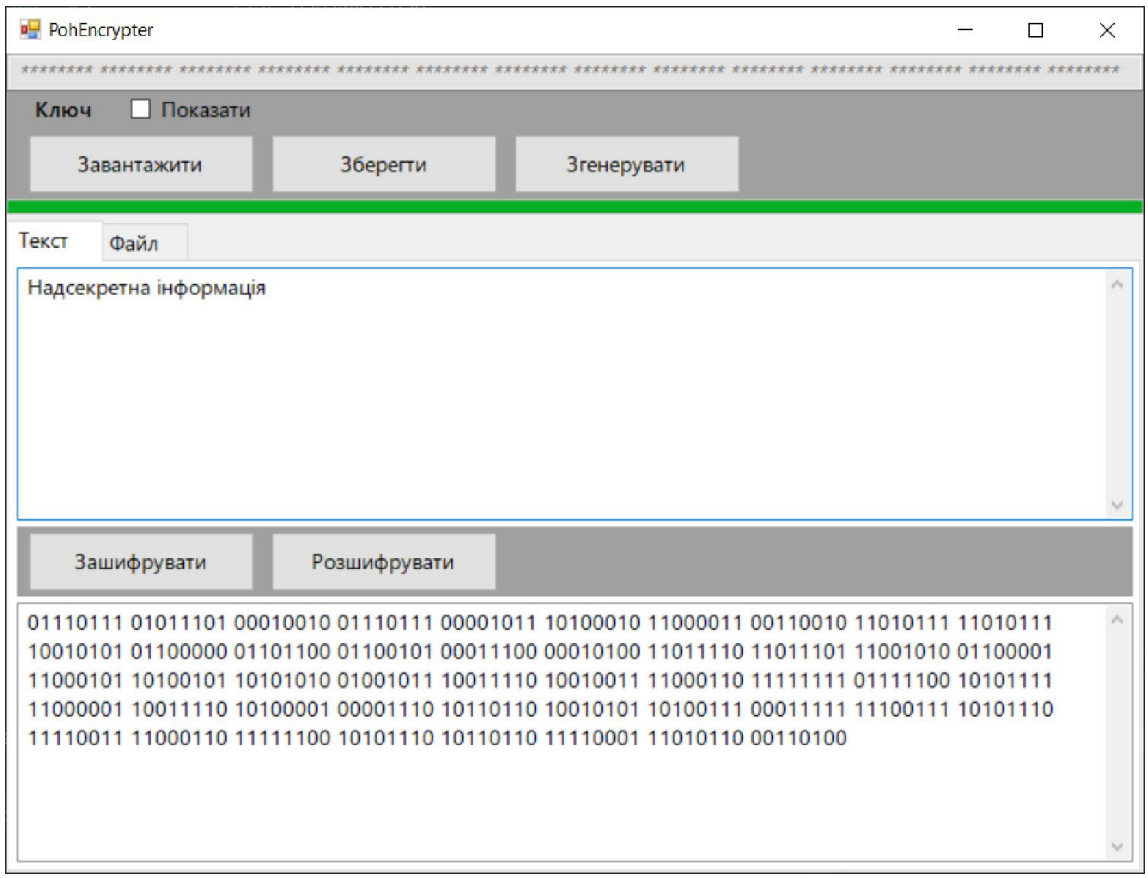

Рисунок 4.5 – Шифрування текстового повідомлення

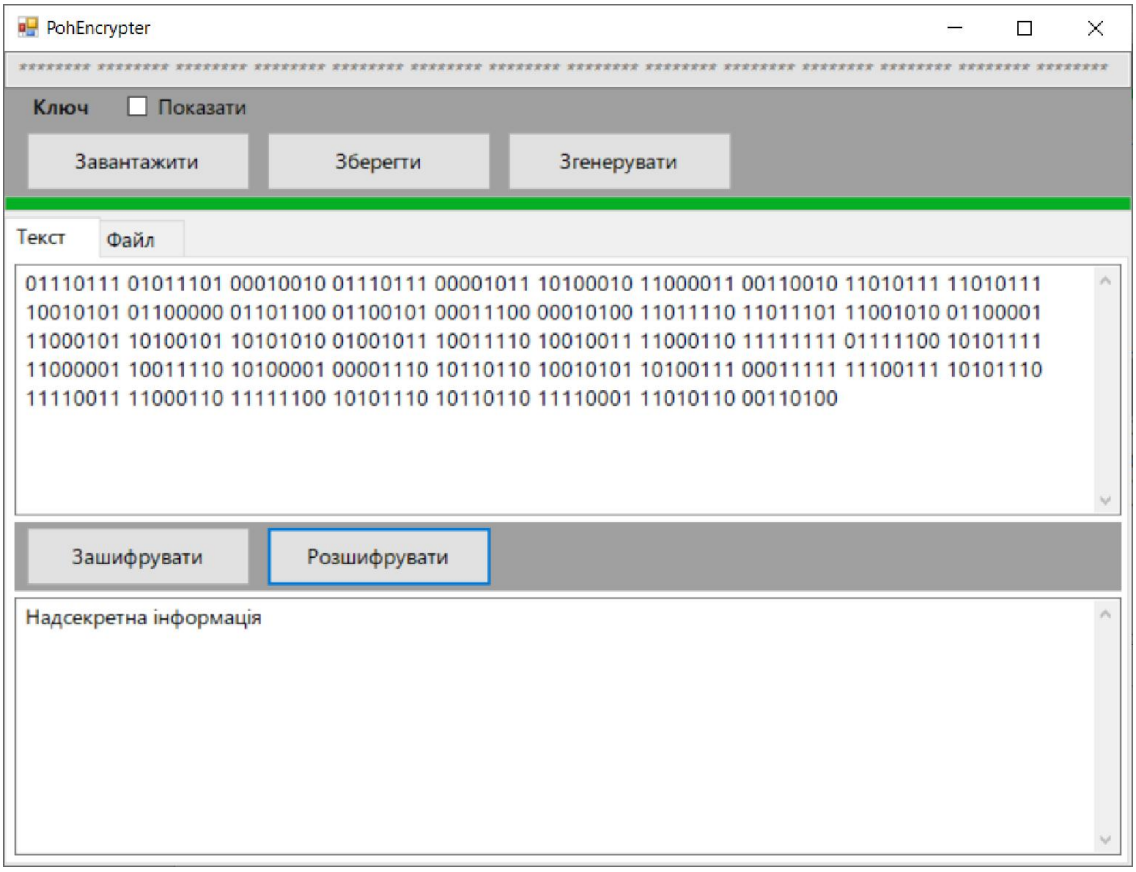

Рисунок 4.6 – Розшифровка текстового повідомлення

Для роботи з файлами необхідно спочатку також обрати з локального сховища власне файл, який необхідно зашифрувати. Файл може бути будь-якого формату. Після цього на відповідній вкладці стають активними кнопки для шифрування та розшифрування.

Кнопка «Зберегти» дозволяє зберегти зашифрований бітовий рядок як бітовий файл на ПК в довільному форматі.

Аналогічно для дешифровки в додаток необхідно завантажити файл та ключ, що був використаний в процесі шифрування. При успішному проведенні всіх маніпуляцій, отримуємо вихідний відкритий файл.

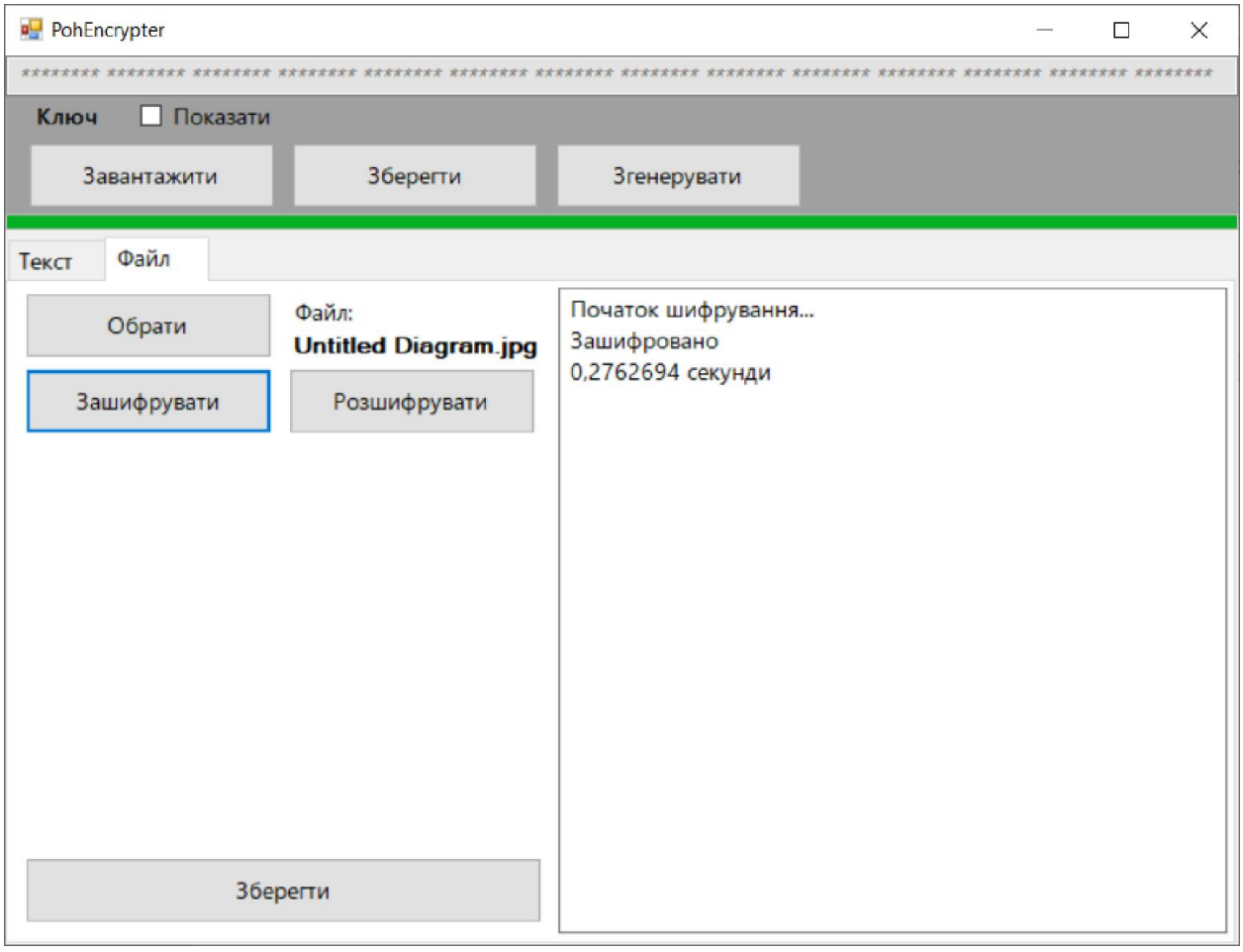

Рисунок 4.7 – Екран режиму шифрування файлів після завершення операції

4.2 Верифікація модуля штучного інтелекту

Для верифікації запропонованого алгоритму генерації криптографічних ключів за допомогою генетичного алгоритму було створено початкову

популяцію ключів. Кожна хромосома представляє одну конфігурацію набору під-ключів. На рис. 4.8a проілюстровано початкову популяцію, що складається із десяти генотипів. Кожен рядок представляє один генотип. Кожен генотип візуально закодований унікальним кольором для наочності. В наступному кроці над цією популяцією відбувається операція кросинговеру в одній випадковій позиції. Результат цієї операції зображено на рис. 4.8b. Кожний рядок тепер складається із генів двох батьківських хромосом, що зображені різними кольорами в кожному рядку.

Розмір популяції подвоюється в наступному кроці, тепер популяція має двадцять генотипів. Після цього над популяцією проводиться ще одна операція кросинговеру (рис. 4.8c).

Для досягнення фінального розміру популяції необхідно подвоїти розмір популяції ще двічі. Спочатку із 20 до 40 генотипів (рис. 4.8d), потім з 40 до 80 (рис. 4.8e).Після кожної зміни в популяції слідує операція кросинговеру над усіма хромосомами популяції.

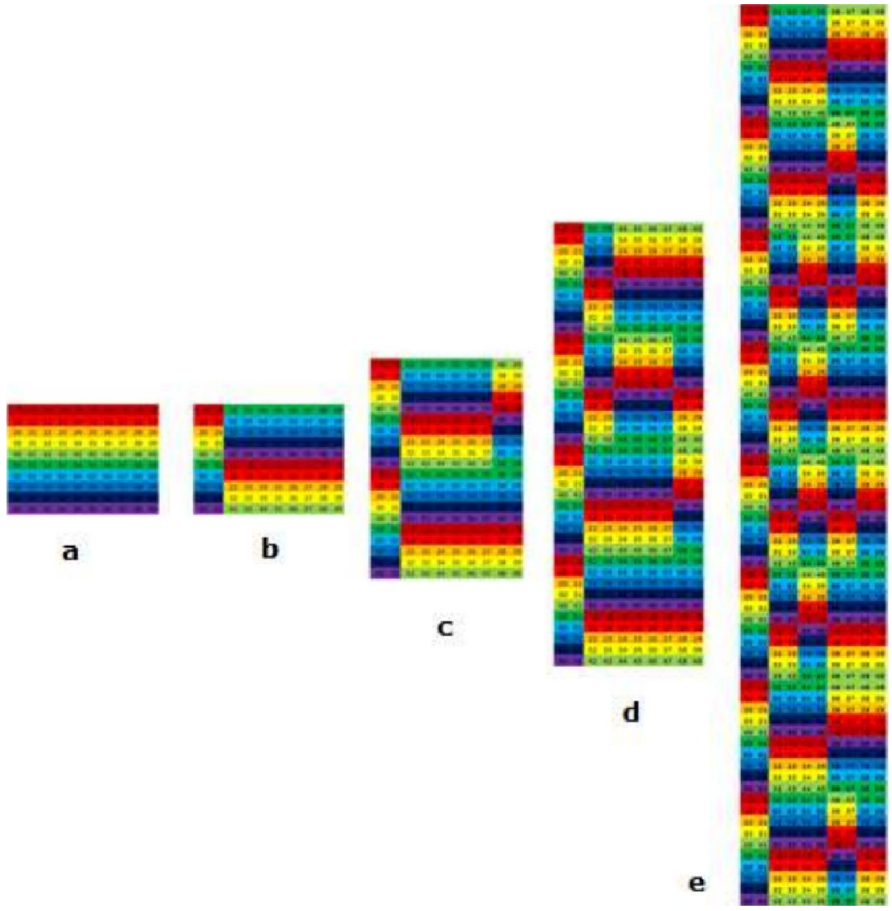

Рисунок 4.8 – Розширення початкової популяції

Диверсифікація популяції криптографічного ключа проілюстрована на рис. 4.9. Вхідна популяція після процесу розширення початкової вибірки запропонованим методом зображена на рис. 4.9a. Можна відслідкувати як саме генотипи із початкової популяції розподіляються по усій популяції для створення необхідної різноманітності.

Популяція криптографічних ключів для системи після 5 поколінь генетичного алгоритму показана на рис. 4.9b, після 10 поколінь – рис 4.9c, 15 поколінь – 4.9d, а популяція після 20 поколінь генетичного алгоритму зображена на рис. 4.9e.

Завдяки використанню різних кольорів для розрізнення різних генотипів можна спостерігати підходящий рівень розмаїття криптографічних ключів для алгоритму після проходження 20 поколінь ГА. При подальшій еволюції значення фітнес-функції зазвичай значно не зростає, тому ГА автоматично зупиняється в області 20-25 поколінь. Якщо обмежитися виключно строгим обмеженням на кількість поколінь, то в діапазоні 50-100 поколінь рівень розмаїття не збільшується пропорційно затраченому часу. Підходящий результат та оптимальний час для отримання прийнятного результату досягається в середньому за 20-25 поколінь запропонованим методом (рис. 4.10).

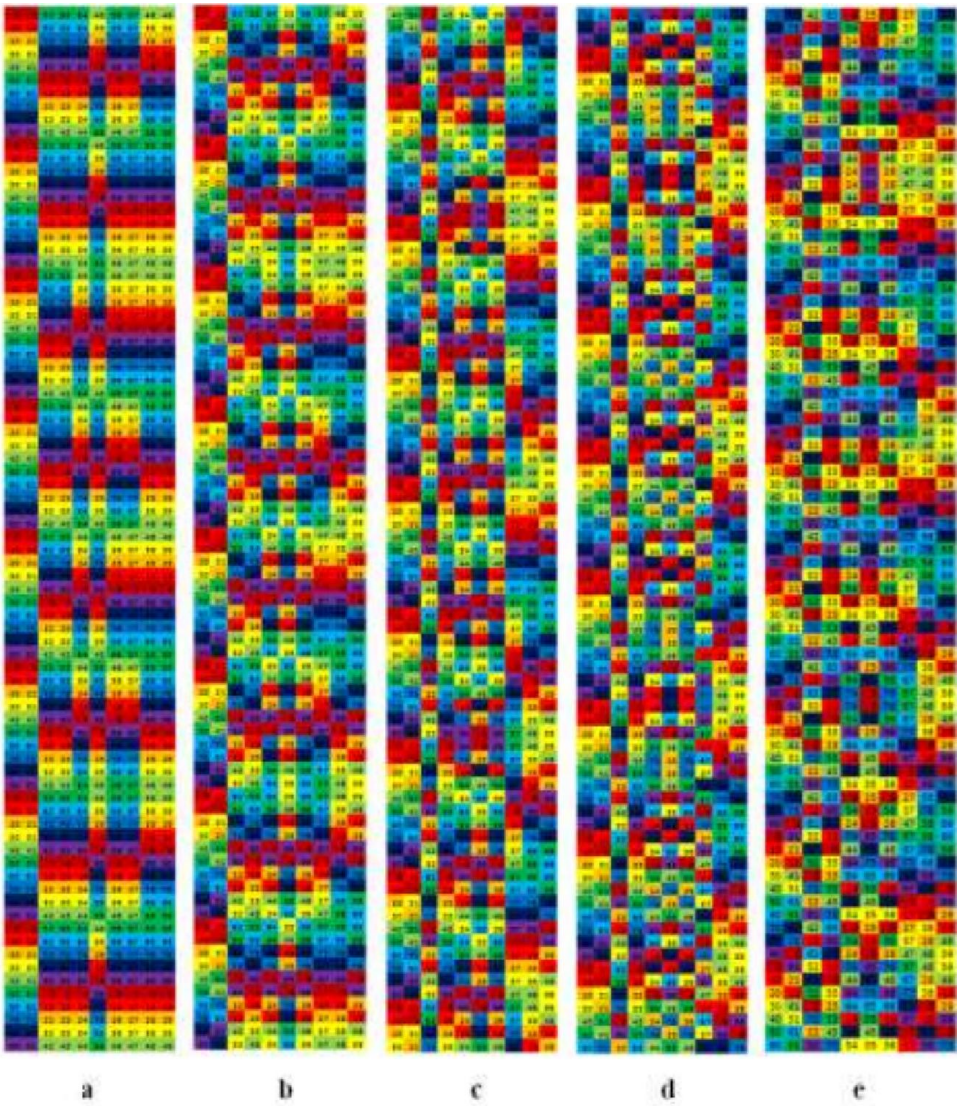

Рисунок 4.9 – диверсифікація популяції.

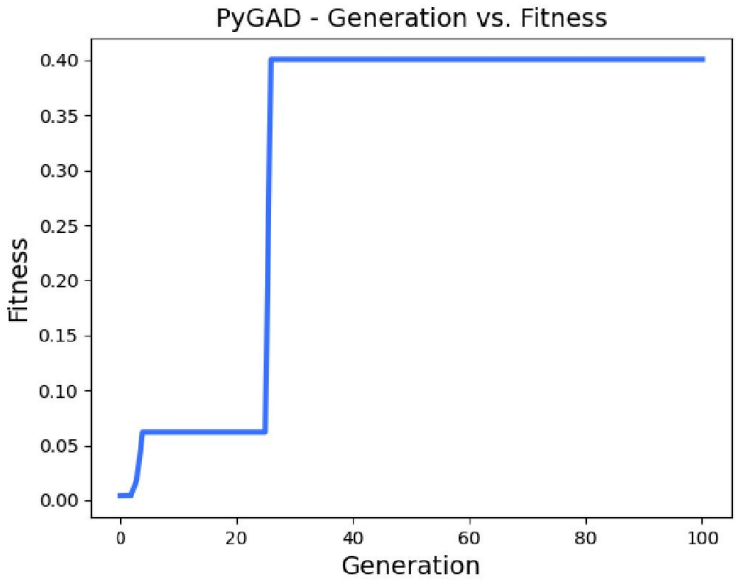

Рисунок 4.10 – графік зміни значення фітнес-функції.

4.3 Впровадження результатів ДР

Дипломна робота на тему «Програмна реалізація дешифратора на основі алгоритму MMB» відповідає замовленню підприємства:

1. проведена оцінка стану систем захисту інформації:
   - проведено аналіз мережі;
   - проведено аналіз програмного забезпечення, що встановлене на робочі комп’ютери;
   - проведено аналіз фізичних засобів захисту інформації;
   - проведено аналіз роботи з конфіденційною інформацією;
2. створено системуінформаційного захисту, програму передачі даних та повідомлень з використанням алгоритму шифрування MMB на базі операційної системи Windows із імплементацією модуля штучного інтелекту, а саме системи генерації криптографічних ключів на базі генетичного алгоритму.

Проведена робота має практичну значимість та планується до використання у ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ.

Повний текст довідки про впровадження вказаний в додатку А.

ВИСНОВКИ

В дипломній роботі проведено аналіз стану та розроблено комплексні методи захисту інформації на підприємстві ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ.

Виходячи з аналізу проблеми сформовано завдання та критерії проведення комплексного аналізу.

В ході роботи було надано рекомендації з захисту інформаційної системи підприємства, а саме захист комп'ютерної мережі, шляхом зміни та оптимізації топології комп'ютерної мережі, рекомендації щодо забезпечення захисту сервера від несанкціонованого доступу, було рекомендовано оновити операційну систему на сервері, встановити антивірус MicrosoftEssential, задля забезпечення криптографічного захисту було розроблено програмне забезпечення на основі алгоритму шифрування ММВ із імплементацією модуля генерації криптографічних ключів на основі штучного інтелекту. По результатам роботи отримано довідку про впровадження на підприємстві ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ (Додаток А).

Таким чином, проведення та впровадження систем комплексних заходів на підприємстві забезпечує безпеку інформації на підприємстві.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ситник Г. Інформаційна безпека [Електронний ресурс] / Галина Ситник // СерчІнформ – Режим доступу до ресурсу: https://searchinform.ru/informatsionnaya-bezopasnost/.

2. Дослідженння IBM і Ponemon Institute [Електронний ресурс] // IBM – Режим доступу до ресурсу: https://www-03.ibm.com/press/ru/ru/pressrelease/50084.wss.Шквір В. Д.

3. Наймасштабніші комп'ютерні атаки у світі [Електронний ресурс] // ТАСС. – 2017. – Режим доступу до ресурсу: http://tass.ru/info/4248876.

4. Економічна енциклопедія: У трьох томах. Т. 1. / Редкол.: …С. В. Мочерний (відп. ред.) та ін. – К.: Видавничий центр “Академія”, 2000. – 864 с.

5. Закон України. Про захист інформації в інформаційно-телекомунікаційних системах // http://zakon.rada.gov.ua/laws/show/80/94-вр

6. Явтуховский Е. Ю., Кошелев С. О. Современные технологии защиты информации в распределённых системах // Молодой ученый. — 2016. — №28. — С. 43-45. — URL https://moluch.ru/archive/132/36986/ (дата обращения: 12.06.2018).

7. Комп'ютерні віруси. Визначення, класифікація і способи захисту [Електронний ресурс] – Режим доступу до ресурсу: http://iteranet.ru/it-novosti/2013/08/10/kompyuternye-virusy-opredelenie-klassifikaciya-i-sposoby-zashhity/.

8. Brandon G. 37 Shocking Computer Virus Statistics [Електронний ресурс] / Gaille Brandon. – 2017. – Режим доступу до ресурсу: https://brandongaille.com/36-shocking-computer-virus-statistics/.

9. [Copeland, B. J., Artificial intelligence. Encyclopedia Britannica, 2020. https://www.britannica.com/technology/artificial-intelligence

10. Turing, A. M., Computing Machinery and Intelligence. Mind, 1950, 49: p. 433-460 [Online] Available from:

https://www.csee.umbc.edu/courses/471/papers/turing.pdf [Accessed 27 April 2020]

11. Wikibooks, Artificial Neural Networks/Neural Network Basics, 2020. https://en.wikibooks.org/wiki/Artificial_Neural_Networks/Neural_Network_Basic

12. Kostadinov, S., Understanding the Back-Propagation Algorithm, Towards Data Science, 2019. https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

13. Joshi, P., Artificial Intelligence with Python: A Comprehensive Guide to Building Intelligent Apps for Python Beginners and Developers, Packet Publishing Limited, 2017. ISBN: 978-1-78646-439- 2. https://www.amazon.com/Artificial-Intelligence-Python-ComprehensiveIntelligent/dp/178646439X

14. Turner, M. J., Blackledge, J. M. and Andrews, P., Fractal Geometry in Digital Imaging, Academic Press, 1998. ISBN: 0-12-703970-8, 1998.

15. Fractal Geometry: Mathematical Methods, Algorithms and Applications (Ed. J. M. Blackledge, A. K.

16. Evans and M. J. Turner), Woodhead Publishing: Series in Mathematics and Applications, 2002. ISBN: 190427500.

17. Zhang, W., Itoh, K., Tanida, J. and Ichioka, Y., Parallel Distributed Processing Model with Local Space-invariant Interconnections and its Optical Architecture, Applied Optics, 1990, 29(32), p. 4790–4797. https://drive.google.com/file/d/0B65v6Wo67Tk5ODRzZmhSR29VeDg/view

18. Blackledge, J. M., Digital Image Processing, Woodhead Publishing Series in Electronic and Optical Materials, 2005, ISBN-13: 978-1898563495. https://arrow.tudublin.ie/engschelebk/3/

19. MathWorks, Introducing Deep Learning with MATLAB, 2020. https://uk.mathworks.com/campaigns/offers/deep-learning-with-matlab.htm

20. Maghrebi, H., Portigliatti, T., Prouf, E., Breaking Cryptographic Implementations Using Deep Learning Techniques, Security, Privacy, and Applied

Cryptography Engineering (SPACE), 6th International Conference, 2016. [Online] Available from: https://eprint.iacr.org/2016/921.pd

21. Bezobrazov, S., Blackledge, J. M. and Tobin, P., Cryptography using Artificial Intelligence, The International Joint Conference on Neural Networks (IJCNN2015), Killarney, Ireland, 12-17 July, 2015.

22. Asiru, O. F., Blackledge, J. M. and Dlamini, M. T., Application of Artificial Intelligence for Detecting Computing Derived Viruses, 16th European Conference on Cyber Warfare and Security (ECCWS 2017), 2017, University College Dublin, Dublin June 29-30, p. 647-655.

23. Anil Kumar and M. K. Ghose, Overview of Information Security Using Genetic Algorithm and Chaos, Information Security Journal: A Globe Perspective, 18:306-315, 2009.

24. Baheti, A., L. Singh and A.U. Khan, 2014. Proposed Method for Multimedia Data Security Using Cyclic Elliptic Curve, Chaotic System and Authentication Using Neural Network. Proceedings of the 4th International Conference on Communication Systems and Network Technologies (CSNT), April 7-9, 2014, IEEE, New York, USA., ISBN:978-1-4799-3070-5, pp: 664-668.

25. Постанова Кабінету міністрів України [Електронний ресурс]. – 2006. – Режим доступу до ресурсу: http://www.ukrbook.net/zakony/Sfera_inform/Pos_373.pdf.

26. Сушко С. А. Практическая Криптология лекция 9 [Електронний ресурс] / С. А. Сушко – Режим доступу до ресурсу: http://bit.nmu.org.ua/ua/student/metod/cryptology/%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F%209.pdf.

27. Daemen, J., Govaerts, R., Vandewalle, J.: BlockCiphersBasedonModularMultiplication. In: Wolfowicz, W. (ed.) Proceedings of 3rd Symposium on State and Progress of Research in Cryptography, Fondazione Ugo Bordoni, pp. 80–89 (1993).

28. Lai, X.: On the Design and Security of Block Ciphers. In: Massey, J.L. (ed.) ETH Series in Information Processing, vol. 1. Hartung-Gorre Verlag, Konstanz (1995)

29. A. Joux, Algorithmic Crypanalysis, United States: CRC Press, 2009.

30. Daemen, J.: Cipher and Hash Function Design – Strategies based on Linear and Differential Cryptanalysis. PhD Thesis, Dept. Elektrotechniek, Katholieke Universiteit Leuven, Belgium (1995)

31. Лекция 3 технологии программирования [Електронний ресурс] // Белорусско-Российский университет – Режим доступу до ресурсу: https://studfiles.net/preview/5618019/page:3/.

ДОДАТОК А
ДОВІДКА ПРО ВПРОВАДЖЕННЯ

ФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ
38610
Полтавська обл., Котелевський район, село Більськ

АКТ

У ході виконання кваліфікаційної роботи було проаналізовано стан інформаційної безпеки на підприємствіФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧта створено програмний додаток для захисту та передачі даних по мережі підприємства з використанням алгоритму шифрування ММВ та імплементацією модуля для генерації криптографічних ключів на основі штучного інтелекту.

Робота виконувалась по замовленнюФОП СТОВПІВСЬКИЙ ОЛЕКСАНДР АНАТОЛІЙОВИЧ, з подальшим можливим працевлаштуванням на посаді спеціаліста з інформаційної безпеки.

Організація задоволена співпрацею зі студентом Походуном В.Р, було виявлено значну кількість недоліків в сфері захисту інформації, які в подальшому могли б привести до значних фінансових втрат.

Вважаємо, що студент Походун Валентин Русланович. є кваліфікованим спеціалістом і заслуговує на відмінну оцінку. Програмний додаток повністю відповідає всім вимогам, поставленим на етапі постановки задачі, та готовий до експлуатації.

ДОДАТОК В
ВИХІДНИЙ КОД ПРОГРАМИ

1. Form1.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Diploma
{
    public partial class Form1 : Form
    {
        public class SubKeyClass
        {
            public byte[] bytes;

            //generates random subkey
            public SubKeyClass(Random rnd)
            {
                bytes = new byte[4];
                rnd.NextBytes(bytes);
            }

            public SubKeyClass(byte[] b, int i)
            {
                bytes = new byte[4];
                for (int j = 0; j < 4; j++)
                {
                    bytes[j] = b[i * 4 + j];
                }
            }

            public string ByteToString()
            {
                string str = "";
                foreach (byte Element in bytes)
                {
                    str += Convert.ToString(Element, 2).PadLeft(8, '0') + " ";
                }
                return str;
            }
        }

        public class KeyClass
        {
            public SubKeyClass[] SubKeys;

            //generates random key
            public KeyClass()
            {
                SubKeys = new SubKeyClass[4];
                Random rnd = new Random();
                for (int i = 0; i < 4; i++)
                {
                    SubKeys[i] = new SubKeyClass(rnd);
                }
            }

            public KeyClass(byte[] b)
            {
                SubKeys = new SubKeyClass[4];
```

```
        for (int i = 0; i < 4; i++)
        {
            SubKeys[i] = new SubKeyClass(b, i);
        }
    }

    public string ByteToString()
    {
        string str = "";
        foreach (SubKeyClass sk in SubKeys)
        {
            str += sk.ByteToString();
        }
        return str;
    }

    public byte[] toByteArray()
    {
        byte[] array = new byte[16];
        int counter = 0;
        for (int i = 0; i < 4; i++)
        {
            for (int j = 0; j < 4; j++)
            {
                array[counter] = this.SubKeys[i].bytes[j];
                counter++;
            }
        }
        return array;
    }
}

public class SubTextBlockClass
{
    public byte[] bytes;

    public SubTextBlockClass(int n, int m, byte[] text)
    {
        bytes = new byte[4];
        for (int i = 0; i < 4; i++)
        {
            try
            {
                bytes[i] = text[n + i + 4 * m];
            }
            catch
            {
                bytes[i] = 0b00000000;
            }
        }
    }
}

public class TextBlockClass
{
    public SubTextBlockClass[] SubTextBlocks;

    public TextBlockClass(int n, byte[] text)
    {
        SubTextBlocks = new SubTextBlockClass[4];
        for (int i = 0; i < 4; i++)
        {
            SubTextBlocks[i] = new SubTextBlockClass(n, i, text);
        }
    }
}

public class PlainTextClass
{
    public byte[] Text;

    public PlainTextClass(byte[] file)
    {
        Text = file;
    }

    public void Change(byte[] b, int n)
```

```
            {
                for (int i = 0; i < b.Length; i++)
                {
                    try
                    {
                        Text[n + i] = b[i];
                    }
                    catch { }
                }
            }

            public void Change(TextBlockClass X, int n)
            {
                for (int i = 0; i < 4; i++)
                {
                    this.Change(X.SubTextBlocks[i].bytes, n + i * 4);
                }
            }
        }

        static PlainTextClass PLAINTEXT;
        static KeyClass Key;
        static byte[] C = new byte[] { 0x2a, 0xaa, 0xaa, 0xaa }; //"2aaaaaaa"
        static byte[] c0 = new byte[] { 0x02, 0x5f, 0x1c, 0xdb }; //"025f1cdb"
        static byte[] c1 = new byte[] { 0x04, 0xbe, 0x39, 0xb6 }; //"2 * c0"
        static byte[] c2 = new byte[] { 0x12, 0xf8, 0xe6, 0xd8 }; //"2^3 * c0"
        static byte[] c3 = new byte[] { 0x2f, 0x8e, 0x6d, 0x81 }; //"2^7 * c0"
        static byte[] c0min1 = new byte[] { 0x0d, 0xad, 0x46, 0x94 }; //"0dad4694"
        static byte[] c1min1 = new byte[] { 0x06, 0xD6, 0xA3, 0x4A }; //"1B5A8D28"
        static byte[] c2min1 = new byte[] { 0x81, 0xB5, 0xA8, 0xD2 }; //"6D6A34A0"
        static byte[] c3min1 = new byte[] { 0x28, 0x1B, 0x5A, 0x8D }; //"6D6A34A00"
        static uint mod_const = 4294967295;

        public Form1() { InitializeComponent(); }

        private void Form1_Load(object sender, EventArgs e) { }

        private void openFile()
        {
            try
            {
                OpenFileDialog OFD = new OpenFileDialog();
                OFD.Filter = "All files(*.*)|*.*";
                OFD.ShowDialog();
                if (OFD.FileName == "") { }
                else
                {
                    label3.Text = OFD.SafeFileName;
                    PLAINTEXT = new PlainTextClass(File.ReadAllBytes(OFD.FileName));
                }
            }
            catch { }
        }

        private void openKey()
        {
            try
            {
                OpenFileDialog OFD = new OpenFileDialog();
                OFD.Filter = "pkey (*.pkey)|*.pkey|All files(*.*)|*.*";
                OFD.ShowDialog();
                if (OFD.FileName == "") { }
                else
                {
                    try
                    {
                        Key = new KeyClass(File.ReadAllBytes(OFD.FileName));
                        if (checkBox1.Checked)
                            textBox1.Text = Key.ByteToString();
                        else  textBox1.Text  =  "********  ********  ********  ********  ********  ********
******** ******** ******** ******** ******** ******** ******** ******** ******** ********";
                    }
                    catch { textBox1.Text = "Error generating Key, try again"; button4.Enabled = false; }
                }
            }
            catch { }
        }
```

```
        private void saveFile(byte[] array)
        {
            try
            {
                SaveFileDialog SFD = new SaveFileDialog();
                SFD.Filter = "All files(*.*)|*.*";

                if (SFD.ShowDialog() == DialogResult.OK && SFD.FileName.Length > 0)
                {
                    File.WriteAllBytes(SFD.FileName, array);
                }
            }
            catch { }
        }

        private void saveKey(KeyClass K)
        {
            try
            {
                SaveFileDialog SFD = new SaveFileDialog();
                SFD.Filter = "pkey (*.pkey)|*.pkey|All files(*.*)|*.*";
                if (SFD.ShowDialog() == DialogResult.OK && SFD.FileName.Length > 0)
                {
                    File.WriteAllBytes(SFD.FileName, K.toByteArray());
                }
            }
            catch { }
        }

        private void encrypt(PlainTextClass P, KeyClass K)
        {
            int newLength, diff;
            newLength = P.Text.Length / 16 * 16 + 16;
            diff = newLength - P.Text.Length;
            Array.Resize(ref P.Text, newLength);
            P.Text[P.Text.Length - 1] = Convert.ToByte(diff);

            progressBar1.Value = 1;
            progressBar1.Maximum = P.Text.Length / 16 * 2 * 6;

            int addition = 0;
            for (int iterations = 0; iterations < 6; iterations++)
            {
                for (int i = 0; i < P.Text.Length; i += 16)
                {
                    TextBlockClass X = new TextBlockClass(i, P.Text);
                    for (int a = 0; a < 4; a++)
                    {
                        X.SubTextBlocks[a].bytes = xor(X.SubTextBlocks[a].bytes, K.SubKeys[(a + addition)
% 4].bytes);
                    }
                    P.Change(X, i);
                    progressBar1.PerformStep();
                }
                addition++;
                if (addition == 3) { addition = 0; }


                for (int i = 0; i < P.Text.Length; i += 16)
                {
                    TextBlockClass X = new TextBlockClass(i, P.Text);
                    X = f(X);
                    P.Change(X, i);
                    progressBar1.PerformStep();
                }
            }
        }

        private void decrypt(PlainTextClass P, KeyClass K)
        {
            progressBar1.Value = 1;
            progressBar1.Maximum = P.Text.Length / 16 * 2 * 6;

            int addition = 2;
            for (int iterations = 0; iterations < 6; iterations++)
            {
```

```
                for (int i = 0; i < P.Text.Length; i += 16)
                {
                    TextBlockClass X = new TextBlockClass(i, P.Text);
                    KeyClass Key = new KeyClass();
                    X = f_reverse(X);
                    P.Change(X, i);
                    progressBar1.PerformStep();
                }

                for (int i = 0; i < P.Text.Length; i += 16)
                {
                    TextBlockClass X = new TextBlockClass(i, P.Text);
                    for (int a = 0; a < 4; a++)
                    {
                        X.SubTextBlocks[a].bytes = xor(X.SubTextBlocks[a].bytes, K.SubKeys[(a + addition)
% 4].bytes);
                    }

                    P.Change(X, i);
                    progressBar1.PerformStep();
                }
                addition--;
                if (addition == -1) { addition = 2; }
            }
            Array.Resize(ref P.Text, P.Text.Length - P.Text[P.Text.Length - 1]);
        }

        private string byteArrayToString(byte[] b)
        {
            string str = "";
            foreach (byte Element in b)
            {
                str += Convert.ToString(Element, 2).PadLeft(8, '0') + " ";
            }
            str += "\r\n";
            return str;
        }

        public TextBlockClass f(TextBlockClass X)
        {
            //1
            for (int i = 0; i < 4; i++)
            {
                switch (i)
                {
                    case 0:
                        X.SubTextBlocks[i].bytes = f1(X.SubTextBlocks[i].bytes, c0);
                        break;
                    case 1:
                        X.SubTextBlocks[i].bytes = f1(X.SubTextBlocks[i].bytes, c1);
                        break;
                    case 2:
                        X.SubTextBlocks[i].bytes = f1(X.SubTextBlocks[i].bytes, c2);
                        break;
                    case 3:
                        X.SubTextBlocks[i].bytes = f1(X.SubTextBlocks[i].bytes, c3);
                        break;
                }
            }
            //2
            for (int i = 0; i < 4; i++)
            {
                if (i == 0 || i == 3)
                {
                    X.SubTextBlocks[i] = f2(X.SubTextBlocks[i], i);
                }
            }
            //3
            for (int i = 0; i < 4; i++)
            {
                X.SubTextBlocks[i] = f3(X, i);
            }
            return X;
        }

        public byte[] f1(byte[] x, byte[] c)
        {
```

```
            if (x[0] == 0b11111111 && x[0] == x[1] && x[0] == x[2] && x[0] == x[3]) { }
            else
            {
                ulong x_int = byteArrayToInt(x);
                ulong c_int = byteArrayToInt(c);
                x = intToByteArray((uint)((x_int * c_int) % mod_const));
            }
            return x;
        }

        public SubTextBlockClass f2(SubTextBlockClass r, int n)
        {
            SubTextBlockClass x = r;
            if (n == 0)
            {
                if (lsb(x.bytes[3]))
                {
                    x.bytes = xor(x.bytes, C);
                }
            }
            if (n == 3)
            {
                if (!lsb(x.bytes[3]))
                {
                    x.bytes = xor(x.bytes, C);
                }
            }
            return x;
        }

        public SubTextBlockClass f3(TextBlockClass TextBlock, int n)
        {
            int i1, i2, i3;
            if (n == 0) { i1 = 3; i2 = 0; i3 = 1; }
            else if (n == 1) { i1 = 0; i2 = 1; i3 = 2; }
            else if (n == 2) { i1 = 1; i2 = 2; i3 = 3; }
            else if (n == 3) { i1 = 2; i2 = 3; i3 = 0; }
            else { i1 = 0; i2 = 0; i3 = 0; MessageBox.Show("ERROR", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error); }
            SubTextBlockClass res = new SubTextBlockClass(0, 0, xor(xor(TextBlock.SubTextBlocks[i1].bytes,
TextBlock.SubTextBlocks[i2].bytes), TextBlock.SubTextBlocks[i3].bytes));

            return res;
        }

        public TextBlockClass f_reverse(TextBlockClass X)
        {
            //1
            for (int i = 3; i >= 0; i--)
            {
                X.SubTextBlocks[i] = f3(X, i);
            }
            //2
            for (int i = 3; i >= 0; i--)
            {
                X.SubTextBlocks[i] = f2(X.SubTextBlocks[i], i);
            }
            //3
            for (int i = 3; i >= 0; i--)
            {
                switch (i)
                {
                    case 0:
                        X.SubTextBlocks[i].bytes = f1(X.SubTextBlocks[i].bytes, c0min1);
                        break;
                    case 1:
                        X.SubTextBlocks[i].bytes = f1(X.SubTextBlocks[i].bytes, c1min1);
                        break;
                    case 2:
                        X.SubTextBlocks[i].bytes = f1(X.SubTextBlocks[i].bytes, c2min1);
                        break;
                    case 3:
                        X.SubTextBlocks[i].bytes = f1(X.SubTextBlocks[i].bytes, c3min1);
                        break;
                }
            }
            return X;
```

```
        }

        //Returns last significant bit
        public bool lsb(byte b)
        {
            bool lastBit = (b & (1 << 0)) != 0;
            return lastBit;
        }

        public byte[] xor(byte[] x, byte[] b)
        {
            byte[] res = new byte[x.Length];
            for (int i = 0; i < 4; i++)
            {
                res[i] = (byte)(x[i] ^ b[i]);
            }
            return res;
        }

        private ulong byteArrayToInt(byte[] Bytes)
        {
            ulong res = 0;
            int counter = 0;
            for (int byteNum = Bytes.Length - 1; byteNum >= 0; byteNum--)
            {
                for (int bitNum = 0; bitNum < 8; bitNum++)
                {
                    bool bit = (Bytes[byteNum] & (1 << bitNum)) != 0;
                    if (bit)
                    {
                        res += 1 * (ulong)Math.Pow(2, counter);
                    }
                    counter++;
                }
            }
            return res;
        }

        private byte[] intToByteArray(uint n)
        {
            byte[] res = BitConverter.GetBytes(n);
            Array.Reverse(res);
            return res;
        }

        //Open File
        private void button1_Click_1(object sender, EventArgs e)
        {
            openFile();
            enableButtons();
        }

        //Generate Random Key
        private void button2_Click(object sender, EventArgs e)
        {
            Key = new KeyClass();
            if (checkBox1.Checked)
                textBox1.Text = Key.ByteToString();
            else textBox1.Text = "******** ******** ******** ******** ******** ******** ******** ********
******** ******** ******** ******** ******** ******** ******** ********";
            enableButtons();
        }

        //Open Key
        private void button3_Click(object sender, EventArgs e)
        {
            openKey();
            enableButtons();
        }

        //Save Key
        private void button4_Click(object sender, EventArgs e)
        {
            saveKey(Key);
        }

        //Encrypt
```

```
        private void button5_Click(object sender, EventArgs e)
        {
            FileOutputText.Text += "Початок шифрування...\r\n";
            TimeSpan TS = DateTime.Now.TimeOfDay;
            encrypt(PLAINTEXT, Key);
            FileOutputText.Text += "Зашифровано\r\n";
            FileOutputText.Text += (DateTime.Now.TimeOfDay - TS).TotalSeconds.ToString() + " секунди";
        }

        //Decrypt
        private void button6_Click(object sender, EventArgs e)
        {
            FileOutputText.Text += "Початок розшифровки...\r\n";
            TimeSpan TS = DateTime.Now.TimeOfDay;
            decrypt(PLAINTEXT, Key);
            FileOutputText.Text += "Розшифровано\r\n";
            FileOutputText.Text += (DateTime.Now.TimeOfDay - TS).TotalSeconds.ToString() + " секунди";
        }

        //Save File
        private void button7_Click(object sender, EventArgs e)
        {
            saveFile(PLAINTEXT.Text);
        }

        private void enableButtons()
        {
            if (PLAINTEXT != null && Key != null)
            {
                button5.Enabled = true;
                button6.Enabled = true;

            }
            if (Key != null)
            {
                button4.Enabled = true;
                button8.Enabled = true;
                button9.Enabled = true;
            }
        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (Key != null)
            {
                if (checkBox1.Checked)
                    textBox1.Text = Key.ByteToString();
                else  textBox1.Text  =  "********  ********  ********  ********  ********  ********  ********
******** ******** ******** ******** ******** ******** ******** ******** ********";
            }
        }
        //Encrypt text
        private void button8_Click(object sender, EventArgs e)
        {
            PlainTextClass P = new PlainTextClass(Encoding.Unicode.GetBytes(TextInput.Text));
            encrypt(P, Key);
            TextOutput.Text = byteArrayToString(P.Text);
        }
        //Decrypt text
        private void button9_Click(object sender, EventArgs e)
        {
            string bitString = TextInput.Text.Replace(" ", String.Empty);
            byte[] output = new byte[bitString.Length / 8];
            for (int i = 0; i < output.Length; i++)
            {
                for (int b = 0; b <= 7; b++)
                {
                    output[i] |= (byte)((bitString[i * 8 + b] == '1' ? 1 : 0) << (7 - b));
                }
            }
            PlainTextClass P = new PlainTextClass(output);
            decrypt(P, Key);
            TextOutput.Text = Encoding.Unicode.GetString(P.Text);
        }
    }
}
```

2. Form1.Designer.cs

```
namespace Diploma
{
    partial class Form1
    {
        /// <summary>
        /// Обязательная переменная конструктора.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        ///  <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе
ложно.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region

        /// <summary>
        /// Требуемый метод для поддержки конструктора — не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        private void InitializeComponent()
        {
            this.button1 = new System.Windows.Forms.Button();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.label2 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.button5 = new System.Windows.Forms.Button();
            this.button6 = new System.Windows.Forms.Button();
            this.progressBar1 = new System.Windows.Forms.ProgressBar();
            this.button7 = new System.Windows.Forms.Button();
            this.tabControl1 = new System.Windows.Forms.TabControl();
            this.tabPage1 = new System.Windows.Forms.TabPage();
            this.tabPage2 = new System.Windows.Forms.TabPage();
            this.button4 = new System.Windows.Forms.Button();
            this.button2 = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.splitter1 = new System.Windows.Forms.Splitter();
            this.button3 = new System.Windows.Forms.Button();
            this.splitContainer1 = new System.Windows.Forms.SplitContainer();
            this.checkBox1 = new System.Windows.Forms.CheckBox();
            this.FileOutputText = new System.Windows.Forms.TextBox();
            this.splitContainer2 = new System.Windows.Forms.SplitContainer();
            this.splitContainer3 = new System.Windows.Forms.SplitContainer();
            this.splitContainer4 = new System.Windows.Forms.SplitContainer();
            this.TextInput = new System.Windows.Forms.TextBox();
            this.TextOutput = new System.Windows.Forms.TextBox();
            this.button8 = new System.Windows.Forms.Button();
            this.button9 = new System.Windows.Forms.Button();
            this.tabControl1.SuspendLayout();
            this.tabPage1.SuspendLayout();
            this.tabPage2.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).BeginInit();
            this.splitContainer1.Panel1.SuspendLayout();
            this.splitContainer1.Panel2.SuspendLayout();
            this.splitContainer1.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer2)).BeginInit();
            this.splitContainer2.Panel1.SuspendLayout();
            this.splitContainer2.Panel2.SuspendLayout();
            this.splitContainer2.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer3)).BeginInit();
            this.splitContainer3.Panel1.SuspendLayout();
            this.splitContainer3.Panel2.SuspendLayout();
            this.splitContainer3.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer4)).BeginInit();
            this.splitContainer4.Panel1.SuspendLayout();
            this.splitContainer4.Panel2.SuspendLayout();
```

```
            this.splitContainer4.SuspendLayout();
            this.SuspendLayout();
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(5, 3);
            this.button1.Margin = new System.Windows.Forms.Padding(5, 3, 5, 3);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(150, 40);
            this.button1.TabIndex = 0;
            this.button1.Text = "Обрати";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click_1);
            //
            // textBox1
            //
            this.textBox1.BackColor = System.Drawing.SystemColors.ControlLight;
            this.textBox1.Dock = System.Windows.Forms.DockStyle.Top;
            this.textBox1.Font      =      new      System.Drawing.Font("Yu      Gothic      UI",      10.2F,
System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.textBox1.ForeColor = System.Drawing.SystemColors.WindowFrame;
            this.textBox1.Location = new System.Drawing.Point(0, 0);
            this.textBox1.Margin = new System.Windows.Forms.Padding(15, 3, 5, 3);
            this.textBox1.Multiline = true;
            this.textBox1.Name = "textBox1";
            this.textBox1.ReadOnly = true;
            this.textBox1.Size = new System.Drawing.Size(749, 25);
            this.textBox1.TabIndex = 1;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(165, 5);
            this.label2.Margin = new System.Windows.Forms.Padding(5, 0, 5, 0);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(54, 23);
            this.label2.TabIndex = 5;
            this.label2.Text = "Файл:";
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Font     =     new     System.Drawing.Font("Microsoft     Sans     Serif",     10.2F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label3.Location = new System.Drawing.Point(165, 26);
            this.label3.Margin = new System.Windows.Forms.Padding(5, 0, 5, 0);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(139, 20);
            this.label3.TabIndex = 6;
            this.label3.Text = "Оберіть файл";
            //
            // button5
            //
            this.button5.Enabled = false;
            this.button5.Location = new System.Drawing.Point(5, 49);
            this.button5.Margin = new System.Windows.Forms.Padding(5, 3, 5, 3);
            this.button5.Name = "button5";
            this.button5.Size = new System.Drawing.Size(150, 40);
            this.button5.TabIndex = 8;
            this.button5.Text = "Зашифрувати";
            this.button5.UseVisualStyleBackColor = true;
            this.button5.Click += new System.EventHandler(this.button5_Click);
            //
            // button6
            //
            this.button6.Enabled = false;
            this.button6.Location = new System.Drawing.Point(165, 49);
            this.button6.Margin = new System.Windows.Forms.Padding(5, 3, 5, 3);
            this.button6.Name = "button6";
            this.button6.Size = new System.Drawing.Size(150, 40);
            this.button6.TabIndex = 9;
            this.button6.Text = "Розшифрувати";
            this.button6.UseVisualStyleBackColor = true;
            this.button6.Click += new System.EventHandler(this.button6_Click);
            //
            // progressBar1
            //
```

```
            this.progressBar1.Dock = System.Windows.Forms.DockStyle.Bottom;
            this.progressBar1.Location = new System.Drawing.Point(0, 96);
            this.progressBar1.Margin = new System.Windows.Forms.Padding(5, 3, 5, 3);
            this.progressBar1.Minimum = 1;
            this.progressBar1.Name = "progressBar1";
            this.progressBar1.Size = new System.Drawing.Size(749, 10);
            this.progressBar1.Step = 1;
            this.progressBar1.TabIndex = 11;
            this.progressBar1.Value = 1;
            //
            // button7
            //
            this.button7.Anchor                                                                                 =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom                                  |
System.Windows.Forms.AnchorStyles.Left)));
            this.button7.Location = new System.Drawing.Point(5, 339);
            this.button7.Margin = new System.Windows.Forms.Padding(5, 3, 5, 3);
            this.button7.Name = "button7";
            this.button7.Size = new System.Drawing.Size(314, 40);
            this.button7.TabIndex = 12;
            this.button7.Text = "Зберти";
            this.button7.UseVisualStyleBackColor = true;
            this.button7.Click += new System.EventHandler(this.button7_Click);
            //
            // 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.Multiline = true;
            this.tabControl1.Name = "tabControl1";
            this.tabControl1.SelectedIndex = 0;
            this.tabControl1.Size = new System.Drawing.Size(749, 430);
            this.tabControl1.SizeMode = System.Windows.Forms.TabSizeMode.FillToRight;
            this.tabControl1.TabIndex = 15;
            //
            // tabPage1
            //
            this.tabPage1.Controls.Add(this.splitContainer3);
            this.tabPage1.Location = new System.Drawing.Point(4, 32);
            this.tabPage1.Name = "tabPage1";
            this.tabPage1.Padding = new System.Windows.Forms.Padding(3);
            this.tabPage1.Size = new System.Drawing.Size(741, 394);
            this.tabPage1.TabIndex = 0;
            this.tabPage1.Text = "Текст";
            this.tabPage1.UseVisualStyleBackColor = true;
            //
            // tabPage2
            //
            this.tabPage2.Controls.Add(this.splitContainer2);
            this.tabPage2.Controls.Add(this.splitter1);
            this.tabPage2.Location = new System.Drawing.Point(4, 32);
            this.tabPage2.Name = "tabPage2";
            this.tabPage2.Padding = new System.Windows.Forms.Padding(3);
            this.tabPage2.Size = new System.Drawing.Size(741, 394);
            this.tabPage2.TabIndex = 1;
            this.tabPage2.Text = "Файл";
            this.tabPage2.UseVisualStyleBackColor = true;
            //
            // button4
            //
            this.button4.Enabled = false;
            this.button4.Location = new System.Drawing.Point(174, 53);
            this.button4.Margin = new System.Windows.Forms.Padding(5, 3, 5, 3);
            this.button4.Name = "button4";
            this.button4.Size = new System.Drawing.Size(150, 40);
            this.button4.TabIndex = 7;
            this.button4.Text = "Зберти";
            this.button4.UseVisualStyleBackColor = true;
            this.button4.Click += new System.EventHandler(this.button4_Click);
            //
            // button2
            //
            this.button2.Location = new System.Drawing.Point(334, 53);
            this.button2.Margin = new System.Windows.Forms.Padding(5, 3, 5, 3);
            this.button2.Name = "button2";
```

```
            this.button2.Size = new System.Drawing.Size(150, 40);
            this.button2.TabIndex = 3;
            this.button2.Text = "Згенерувати";
            this.button2.UseVisualStyleBackColor = true;
            this.button2.Click += new System.EventHandler(this.button2_Click);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font      =      new      System.Drawing.Font("Yu      Gothic      UI",      10.2F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label1.Location = new System.Drawing.Point(10, 27);
            this.label1.Margin = new System.Windows.Forms.Padding(10);
            this.label1.Name = "label1";
            this.label1.Padding = new System.Windows.Forms.Padding(5, 0, 0, 0);
            this.label1.Size = new System.Drawing.Size(59, 23);
            this.label1.TabIndex = 2;
            this.label1.Text = "Ключ";
            //
            // splitter1
            //
            this.splitter1.Dock = System.Windows.Forms.DockStyle.Bottom;
            this.splitter1.Location = new System.Drawing.Point(3, 388);
            this.splitter1.Name = "splitter1";
            this.splitter1.Size = new System.Drawing.Size(735, 3);
            this.splitter1.TabIndex = 1;
            this.splitter1.TabStop = false;
            this.splitter1.Visible = false;
            //
            // button3
            //
            this.button3.Location = new System.Drawing.Point(14, 53);
            this.button3.Margin = new System.Windows.Forms.Padding(5, 3, 5, 3);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(150, 40);
            this.button3.TabIndex = 4;
            this.button3.Text = "Завантажити";
            this.button3.UseVisualStyleBackColor = true;
            this.button3.Click += new System.EventHandler(this.button3_Click);
            //
            // splitContainer1
            //
            this.splitContainer1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.splitContainer1.FixedPanel = System.Windows.Forms.FixedPanel.Panel1;
            this.splitContainer1.Location = new System.Drawing.Point(0, 0);
            this.splitContainer1.Name = "splitContainer1";
            this.splitContainer1.Orientation = System.Windows.Forms.Orientation.Horizontal;
            //
            // splitContainer1.Panel1
            //
            this.splitContainer1.Panel1.BackColor = System.Drawing.SystemColors.ControlDark;
            this.splitContainer1.Panel1.Controls.Add(this.progressBar1);
            this.splitContainer1.Panel1.Controls.Add(this.button2);
            this.splitContainer1.Panel1.Controls.Add(this.button4);
            this.splitContainer1.Panel1.Controls.Add(this.textBox1);
            this.splitContainer1.Panel1.Controls.Add(this.label1);
            this.splitContainer1.Panel1.Controls.Add(this.button3);
            this.splitContainer1.Panel1.Controls.Add(this.checkBox1);
            //
            // splitContainer1.Panel2
            //
            this.splitContainer1.Panel2.Controls.Add(this.tabControl1);
            this.splitContainer1.Size = new System.Drawing.Size(749, 540);
            this.splitContainer1.SplitterDistance = 106;
            this.splitContainer1.TabIndex = 16;
            //
            // checkBox1
            //
            this.checkBox1.AutoSize = true;
            this.checkBox1.Location = new System.Drawing.Point(82, 26);
            this.checkBox1.Name = "checkBox1";
            this.checkBox1.Size = new System.Drawing.Size(105, 27);
            this.checkBox1.TabIndex = 12;
            this.checkBox1.Text = "Показати";
            this.checkBox1.UseVisualStyleBackColor = true;
            this.checkBox1.CheckedChanged += new System.EventHandler(this.checkBox1_CheckedChanged);
            //
```

```
// FileOutputText
//
this.FileOutputText.Dock = System.Windows.Forms.DockStyle.Fill;
this.FileOutputText.Location = new System.Drawing.Point(0, 0);
this.FileOutputText.Multiline = true;
this.FileOutputText.Name = "FileOutputText";
this.FileOutputText.Size = new System.Drawing.Size(406, 385);
this.FileOutputText.TabIndex = 15;
//
// splitContainer2
//
this.splitContainer2.Dock = System.Windows.Forms.DockStyle.Fill;
this.splitContainer2.FixedPanel = System.Windows.Forms.FixedPanel.Panel1;
this.splitContainer2.Location = new System.Drawing.Point(3, 3);
this.splitContainer2.Name = "splitContainer2";
//
// splitContainer2.Panel1
//
this.splitContainer2.Panel1.Controls.Add(this.button1);
this.splitContainer2.Panel1.Controls.Add(this.button7);
this.splitContainer2.Panel1.Controls.Add(this.button5);
this.splitContainer2.Panel1.Controls.Add(this.button6);
this.splitContainer2.Panel1.Controls.Add(this.label3);
this.splitContainer2.Panel1.Controls.Add(this.label2);
//
// splitContainer2.Panel2
//
this.splitContainer2.Panel2.Controls.Add(this.FileOutputText);
this.splitContainer2.Size = new System.Drawing.Size(735, 385);
this.splitContainer2.SplitterDistance = 325;
this.splitContainer2.TabIndex = 16;
//
// splitContainer3
//
this.splitContainer3.Dock = System.Windows.Forms.DockStyle.Fill;
this.splitContainer3.Location = new System.Drawing.Point(3, 3);
this.splitContainer3.Name = "splitContainer3";
this.splitContainer3.Orientation = System.Windows.Forms.Orientation.Horizontal;
//
// splitContainer3.Panel1
//
this.splitContainer3.Panel1.Controls.Add(this.TextInput);
//
// splitContainer3.Panel2
//
this.splitContainer3.Panel2.Controls.Add(this.splitContainer4);
this.splitContainer3.Size = new System.Drawing.Size(735, 388);
this.splitContainer3.SplitterDistance = 166;
this.splitContainer3.TabIndex = 0;
//
// splitContainer4
//
this.splitContainer4.Dock = System.Windows.Forms.DockStyle.Fill;
this.splitContainer4.FixedPanel = System.Windows.Forms.FixedPanel.Panel1;
this.splitContainer4.Location = new System.Drawing.Point(0, 0);
this.splitContainer4.Name = "splitContainer4";
this.splitContainer4.Orientation = System.Windows.Forms.Orientation.Horizontal;
//
// splitContainer4.Panel1
//
this.splitContainer4.Panel1.BackColor = System.Drawing.SystemColors.ControlDark;
this.splitContainer4.Panel1.Controls.Add(this.button9);
this.splitContainer4.Panel1.Controls.Add(this.button8);
//
// splitContainer4.Panel2
//
this.splitContainer4.Panel2.Controls.Add(this.TextOutput);
this.splitContainer4.Size = new System.Drawing.Size(735, 218);
this.splitContainer4.SplitterDistance = 46;
this.splitContainer4.TabIndex = 0;
//
// TextInput
//
this.TextInput.Dock = System.Windows.Forms.DockStyle.Fill;
this.TextInput.Location = new System.Drawing.Point(0, 0);
this.TextInput.Multiline = true;
this.TextInput.Name = "TextInput";
```

```
            this.TextInput.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
            this.TextInput.Size = new System.Drawing.Size(735, 166);
            this.TextInput.TabIndex = 0;
            //
            // TextOutput
            //
            this.TextOutput.Dock = System.Windows.Forms.DockStyle.Fill;
            this.TextOutput.Location = new System.Drawing.Point(0, 0);
            this.TextOutput.Multiline = true;
            this.TextOutput.Name = "TextOutput";
            this.TextOutput.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
            this.TextOutput.Size = new System.Drawing.Size(735, 168);
            this.TextOutput.TabIndex = 1;
            //
            // button8
            //
            this.button8.Enabled = false;
            this.button8.Location = new System.Drawing.Point(7, 3);
            this.button8.Name = "button8";
            this.button8.Size = new System.Drawing.Size(150, 40);
            this.button8.TabIndex = 0;
            this.button8.Text = "Зашифрувати";
            this.button8.UseVisualStyleBackColor = true;
            this.button8.Click += new System.EventHandler(this.button8_Click);
            //
            // button9
            //
            this.button9.Enabled = false;
            this.button9.Location = new System.Drawing.Point(167, 3);
            this.button9.Name = "button9";
            this.button9.Size = new System.Drawing.Size(150, 40);
            this.button9.TabIndex = 1;
            this.button9.Text = "Розшифрувати";
            this.button9.UseVisualStyleBackColor = true;
            this.button9.Click += new System.EventHandler(this.button9_Click);
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 23F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(749, 540);
            this.Controls.Add(this.splitContainer1);
            this.Font = new System.Drawing.Font("Yu Gothic UI", 10.2F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.Margin = new System.Windows.Forms.Padding(5, 3, 5, 3);
            this.Name = "Form1";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "PohEncrypter";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.tabControl1.ResumeLayout(false);
            this.tabPage1.ResumeLayout(false);
            this.tabPage2.ResumeLayout(false);
            this.splitContainer1.Panel1.ResumeLayout(false);
            this.splitContainer1.Panel1.PerformLayout();
            this.splitContainer1.Panel2.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).EndInit();
            this.splitContainer1.ResumeLayout(false);
            this.splitContainer2.Panel1.ResumeLayout(false);
            this.splitContainer2.Panel1.PerformLayout();
            this.splitContainer2.Panel2.ResumeLayout(false);
            this.splitContainer2.Panel2.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer2)).EndInit();
            this.splitContainer2.ResumeLayout(false);
            this.splitContainer3.Panel1.ResumeLayout(false);
            this.splitContainer3.Panel1.PerformLayout();
            this.splitContainer3.Panel2.ResumeLayout(false);
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer3)).EndInit();
            this.splitContainer3.ResumeLayout(false);
            this.splitContainer4.Panel1.ResumeLayout(false);
            this.splitContainer4.Panel2.ResumeLayout(false);
            this.splitContainer4.Panel2.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer4)).EndInit();
            this.splitContainer4.ResumeLayout(false);
            this.ResumeLayout(false);

        }
```

```
        #endregion

        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Button button5;
        private System.Windows.Forms.Button button6;
        private System.Windows.Forms.ProgressBar progressBar1;
        private System.Windows.Forms.Button button7;
        private System.Windows.Forms.TabControl tabControl1;
        private System.Windows.Forms.TabPage tabPage1;
        private System.Windows.Forms.TabPage tabPage2;
        private System.Windows.Forms.Splitter splitter1;
        private System.Windows.Forms.Button button4;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.SplitContainer splitContainer1;
        private System.Windows.Forms.CheckBox checkBox1;
        private System.Windows.Forms.TextBox FileOutputText;
        private System.Windows.Forms.SplitContainer splitContainer2;
        private System.Windows.Forms.SplitContainer splitContainer3;
        private System.Windows.Forms.SplitContainer splitContainer4;
        private System.Windows.Forms.TextBox TextInput;
        private System.Windows.Forms.Button button9;
        private System.Windows.Forms.Button button8;
        private System.Windows.Forms.TextBox TextOutput;
    }
}
```