МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 2019 р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Програмний модуль для кодування текстової інформації з використанням DES-шифрування та LSB-стеганографії.

Освітньо-кваліфікаційний рівень "бакалавр"
Напрям підготовки 6.050101 "Комп'ютерні науки"
(шифр і назва)

Керівник проекту:	_____ (підпис)	Лифар О. К. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	_____ (ініціали, прізвище)
Здобувач вищої освіти:	_____ (підпис)	Сідєльніков В.В. (ініціали, прізвище)
Група:		КН-15бд

Сєвєродонецьк 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 6.050101 "Комп'ютерні науки"
(шифр і назва)
Спеціальність 122 "Комп'ютерні науки та інформаційні технології"
(шифр і назва)

ЗАТВЕРДЖУЮ:
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 2019 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Сідєльнікова Владислава Валерійовича
(прізвище, ім'я, по батькові)

1. Тема роботи Програмний модуль для кодування текстової інформації з використанням DES-шифрування та LSB-стеганографії.
керівник проекту (роботи) Лифар Олена Костянтинівна, старший викладач
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від " " 201_ р. № ____
2. Термін подання студентом роботи до 12.06.2018
3. Зміст розрахунково-пояснювальної записки структура та оформлення пояснювальної записки має відповідати вимогам методичних вказівок до дипломної роботи та ДСТУ. Основна частина записки повинна містити: аналіз сучасного стану в галузі розробки , вибір алгоритмів шифрування та середовища розробки, опис етапів розробки програмного модуля
4. Перелік графічного матеріалу графічні матеріали дипломної роботи відсутні
5. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст.викл. Критська Я.О.		

6. Дата видачі завдання 20.05.2019

Керівник _____
(підпис)
Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1.	Затвердження теми дипломного проекту	До 03.05.2019	
1.	Аналіз технічного завдання	До 06.05. 2019	
2.	Аналіз сучасного стану в галузі розробки програмних продуктів для шифрування інформації	До 10.05.2019	
3.	Формування вимог до програмного продукту	До 13.05.2019	
4.	Вибір алгоритмів шифрування	До 18.05.2019	
5.	Вибір середовища розробки	До 20.05.2019	
6.	Виконання та оформлення розділу з охорони праці	До 26.05.2019	
7.	Розробка програмного модуля	До 05.06.2019	
8.	Оформлення пояснювальної записки дипломного проекту	До 08.06.2019	
9.	Подання роботи на попередній розгляд	До 09.06.2019	
10.	Подання підписаної керівником і консультантом ДП до ЕК	До 12.06.2019	
11.	Захист дипломного проекту	19.06.2019	

Здобувач вищої освіти _____ Сідєльніков В.В.
(підпис) (прізвище та ініціали)

Керівник Лифар О. К.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту (роботи) бакалавра: 60 с., 14 рис., 4 табл., 15 бібліографічних джерел посилань , 2 додатка.

Об'єкт розробки: Програмний модуль для кодування текстової інформації.

Мета роботи: розробка програмного модуля для подвійного шифрування інформації симетричним методом DES та алгоритмом стеганографії LSB.

В проекті виконано:

1. Проведено аналіз сучасного стану в галузі розробки програмних продуктів для шифрування інформації.
2. Детально розглянуто популярні методи шифрування текстової інформації та алгоритми стеганографії.
3. Зроблено та обґрунтовано вибір середовища розробки програмного модуля.
4. Поділено на етапи, виконано та детально описано процес розробки програмного модуля.

Отримано наступні результати: Результатом роботи є програмний модуль, що надає можливість захищати текстову інформацію, шляхом кодування її в зображення у форматі bmp.

Практичне значення, галузь застосування роботи: Дипломний проект вирішить проблему захисту важливої корпоративної/персональної інформації від зловмисників, шляхом її кодування і подальшого приховування в зображення.

Програмний модуль буде цікавим як звичайним користувачам так і корпораціям, які турбуються про захист персональної, комерційної інформації, розроблені технології, проведені дослідження та корпоративні таємниці.

Ключові слова: ПРОГРАМНИЙ МОДУЛЬ, ШИФРУВАННЯ, СТЕГАНОГРАФІЯ, МЕТОД DES, АЛГОРИТМ LSB.

Умови одержання дипломного проекту: СНУ ім. В. Даля, пр. Центральний 59-А,. м. Сєвєродонецьк, 93400.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

DES (англ. Data Encryption Standard, стандарт шифрування даних) — це симетричний алгоритм шифрування певних даних , стандарт шифрування прийнятий урядом США із 1976 до кінця 1990-х, з часом набув міжнародного застосування.

LSB (англ. Least Significant Bit, найменший значущий біт) — метод стеганографії, суть якого полягає в заміні останніх значущих бітів у контейнері (зображення, аудіо або відеозапису) на біти приховуваного повідомлення.

AES (англ. Advanced Encryption Standard) — симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128/192/256 біт), американський стандарт шифрування США.

Цифровий водяний знак (ЦВЗ) — технологія, створена для захисту авторських прав мультимедійних файлів. Зазвичай цифрові водяні знаки невидимі.

BMP (англ. Bitmap) — bitmap-формат або DIB (англ. device independent bitmap) - формат файлу зображень растрової графіки, в якому зображення зберігається у вигляді двовимірного масиву пікселів.

RGB (англ. Red, Green, Blue — червоний, зелений, синій) — адитивна колірна модель, що описує спосіб синтезу кольору, за якою червоне, зелене та синє світло накладаються разом, змішуючись у різноманітні кольори.

Encrypting File System (англ. EFS, шифрувальна файлова система) - система шифрування даних, що реалізує шифрування на рівні файлів в операційних системах Microsoft Windows NT.

FAT32 (англ. File Allocation Table, таблиця розташування файлів) — це файлова система, яка підтримує томи (логічні диски) обсягом до 8 терабайт і використовує для зберігання файлів менші фрагменти диска, ніж файлова система FAT16.

LACK (англ. Lost Audio Packets Steganography) — приховування інформації в полях заголовку.

ВСТУП

Широке застосування комп'ютерних технологій в автоматизованих системах обробки інформації та управління призвело до загострення проблеми захисту інформації від несанкціонованого доступу. Захист інформації в комп'ютерних системах має низку специфічних особливостей, пов'язаних з тим, що інформація може легко і швидко копіюватися і передаватися по каналах зв'язку. Сьогодні навряд чи вдасться знайти компанію, між комп'ютерами якої не передавалася б важлива комерційна інформація. Повноцінна система захисту такої передачі має виняткове значення і без шифрування тут ніяк не обійтися. Це завдання вирішується різними засобами із застосуванням різних криптографічних технологій.

Про своєчасність і актуальність розглянутої проблеми говорить той факт, що все більше поширюються “хакерскі атаки” на компанії з ціллю викрадення інформації.

Тому в даний час все більш актуальним стає захист важливої інформації методом її шифрування.

Завданням на дипломний проект є створення програмного модуля для кодування текстової інформації.

Модуль надасть можливість захищати текстову інформацію, шляхом кодування її в зображення у форматі bmp, для подальшої передачі за допомогою мережі Інтернет або змінних носіїв. При втраті зашифрованого зображення власником, потенційний викрадач не зможе розшифрувати інформацію без необхідного ключа.

Дипломний проект вирішить проблему захисту важливої корпоративної/персональної інформації від зловмисників, шляхом її кодування і подальшого приховування в зображення. Інформація, яка міститься в зображенні непомітна для зору, що ускладнює процес її пошуку та розшифрування зловмисником. За час, витрачений на цей процес, інформація втратить свою актуальність чи буде недійсна.

Програмний модуль буде цікавим як звичайним користувачам так і корпораціям, які турбуються про захист персональної, комерційної інформації,

1 АНАЛІЗ СУЧАСНОГО СТАНУ В ГАЛУЗІ РОЗРОБКИ ПРОГРАМНИХ ПРОДУКТІВ ДЛЯ ШИФРУВАННЯ ІНФОРМАЦІЇ

1.1 Технічне завдання

Завданням дипломного проекту є розробка програмного модуля для подвійного шифрування інформації. Перший етап шифрування — шифрування текстової інформації методом DES, другий етап – приховування зашифрованої інформації в bmp- файлі.

Програмний модуль захистить важливу інформацію та приховає факт того, що ця інформація була зашифрована або взагалі існувала.

Процес виконання завдання дипломного проекту можна поділити на етапи:

- аналіз сучасного стану в галузі розробки;
- вибір алгоритмів шифрування;
- вибір середовища розробки;
- розробка інтерфейсу користувача;
- реалізація алгоритму DES;
- реалізація LSB алгоритму;
- розробка інструкцій для користувача.

На даний час існує чимало програм для шифрування інформації. Кожна з них має свої переваги та недоліки. Плюси і мінуси найпоширеніших з них представлені нижче.

1.1 Програма Folder Lock

Основні можливості програми Folder Lock наступні [1]:

– AES-шифрування, довжина ключа 256 біт.

– Приховування файлів і папок.

– Шифрування файлів (за допомогою створення віртуальних дисків - сейфів).

– Резервне копіювання он-лайн.

– Створення захищених USB / CD / DVD-дисків.

– Шифрування електронної пошти.

– Створення зашифрованих «гаманців», що зберігають інформацію про кредитні картки, рахунки і т.д.

Переваги програми Folder Lock:

– Привабливий і зрозумілий інтерфейс, який сподобається починаючим користувачам, які володіють англійською мовою.

– Прозоре шифрування, створення віртуальних зашифрованих дисків, з якими можна працювати, як зі звичайними дисками.

– Можливість резервного он-лайн копіювання і синхронізації зашифрованих контейнерів (сейфів).

– Можливість створення саморозшифровуваних контейнерів на USB / CD / DVD-дисках.

Недоліки програми:

– Немає підтримки української мови, що ускладнить роботу з програмою користувачів, не знайомих з англійською мовою.

– Сумнівні функції Lock Files (яка просто приховує, а не «замикає» файли) і Make Wallets (малоефективна без експорту інформації).

– Відсутність можливості підписання файлів, перевірки цифрового підпису.

– При відкритті сейфа не дозволяє вибрати букву диска, яка буде призначена віртуальному диску, який відповідає сейфу. У налаштуваннях програми можна вибрати тільки порядок, в якому програма буде призначати букву диска - по зростанню (від A до Z) або по спаданню (від Z до A).

– Нема інтеграції з поштовими клієнтами, є тільки можливість зашифрувати вкладення.

– Висока вартість хмарного резервного копіювання.

1.2 Програма PGP Desktop

Переваги програми PGP Desktop [1]:

– Повноцінна програма, що використовується для шифрування файлів, підписання файлів і перевірки електронного підпису, прозорого шифрування (віртуальні диски і шифрування всього розділу), шифрування електронної пошти.

– Підтримка сервера ключів keyserver.pgp.com.

– Можливість створення саморозшифровуваних архівів.

– Можливість шифрування системного жорсткого диска.

– Функція PGP NetShare.

– Можливість затирання вільного місця.

– Тісна інтеграція з Провідником.

Недоліки програми:

– Відсутність підтримки української мови, що ускладнить роботу з програмою користувачам, які не знають англійську мову.

– Нестабільна робота програми.

– Низька продуктивність програми.

– Є підтримка AOL IM, але немає підтримки Skype і Viber.

– Вже розшифровані листи залишаються незахищеними на клієнті.

– Захист пошти працює тільки в режимі перехоплення, який швидко вам набридне, оскільки вікно захисту пошти буде з'являтися кожен раз для кожного нового сервера.

1.3 Програма CyberSafe Top Secret

Переваги програми CyberSafe Top Secret [1]:

– Підтримка алгоритмів шифрування ГОСТ та сертифікованого криптопровайдера CryptoPro, що дозволяє використовувати програму не тільки приватним особам і комерційним організаціям, а й державним установам.

– Підтримка прозорого шифрування папки, що дозволяє використовувати програму в якості заміни EFS. Оскільки програма забезпечує кращий рівень продуктивності і безпеки, то ця заміна цілком виправдана.

– Можливість підписання файлів електронним цифровим підписом і можливість перевірки підпису файлу.

– Вбудований сервер ключів. Надає доступ до вже опублікованих ключів та можливість додавати власні.

– Можливість створення віртуального зашифрованого диска і можливість шифрування всього розділу.

– Можливість створення саморозшифровуваних архівів.

– Можливість безкоштовного хмарного резервного копіювання, яке працює з будь-яким сервісом - як платним, так і безкоштовним.

– Двофакторна аутентифікація користувача.

– Система довірених додатків, що дозволяє вирішити доступ до зашифрованих файлів тільки певних додатків.

– Додаток CyberSafe підтримує набір інструкцій AES-NI, що позитивно позначається на продуктивності програми.

– Драйвер програми CyberSafe дозволяє працювати по мережі, що дає можливість організувати корпоративне шифрування.

– Український інтерфейс програми. Для англомовних користувачів є можливість перемикання на англійську мову.

Недоліки програми:

– Погана локалізація.

– Немає можливості шифрувати системний диск.

Всі представлені продукти є прекрасними шифрувальниками, але у них є один спільний вагомий недолік – відсутність подвійного шифрування. Тобто і текстове шифрування, і графічне шифрування.

Тому створення програмного модуля, який надасть можливість подвійного шифрування є актуальною темою для дипломного проекту.

2 ВИБІР АЛГОРИТМІВ ШИФРУВАННЯ

2.1 Шифрування текстової інформації

Сучасна криптографія базується на алгоритмах шифрування з ключем. Шифрування з ключем - це методи кодування, коли ключ зберігається в таємниці від третьої сторони (зберігати криптографічний алгоритм у цьому випадку не потрібно).

Алгоритми шифрування з ключем поділяють на дві великі групи - алгоритми симетричного шифрування і алгоритми асиметричного шифрування.

Асиметричне шифрування - набір методів криптографічного шифрування, в яких використовують два ключі - таємний (приватний) і відкритий; жоден із ключів не може бути обчислений з іншого за прийнятний час. Таке шифрування ще називають шифруванням з відкритим ключем.

Симетричне шифрування - це метод, за якого ключі шифрування і розшифрування або однакові, або легко виводяться один з одного, забезпечуючи таким чином спільний ключ, який є таємним.

Наразі з-поміж криптографічних методів домінують симетричне й асиметричне шифрування. Симетричні та асиметричні криптоалгоритми мають свої перевага та недоліки. Симетричні криптоалгоритми порівняно з асиметричними мають більшу швидкодію та меншу довжину ключа. Асиметричне шифрування застосовують за такої організації криптосистем, коли використання симетричних алгоритмів є неможливим.

2.1.1 Асиметричне шифрування

Алгоритми асиметричного шифрування [2] базуються на шифрах з відкритими ключами. Шифрування на відкритому ключі - відносно нова галузь криптографії.

В асиметричних криптоалгоритмах для шифрування і розшифрування використовують різні ключі: для шифрування - відкриті, для розшифрування - таємні.

Асиметрична криптографія базується на ідеях В. Діффі та М. Хеллмана про шифрування з двома ключами, що стали відомими у 1976 році. Але першим алгоритмом асиметричного шифрування (шифрування на відкритому ключі), який набув практичного значення, став алгоритм, запропонований Р. Рівестом (Rivest), А. Шаміром (Shamir) і Л. Адлеманом (Adleman) у 1978 році. Він дістав назву алгоритм RSA (за першими літерами прізвищ його авторів).

На рис. 2.1 наведено структурну схему шифрування на відкритому ключі.

Рис. 2.1. Структурна схема шифрування на відкритому ключі

Математичне підґрунтя асиметричних криптоалгоритмів складають важкооборотні (односторонні) функції. У теорії складності обчислень розглядають поняття, яке характеризує рівень складності обчислень (кількість операцій) залежно від розміру вхідних даних. Сучасна асиметрична криптографія базується на алгоритмах Ель-Гамаля та Міллера - Коблиця.

Алгоритми асиметричного шифрування, так само як і симетричного, застосовують для шифрування масивів даних, але їхня швидкість значно нижча. Основне призначення асиметричних алгоритмів - забезпечення ефективного функціонування сучасних криптосистем. Саме ці алгоритми покладено в основу задач автентифікації користувачів, контролю цілісності інформації, унеможливлений відмови від авторства чи факту одержання даних тощо.

2.1.2. Симетричне шифрування

Симетричне шифрування [3] ще називають шифруванням на таємному ключі, тобто на ключі, який обидві сторони обміну таємно від інших використовують для шифрування та розшифрування повідомлень. На рис. 2.2 наведено структурну схему шифрування на таємному ключі.

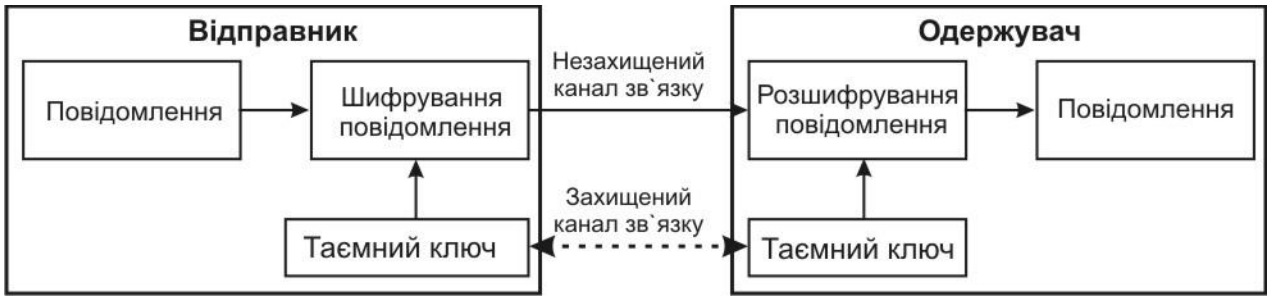


Рис. 2.2. Структурна схема шифрування на таємному ключі

Основне призначення симетричних криптоалгоритмів - шифрування великих масивів даних із великою швидкістю. Разом із тим, через потребу мати захищений канал передавання таємного ключа ці криптоалгоритми під час створення сучасних криптосистем виявляють дуже низьку гнучкість.

Розрізняють дві великі групи алгоритмів симетричного шифрування: потокове шифрування та блокове шифрування.

2.1.2.1 Потокове шифрування

Потокове шифрування - це спосіб шифрування даних, коли кожний знак шифрується окремо. Цей криптоалгоритм обробляє інформацію посимвольно, тому, користуючись ним, можна послідовно шифрувати та розшифровувати інформацію будь-якого обсягу. Таке шифрування застосовують переважно в каналах зв'язку.

Найбільш поширені потокові шифри, накладаючи вихідний текст на попередньо згенеровану шифрувальну послідовність, посимвольно опрацьовують його. Таке накладання здійснюється з використанням відомої в арифметиці оборотної операції за модулем 2, яку застосовують для шифрування тексту. Знаючи результат і всі операнди, крім одного, за допомогою арифметичної операції за модулем 2 можна визначити цей невідомий операнд.

За використання потокового шифрування окремі символи згенерованої шифрувальної послідовності зазвичай позначаються грецькою літерою у (гамма). Тому потокові шифри ще називають шифрами гаммування.

Шифр Цезаря [4] — симетричний алгоритм шифрування підстановками. Використовувався римським імператором Юлієм Цезарем для приватного листування.

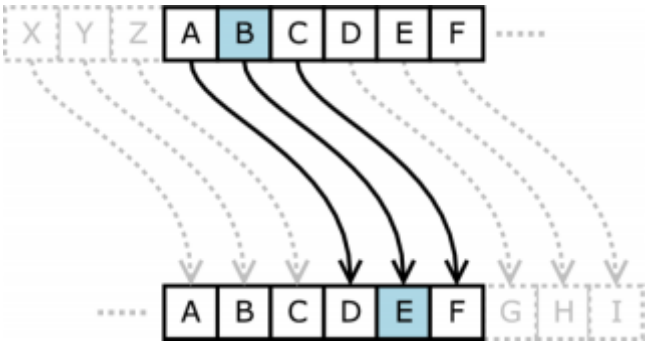


Рисунок 2.3 – Принцип дії шифру Цезаря

Принцип дії полягає в тому, щоб циклічно зсунути алфавіт, а ключ — це кількість літер, на які робиться зсув. Принцип показано на рис. 2.3.

Якщо зіставити кожному символу алфавіту його порядковий номер (нумеруючи з 0), то шифрування і дешифрування можна виразити формулами (5.1) та (5.2) :

$$y = (x + k) \bmod n \tag{2.1}$$

$$y = (y - k) \bmod n \tag{2.2}$$

де x — порядковий номер символу відкритого тексту;

y — порядковий номер символу шифрованого тексту;

n — потужність алфавіту;

k — ключ.

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

Припустимо, що, використовуючи шифр Цезаря, з ключем, який дорівнює 3, необхідно зашифрувати словосполучення «ШИФР ЦЕЗАРЯ».

Для цього зрушимо алфавіт так, щоб він починався з четвертої букви (Г). Отже, беручи вихідний алфавіт:

"АБВГДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ",

і зміщуючи всі літери вліво на 3, отримуємо відповідність:

"А Б В Г Д Е Є Ж З И І Ї Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ю Я Г Д Е Є Ж З И І Ї Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ю Я А Б В", де Г=А, Д=Б, Е=В, і т. д.

Використовуючи цю схему, відкритий текст «ШИФР ЦЕЗАРЯ» перетворюється на «ЮЙЧУ ЩЗЇГУВ». Для того, щоб одержувач повідомлення міг відновити вихідний текст, необхідно повідомити йому, що ключ — 3.

Шифр Віженера [5] — це метод шифрування літерного тексту з використанням ключового слова.

У шифрі Цезаря кожна буква алфавіту зсувається на кілька позицій; наприклад в шифрі Цезаря при зсуві +3, A стало б D, B стало б E і так далі. Шифр Віженера складається з послідовності декількох шифрів Цезаря з різними значеннями зсуву. Для шифрування може використовуватися таблиця алфавітів, звана квадрат Віженера.

Таблиця Віженера латинської абетки (рис. 2.4) складається з рядків по 26 символів, де кожний наступний рядок зсувається на кілька позицій. Таким чином, в таблиці виходить 26 різних шифрів Цезаря.

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

Рисунок 2.4 – Таблиця Віженера латинської абетки

На різних етапах кодування шифр Віженера використовує різні алфавіти з цієї таблиці. На кожному етапі шифрування використовуються різні алфавіти, які обираються в залежності від символу ключового слова. Наприклад, припустимо, що вихідний текст має вигляд:

ATTACKATDAWN

Людина, яка посилає повідомлення, записує ключове слово ("LEMON") циклічно доти, поки його довжина не буде відповідати довжині вихідного тексту:

LEMONLEMONLE

Перший символ вихідного тексту A зашифрований послідовністю L, яка є першим символом ключа. Перший символ L шифрованого тексту знаходиться на перетині рядка L і стовпці A в таблиці Віженера. Так само для другого символу вихідного тексту використовується другий символ ключа; тобто другий символ шифрованого тексту X виходить на перетині рядка E і стовпці T. Інша частина вихідного тексту шифрується подібним способом.

Оригінальний текст: ATTACKATDAWN

Ключ: LEMONLEMONLE

Зашифрований текст: LXFOPVEFRNHR

Розшифрування проводиться таким чином: знаходимо в таблиці Віженера рядок, відповідний першому символу ключового слова; в цьому рядку знаходимо перший символ зашифрованого тексту. Стовпець, в якому знаходиться даний символ, відповідає першому символу вихідного тексту. Наступні символи зашифрованого тексту розшифровуються так само за формулою (2.3).

$$C_i = P_i + K_i \pmod{26} \tag{2.3}$$

З спостереження за частотою збігу витікає формула (2.4):

$$P_i = C_i - K_i \pmod{26} \tag{2.4}$$

2.1.2.2 Блокове шифрування

Блокове шифрування - спосіб шифрування даних, коли кожний блок, що передається, може шифруватися окремо. Блоковий шифр - процедура відображення множини вхідних блоків вихідного тексту на множину блоків зашифрованого тексту. Блоки вихідних даних можуть налічувати від одного до кількох сотень бітів. У сучасних системах блокового шифрування використовують блоки з 64, 128, 192 або 256 біт. Алгоритми блокового шифрування - це комбінація оборотних криптоперетворень, які виконуються багаторазово.

Стандарт шифрування - це повний опис алгоритму шифрування (та правил його використання), призначеного для програмної або апаратної реалізації, обов'язкового для використання організаціями, які зазначені в цьому стандарті. На сьогодні всі розвинені країни світу мають свої стандарти шифрування.

Завдяки стійкості та легкості апаратної реалізації алгоритмів блокового шифрування (за наявності можливості організувати таємний канал обміну

ключами) вони мають суттєві переваги порівняно з іншими симетричними алгоритмами.

Найбільшого поширення серед блокових шифрів набули шифри, які базуються на багаторазових циклічних повтореннях. Шифр такого типу називають мережею. Відомим шифром цього типу є мережа Фейстела, або мережа Фейстела 1-го типу - ітеративний блоковий шифр, побудований за принципом Фейстела.

Алгоритм DES [6] – блоковий алгоритм, тобто при шифруванні вихідне повідомлення перекладається в двійковий код, а потім розбивається на блоки і кожен блок окремо зашифровується (розшифровується). За стандартом (прийнятий в 1977 році) розмір блоку DES дорівнює 64 біта, тобто використовуючи 8-ми бітове кодування ASCII, в одному блоці - 8 символів.

Тепер же в основному використовується 16-ти бітова кодування Юнікод (UTF-16), тому, щоб зберегти довжину блоку рівну 8-ми символів, збільшимо розмір блоку DES до 128 біт.

Для того, щоб зашифрувати повідомлення алгоритмом DES, необхідно виконати наступну послідовність кроків:

– довести вихідне повідомлення до такого розміру (в бітах), щоб воно без остачі поділялося на розмір блоку (sizeOfBlock = 128 біт);

– розділити вихідне повідомлення на блоки;

– довести довжину ключа до довжини половини блоку;

– перевести ключ в бінарний формат (в нулі і одиниці);

– провести над кожним блоком пряме перетворення мережею Фейстеля протягом 16-ти раундів. Після кожного раунду необхідно виконувати циклічний зсув ключа на задану кількість символів;

– з'єднати всі блоки разом; таким чином отримаємо повідомлення, зашифроване алгоритмом DES.

Розшифрування DES проводиться за аналогією. Використовується зворотне перетворення мережею Фейстеля.

Мережа Фейстеля використовується в алгоритмі DES для зашифровування (пряме перетворення мережею) та розшифрування (зворотне перетворення). Ці перетворення зображені на малюнках 1 і 2 відповідно.

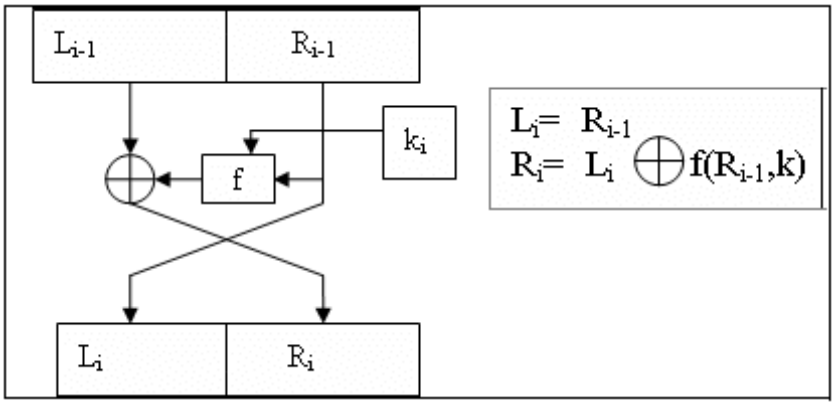


Рисунок 2.5 - Пряме перетворення мережею Фейстеля

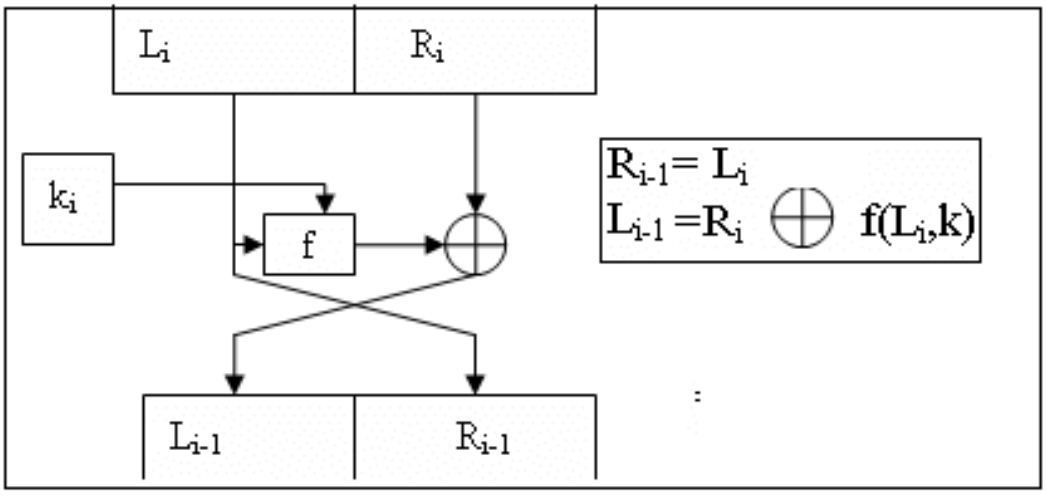


Рисунок 2.6 - Зворотне перетворення мережею Фейстеля

2.2 Шифрування графічної інформації. Стеганографія.

Стеганографія — тайнопис, при якому повідомлення, закодоване таким чином, що не виглядає як повідомлення — на відміну від криптографії. Таким чином непосвячена людина принципово не може розшифрувати повідомлення — бо не знає про факт його існування.

Якщо криптографія приховує зміст повідомлення, то стеганографія приховує сам факт існування повідомлення.

Наприкінці 90-х років виділилося кілька напрямків стеганографії:

— Класична стеганографія

— Комп'ютерна стеганографія

— Цифрова стеганографія

2.2.1 Комп'ютерна стеганографія

Комп'ютерна стеганографія [7] — напрям класичної стеганографії, заснований на особливостях комп'ютерної платформи. Приклади — стеганографічна файлова система StegFS для Linux, приховування даних в областях форматів файлів, які не використовуються, підміна символів в назвах файлів, текстова стеганографія і т. д. Наведемо деякі приклади:

— Використання зарезервованих полів комп'ютерних форматів файлів — суть методу полягає в тому, що частина поля розширень, не заповнена інформацією про розширення, за замовчуванням заповнюється нулями. Відповідно ми можемо використовувати цю «нульову» частину для запису своїх даних. Недоліком цього методу є низька ступінь скритності і малий обсяг переданої інформації.

— Метод приховування інформації в невикористовуваних місцях гнучких дисків — при використанні цього методу інформація записується в невживані частини диска, наприклад, на нульову доріжку. Недоліки: маленька продуктивність, передача невеликих за обсягом повідомлень.

— Метод використання особливих властивостей полів форматів, які не відображаються на екрані — цей метод ґрунтується на спеціальних «невидимих» полях для отримання виносок, покажчиків. Наприклад, написання чорним шрифтом на чорному тлі. Недоліки: маленька продуктивність, невеликий обсяг переданої інформації.

— Використання особливостей файлових систем — при зберіганні на жорсткому диску файл завжди (не рахуючи деяких файлових систем, наприклад, ReiserFS) займає ціле число кластерів (мінімальних обсягів інформації, що адресуються). Наприклад, в файловій системі FAT32 (використовувалася в Windows98/Me/2000) стандартний розмір кластера — 4 Кб. Відповідно для зберігання 1 Кб інформації на диску виділяється 4 Кб інформації, з яких 1Кб потрібен для зберігання файлу, а інші 3 ні на що не використовуються — відповідно їх можна використовувати для зберігання інформації. Недолік даного методу: легкість виявлення.

2.2.2 Цифрова стеганографія

Розвиток засобів цифрової обчислювальної техніки дав поштовх для розвитку комп'ютерної стеганографії, яка ґрунтується на вбудовуванні секретного повідомлення в цифрові дані, що, як правило, мають аналогову природу (аудіозаписи, зображення, відео). Можливе також вбудовування інформації в текстові та скомпресовані файли.

Цифрова стеганографія [7] — напрям класичної стеганографії, заснований на приховуванні або впровадженні додаткової інформації в цифрові об'єкти, викликаючи при цьому деякі спотворення цих об'єктів. Але, як правило, дані об'єкти є мультимедіа-об'єктами (зображення, відео, аудіо, текстури 3D-об'єктів) та внесення спотворень, які знаходяться нижче межі чутливості середньостатистичної людини, не призводить до помітних змін цих об'єктів.

Крім того, в оцифрованих об'єктах, тобто таких, що спочатку мають аналогову природу, завжди присутній шум квантування; також, при відтворенні цих об'єктів з'являється додатковий аналоговий шум і нелінійні спотворення апаратури, все це сприяє більшій непомітності прихованої інформації.

2.2.3 Мережна стеганографія

Останнім часом набули популярності методи, коли прихована інформація передається через комп'ютерні мережі з використанням особливостей роботи протоколів передачі даних. Такі методи одержали назву «мережна стеганографія» [7]. Цей термін вперше ввів Кжиштоф Щипьорский (Krzysztof Szczypiorski) в 2003 році. Типові методи мережевої стеганографії включають зміну властивостей одного з мережевих протоколів. Крім того, може використовуватися взаємозв'язок між двома або більше різними протоколами з метою більш надійного приховування передачі секретного повідомлення. Мережна стеганографія охоплює широкий спектр методів, зокрема:

— WLAN стеганографія ґрунтується на методах, які використовуються для передачі стеганограм в бездротових локальних мережах (Wireless Local Area Networks). Практичний приклад WLAN стеганографії — система HICCUPS (Hidden Communication System for Corrupted Networks).

— LACK стеганографія — приховування повідомлень під час розмов з використанням IP-телефонії. Наприклад: використання пакетів, що затримуються, або навмисно пошкоджуються та ігноруються приймачем (цей метод називають LACK — Lost Audio Packets Steganography), або приховування інформації в полях заголовку, які не використовуються.

VoIP (англ. voice over IP) — технологія передачі медіа даних в реальному часі за допомогою сімейства протоколів TCP/IP. IP-телефонія — система зв'язку, при якій аналоговий звуковий сигнал від одного абонента дискретизується (кодується в цифровий вигляд), компресується і пересилається по цифрових каналах зв'язку до іншого абонента, де проводиться зворотна операція — декомпресія, декодування і відтворення. Розмова відбувається у формі аудіо-потоків за допомогою протоколів RTP (Real-Time Transport Protocol)

LACK — це метод стеганографії для IP-телефонії, який модифікує пакети з голосовим потоком. Він використовує те, що в типових мультимедійних

комунікаційних протоколах, таких як RTP, надмірно затримані пакети вважаються приймачем марними і відкидаються.

Принцип функціонування LACK виглядає наступним чином. Передавач (Аліса) вибирає один з пакетів з голосового потоку і його корисне навантаження замінює бітами таємного повідомлення — стеганограмою, яка вбудовується в пакет. Потім обраний пакет навмисно затримується. Кожного разу, коли надмірно затриманий пакет досягає отримувача, незнайомого з стеганографічною процедурою, він відкидається. Однак, якщо отримувач (Боб) знає про прихований зв'язок, то замість видалення отриманих RTP пакетів, він вилучає приховану інформацію.

2.2.4 Алгоритми стеганографії

Існуючі алгоритми вбудовування таємної інформації можна поділити на декілька підгруп:

— Працюючі з самим цифровим сигналом. Наприклад, метод LSB (Least Significant Bit);

— «Впаювання» прихованої інформації. В даному випадку відбувається накладення приховуваного зображення (звуку, іноді тексту) поверх оригіналу. Часто використовується для вбудовування ЦВЗ (цифровий водяний знак).

— Використання особливостей форматів файлів. Сюди можна віднести запис інформації в метадані або в різні інші не використовувані зарезервовані поля файлу.

За способом вбудовування інформації стегоалгоритми можна розділити на лінійні (адитивні: A17, A18, Ь18Б, A21, A25), нелінійні та інші.

2.2.4.1 LSB (Least Significant Bit)

LSB (Least Significant Bit, найменший значущий біт) [8] — суть цього методу полягає в заміні останніх значущих бітів у контейнері (зображення, аудіо

або відеозапису) на біти приховуваного повідомлення. Різниця між порожнім і заповненим контейнерами повинна бути не відчутна для органів сприйняття людини.

Принцип цього методу полягає в наступному: Припустимо, є 8-бітне зображення в градаціях сірого. 00h (00000000b) позначає чорний колір, FFh (11111111b) — білий. Усього є 256 градацій (2 ^ 8). Також припустимо, що повідомлення складається з 1 байта — наприклад, 01101011b. При використанні 2 молодших біт в описах пікселів, нам буде потрібно 4 пікселя. Припустимо, вони чорного кольору. Тоді пікселі, що містять приховане повідомлення, будуть виглядати наступним чином: 00000001 00000010 00000010 00000011. Тоді колір пікселів зміниться: першого — на 1/255, другого і третього — на 2/255 і четвертого — на 3/255. Такі градації, мало того що непомітні для людини, можуть взагалі не відобразитися при використанні низькоякісних пристроїв виведення. В ролі базового контейнера пропонується використовувати файли BMP-зображень високої роздільності з глибиною кольору 24 та 32 біти, таємне зображення може мати розширення .BMP, .GIF, .PNG, .JPEG.

Недоліком методу LSB є чутливість до розміру зображення, тобто чим менший розмір зображення, тим більше будуть відрізнятися два сусідні пікселі, тому пропонується використовувати зображення з великою роздільністю. Також метод «видає себе» при побітовому перегляді зображення, де чітко видно області зображення в які «вбудовано» таємну інформацію. Попри це, метод запису Least Sagnificant Bit є досить популярним, стійким та простим в реалізації.

BlindHide (приховування наосліп). Найпростіший алгоритм: дані записують, починаючи з верхнього лівого кута зображення до правого нижнього — піксель за пікселем. Приховані дані програма записує у наймолодших бітах кольорів пікселя. Приховані дані розподіляються у контейнері нерівномірно. Якщо приховані дані не заповнять повністю контейнер, то лише верхня частина зображення буде засміченою.

HideSeek (заховати-знайти). Цей алгоритм у псевдовипадковий спосіб розподіляє приховане повідомлення у контейнері. Для генерації випадкової

послідовності використовує пароль. Дещо «розумніший» алгоритм, але все ж не враховує особливостей зображення-контейнера.

FilterFirst (попередня фільтрація). Виконує фільтрацію зображення-контейнера — пошук пікселів, у які записуватиметься прихована інформація (для яких зміна наймолодших розрядів буде найменш помітною для ока людини).

BattleSteg (стеганографія морської битви). Найскладніший і найдосконаліший алгоритм. Спочатку виконує фільтрацію зображення-контейнера, після чого прихована інформація записується у «найкращі місця» контейнера у псевдовипадковий спосіб (подібно, як у HideSeek).

Інші методи приховування інформації в графічних файлах орієнтовані на формати файлів з втратою, наприклад, JPEG. На відміну від LSB вони більш стійкі до геометричних перетворень. Це виходить за рахунок варіювання в широкому діапазоні якості зображення, що призводить до неможливості визначення джерела зображення.

2.2.4.2 Цифрові водяні знаки (ЦВЗ)

Найчастіше стеганографія використовується для створення цифрових водяних знаків. На відміну від звичайних їх можна нанести і відшукати тільки за допомогою спеціального програмного забезпечення — цифрові водяні знаки записуються як псевдовипадкові послідовності шумових сигналів, згенерованих на основі секретних ключів. Такі знаки можуть забезпечити автентичність або недоторканість документа, ідентифікувати автора або власника, перевірити права дистриб'ютора або користувача, навіть якщо файл був оброблений або спотворений.

Цифровий водяний знак (ЦВЗ) [9] — технологія, створена для захисту авторських прав мультимедійних файлів та інтелектуальної власності контейнера (Intellectual Property). Зазвичай цифрові водяні знаки невидимі. Однак ЦВЗ можуть бути видимими на зображенні або відео. Зазвичай це інформація являє собою текст або логотип, який ідентифікує автора.

Стеганографія застосовує ЦВЗ, коли сторони обмінюються секретними повідомленнями, впровадженими в цифровий сигнал. Використовується як засіб захисту документів з фотографіями — паспортів, водійських посвідчень, кредитних карт з фотографіями.

ЦВЗ можна також використовувати для виявлення потенційних піратів: під час продажу в зображення вбудовують інформацію про час продажу та інформацію про покупця. Ключовою відмінністю ЦВЗ від звичайного приховання інформації є наявність активного противника. Наприклад, використовуючи ЦВЗ для захисту авторського права, активний противник намагатиметься видалити чи змінити вбудовані ЦВЗ. Тому основною вимогою є стійкість вбудованих даних до атак. Таємність не є настільки важливою, як у прихованій комунікації.

2.3 Обґрунтування вибору методів шифрування

Для шифрування текстової інформації було обрано метод DES. Тому, що цей метод використовує тільки один ключ довжиною 56 біт; зашифрувавши повідомлення за допомогою одного пакета, для розшифрування можна використовувати будь-який інший; відносна простота алгоритму забезпечує високу швидкість обробки інформації. Також у цього алгоритму досить висока стійкість до взламу.

Для шифрування графічної інформації в рамках даного проекту обрано метод LSB. Тому, що по-перше, мультімедіаконтейнери не викликають підозр: можна без проблем надіслати другу свою фотографію або симпатичний пейзаж. По-друге, молодші біти оцифрованих зображень, звуку або відео можуть мати різний розподіл в залежності від застосовуваних параметрів аналого-цифрового перетворення, від додаткової комп'ютерної обробки і від інших факторів. Ця особливість робить метод найменш значущих бітів найбільш захищеним від виявлення вкладення. Нарешті, по-третє, реалізації LSB для більшості стандартів файлів-контейнерів не вимагають значних витрат часу і сил - ідея зазначеного методу проста, як все геніальне.

3 ВИБІР СЕРЕДОВИЩА РОЗРОБКИ

Середовищем розробки своєї програми я обрав Microsoft Visual Studio з підтримкою мови програмування C#.

Microsoft Visual Studio - це набір інструментів для створення програмного забезпечення: від планування до розробки інтерфейсу користувача, написання коду, тестування, налагодження, аналізу якості коду і продуктивності, розгортання в середовищах клієнтів і збору даних телеметрії по використанню. Ці інструменти призначені для максимально ефективної спільної роботи; всі вони доступні в інтегрованому середовищі розробки (IDE) Visual Studio.

Visual Studio можна використовувати для створення різних типів додатків, від простих додатків для магазину та ігор для мобільних клієнтів до великих і складних систем, які обслуговують підприємства та центри обробки даних.

За допомогою Visual Studio можна створювати:

1. додатки та ігри, які виконуються не тільки на платформі Windows, але і на Android і iOS;
2. веб-сайти і веб-служби на основі ASP.NET, JQuery, AngularJS і інших популярних платформ;
3. додатки для самих різних платформ і пристроїв, включаючи, але не обмежуючись: Office, Sharepoint, Hololens, Kinect і "Інтернету речей";
4. ігри і графічні додатки для різних пристроїв Windows, включаючи Xbox, з підтримкою DirectX.

За замовчуванням Visual Studio забезпечує підтримку C #, C і C ++, JavaScript, F # і Visual Basic.

C# — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET.

Порівнюючи C# з аналогом C++ можна виділити переваги першої.

C# дозволяє стартувати розробку швидше, а це дозволяє швидше отримати прототип рішення. Швидкість розробки на C# на початкових етапах проекту значно вище в порівнянні з C ++.

Якщо говорити про сукупність суб'єктивних «простоти розробки», «краси коду» і об'єктивної продуктивності, то використовуючи C # простіше написати код, що задовольняє цим критеріям одночасно.

Величезна кількість бібліотек з .NET йде в базі, плюс до них багато вільно доступних бібліотек, це покриває практично всі першорядні завдання розробки під Windows. Наявність великої кількості стандартних типів майже позбавляє від бібліотек, де базові типи перевизначені. І в силу того, що бібліотеки C # порівняно молоді, - інтерфейси бібліотек, як правило, краще вписуються в ті чи інші шаблони проектування, що часто спрощує їх вивчення.

Використовуючи C #, менше шансів допустити помилку в принципово складному коді і більше шансів написати чистий код, володіючи тими ж ресурсами. Це може бути корисно при вирішенні досить складних, але не вимогливих до продуктивності завдань.

4 РОЗРОБКА ПРОГРАМНОГО МОДУЛЯ

Процес розробки програмного модуля умовно можна поділити на два етапи: реалізація алгоритму шифрування текстової інформації DES та реалізація алгоритму шифрування графічної інформації LSB.

Повний лістинг програмного модуля наведено в Додатку А.

4.1 Реалізація алгоритму DES

4.1.1 Розробка інтерфейсу користувача

Для реалізації алгоритму DES було розроблено інтерфейс користувача, наведений на рисунку 4.1.

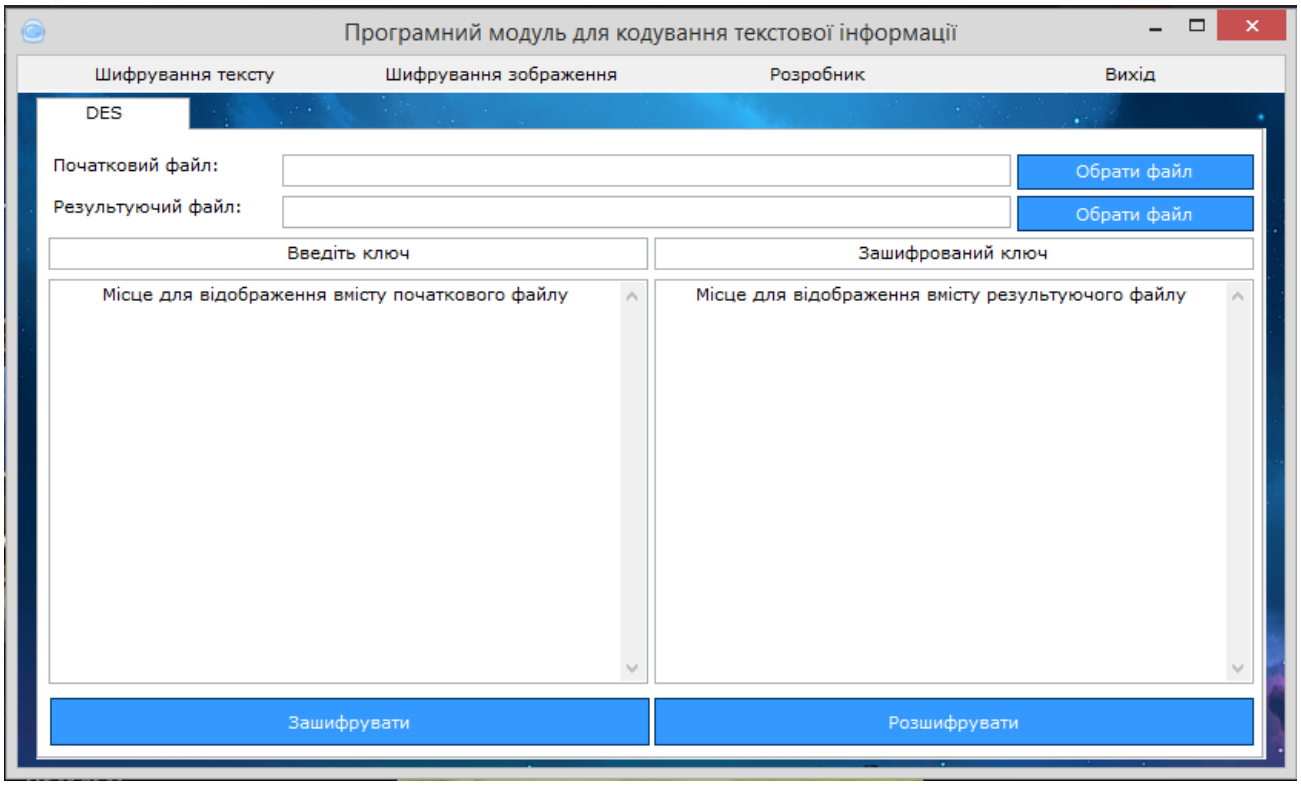


Рисунок 4.1 – DES інтерфейс користувача

Інтерфейс користувача для реалізації алгоритму DES містить наступні керуючі елементи:

– Елемент керування "шифрованиеDesToolStripMenuItem" – для вибору методу шифрування DES.

– Елемент керування "шифрованиеLBSToolStripMenuItem" – для вибору методу шифрування LSB.

– Елемент керування "textBox3" – для відображення шляху до початкового файлу.

– Елемент керування "textBox4" – для відображення шляху до результуючого файлу.

– Елемент керування "оПрограммеToolStripMenuItem" – для отримання інформації про розробника.

– Елемент керування "выходToolStripMenuItem" – для виходу із програмного модуля.

– Елемент керування "button3" – для вибору початкового файлу.

– Елемент керування "button4" – для вибору результуючого файлу.

– Елемент керування "textBoxEncodeKeyWord" – для вводу ключа шифрування.

– Елемент керування "textBoxDecodeKeyWord" – для виводу закодованого ключа.

– Елемент керування "textBox1" – для відображання змісту початкового файлу.

– Елемент керування "textBox2" – для відображання змісту зашифрованого файлу.

– Елемент керування "buttonEncrypt" – для шифрування початкового файлу.

– Елемент керування "buttonDecrypt" – для розшифрування кінцевого файлу.

4.1.2 Розробка програмного модуля DES.

В програмному модулі DES реалізовано наступні функції та методи:

– Метод "private string StringToRightLength(string input)" - метод, який доводить рядок до такого розміру, щоб ділився на sizeOfBlock. Розмір збільшується за допомогою додавання до вихідної рядку символу "#".

– Метод "private void CutStringIntoBlocks(string input)" - метод, який розбиває рядок в звичайному (символьному) форматі на блоки.

– Метод "private void CutBinaryStringIntoBlocks(string input)" - метод, який розбиває рядок в двійковому форматі на блоки.

– Метод "private string StringToBinaryFormat(string input)" - метод, що переводить рядок в двійковий формат.

– Метод "private string CorrectKeyWord(string input, int lengthKey)" - метод, який доводить довжину ключа до потрібної.

– Метод "private string EncodeDES_One_Round(string input, string key)" - перший етап шифрування алгоритмом DES.

– Метод "private string DecodeDES_One_Round(string input, string key)" – перший етап розшифрування алгоритмом DES.

– Метод "private string XOR(string s1, string s2)" – операція XOR двох рядків з двійковими даними.

– Метод "private string f(string s1, string s2)" – реалізація шифруючої функції f. Вирішено використовувати в якості неї також логічну операцію XOR.

– Метод "private string KeyToNextRound(string key)" - обчислення ключа для наступного етапу шифрування DES.

– Метод "private string KeyToPrevRound(string key) " - обчислення ключа для попереднього етапу розшифрування DES.

– Метод "private string StringFromBinaryToNormalFormat(string input)" - метод, що переводить рядок з двійковими даними в символьний формат.

– Метод "private void buttonEncrypt_Click(object sender, EventArgs e)" - функціонал кнопки "Зашифрувати".

– Метод "private void buttonDecipher_Click(object sender, EventArgs e)" - функціонал кнопки " Розшифрувати ".

4.2 Реалізація алгоритму LSB стеганографії

4.2.1 Розробка візуального інтерфейсу

Для реалізації алгоритму LSB стеганографії було розроблено візуальний інтерфейс, наведений на рисунку 4.2.

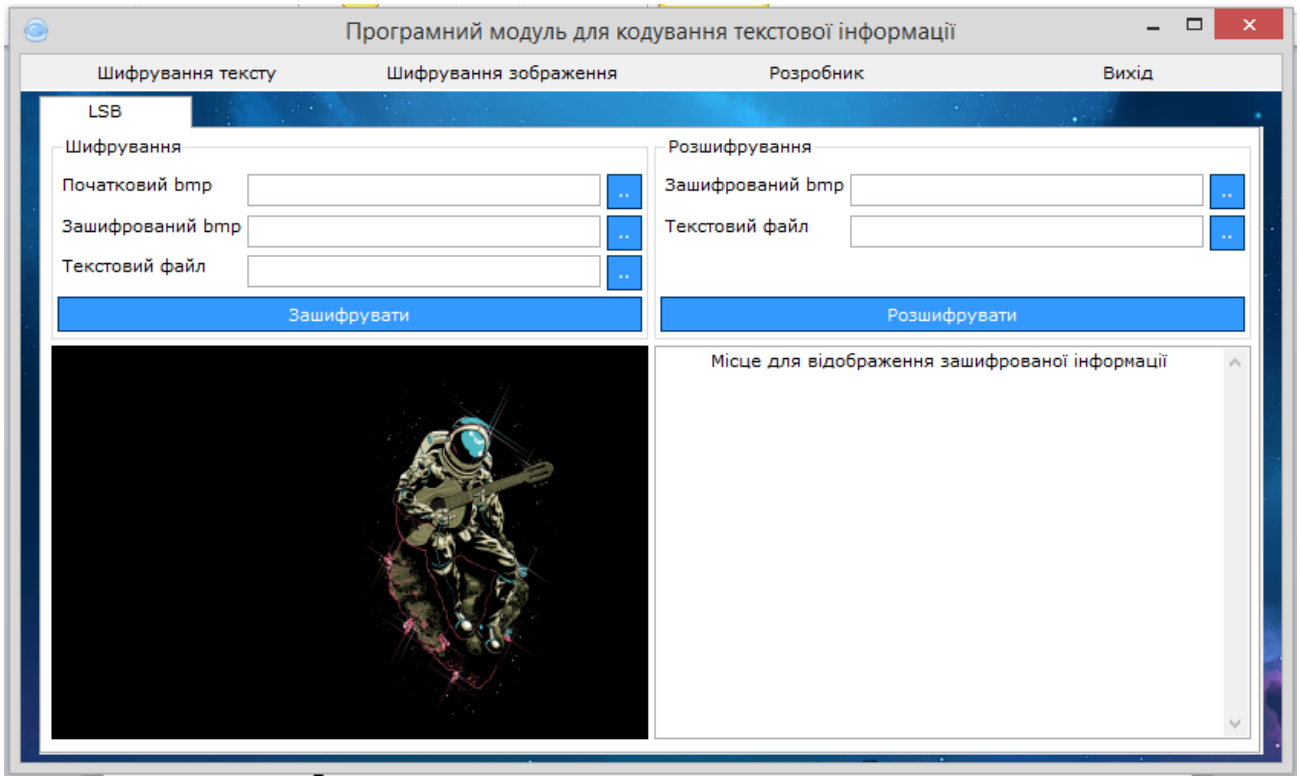


Рисунок 4.2 – LSB інтерфейс користувача

Інтерфейс користувача для реалізації алгоритму LBS-стеганографії містить наступні керуючі елементи:

– Елемент керування "button5" – для вибору початкового bmp-файлу.

– Елемент керування "textBox5" – для відображення шляху до початкового bmp-файлу.

– Елемент керування "button6" – для вибору bmp-файлу, який буде зашифрований.

– Елемент керування "textBox6" – для відображення шляху до bmp-файлу, який буде зашифрований.

– Елемент керування "button7" – для вибору текстового файлу.

– Елемент керування "textBox7" – для відображення шляху до текстового файлу.

– Елемент керування "button1" – для початку процесу шифрування bmp-файлу.

– Елемент керування "button9" – для вибору зашифрованого bmp-файлу.

– Елемент керування "textBox10" – для відображення шляху до зашифрованого bmp-файлу.

– Елемент керування "button8" – для вибору зашифрованого текстового файлу.

– Елемент керування "textBox9" – для відображення шляху до зашифрованого текстового файлу.

– Елемент керування "button2" – для розшифрування bmp-файлу.

– Елемент керування "pictureBox" – для відображення bmp-файлу.

– Елемент керування "textBox8" – для відображення зашифрованої інформації.

4.2.2 Дослідження структури графічного формату BMP

Найпростіший спосіб для представлення зображення є набір пікселів (тобто точок), кожен з яких може бути різного кольору. Для чорно-білих зображень, таким чином, потрібно 1 біт на піксель, де "0" може представляти чорний і "1" може представляти білий, як показано нижче на рис. 4.3.

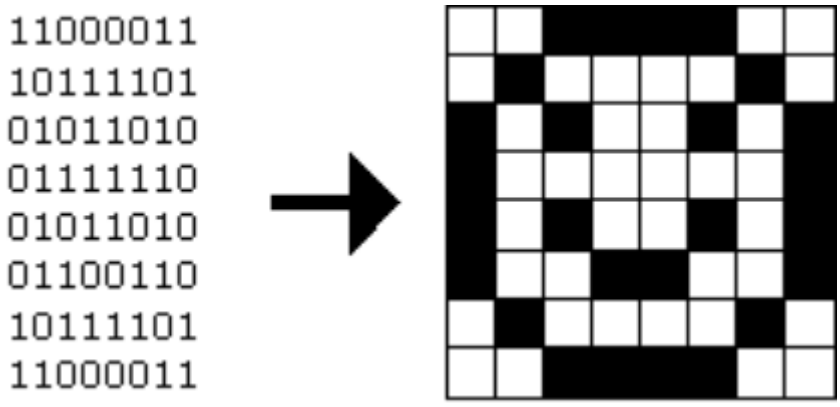


Рисунок 4.3 – Приклад однобітного файлу "smiley.bmp"

За цим принципом, зображення - це просто карта з бітів (тобто відображення бітів). Для отримання більш барвистих зображень вам просто потрібно більше бітів на піксель. Формат файлу (наприклад, GIF), який підтримує "8-бітний колір" використовує 8 біт на піксель. Формат файлу (наприклад, BMP, JPG, PNG або), який підтримує "24-бітний колір" використовує 24 біти на піксель. (BMP фактично підтримує 1-, 4-, 8-, 16-, 24- і 32-бітний колір.)

24-бітний BMP, визначає по 8 біт для позначення кількості червоного кольору, 8 біт для позначення кількості зеленого кольору, і 8 біт для позначення кількості синього кольору пікселів. Якщо використана модель RGB, то у вас є: червоний, зелений, синій.

Якщо значення R, G, і B деякого пікселя в BMP, скажімо, 0xff, 0x00 і 0x00 в шістнадцятковій системі числення, то піксель буде чисто червоним, так як 0xff (інакше відоме як 255 в десятковій системі) означає "багато червоного" в той час як 0x00 і 0x00 означають “немає зеленого” і “немає синього" відповідно.

Файл - це просто послідовність бітів, розташованих в деякому порядку. 24-бітний BMP-файл це тільки послідовність бітів, (майже) кожні 24 з яких визначають колір якогось пікселя. Але файл BMP також містить деякі "метадані" - інформацію, такі як висота і ширина зображення. Ці метадані зберігаються на початку файлу у вигляді двох структур даних, зазвичай званих "заголовками" (рис. 4.4).

Першим з цих заголовків є BITMAPFILEHEADER, довжиною 14 байт. Другий є BITMAPINFOHEADER, довжиною 40 байт. Відразу після цих заголовків є фактичне растрове зображення: масив байтів, трійки з яких представляють колір пікселя. (В 1, 4 і 16-біт BMP, але не 24- або 32,у них є додатковий заголовок відразу після BITMAPINFOHEADER називається RGBQUAD, масив, який визначає "значення інтенсивності" для кожного з кольорів в палітрі пристрою.) Проте, BMP зберігає ці трійки в зворотному напрямку (тобто як BGR), з 8 бітами на синій, 8 бітами на зелений і 8 бітами на червоний кольори.

offset	type	name	
0	WORD	bfType	BITMAPFILEHEADER
2	DWORD	bfSize	
6	WORD	bfReserved1	
8	WORD	bfReserved2	
10	DWORD	bfOffBits	
14	DWORD	biSize	BITMAPINFOHEADER
18	LONG	biWidth	
22	LONG	biHeight	
26	WORD	biPlanes	
28	WORD	biBitCount	
30	DWORD	biCompression	
34	DWORD	biSizeImage	
38	LONG	biXPelsPerMeter	
42	LONG	biYPelsPerMeter	
46	DWORD	biClrUsed	
50	DWORD	biClrImportant	
54	BYTE	rgbtBlue	RGBTRIPLE
55	BYTE	rgbtGreen	
56	BYTE	rgbtRed	
57	BYTE	rgbtBlue	RGBTRIPLE
58	BYTE	rgbtGreen	
59	BYTE	rgbtRed	
...			
243	BYTE	rgbtBlue	RGBTRIPLE
244	BYTE	rgbtGreen	
245	BYTE	rgbtRed	

Рисунок 4.4 – Структура файлу формату BMP

Іншими словами , можна перетворити 1-бітне зображення, наведене на рисунку 4.4, у 24-бітний, замінюючи чорний червоним, 24-бітний BMP буде зберігати цей бітовий масив таким чином, де 0000ff позначає червоний і ffffff позначає білий. На рис. 4.5 виділено червоним кольором всі екземпляри 0000ff.

```
0000036: ffffff ffffff 0000ff 0000ff 0000ff 0000ff ffffff ffffff  ........................
000004e: ffffff 0000ff ffffff ffffff ffffff ffffff 0000ff ffffff  ........................
0000066: 0000ff ffffff 0000ff ffffff ffffff 0000ff ffffff 0000ff  ........................
000007e: 0000ff ffffff ffffff ffffff ffffff ffffff ffffff 0000ff  ........................
0000096: 0000ff ffffff 0000ff ffffff ffffff 0000ff ffffff 0000ff  ........................
00000ae: 0000ff ffffff ffffff 0000ff 0000ff ffffff ffffff 0000ff  ........................
00000c6: ffffff 0000ff ffffff ffffff ffffff ffffff 0000ff ffffff  ........................
00000de: ffffff ffffff 0000ff 0000ff 0000ff 0000ff ffffff ffffff  ........................
```

Рисунок 4.5 – Приклад 24-бітного файлу "smiley.bmp"

Цифра в шістнадцятковій системі числення представляє 4 біта. Відповідно, ffffff в шістнадцятковому варіанті фактично означає 111111111111111111111111 в двійковій системі.

В крайньому лівому стовпчику вище є адреси у файлі, які еквівалентні зміщенню від першого байту файлу, всі вони наведені в шістнадцятковій системі числення. Важливо пам’ятати, що 00000036 записане у шістнадцятковому форматі, а в десятковій системі воно буде 54. Таким чином, це буде байт 54 з рис.4.3. В 24-бітних BMP файлах перші 14 + 40 = 54 байт заповнюються метаданими.

Якщо рис. 4.3 фактично би містив ASCII символи, вони б знаходилися в крайньому правому стовпчику в xxd замість всіх точок.

Так, рис. 4.3 є розміром 8 пікселів в ширину та 8 пікселів у висоту, і це 24-бітний BMP (кожен з пікселів представлений за допомогою 24 ÷ 8 = 3 байтів). Кожен рядок (так званий "Scanline"), таким чином, займає (8 пікселів) × (3 байти на піксель) = 24 байти, яке кратне 4. Виявляється, що BMP зберігається трохи по-іншому, якщо число байтів в рядку не є, кратним 4. В "small.bmp" (рис. 4.6), наприклад, ще один 24-бітний BMP, зелене поле, це 3 пікселі в ширину на 3

пікселі в довжину. Якщо переглядати його у програмі перегляду зображень (як за допомогою подвійного кліку), можна побачити, що воно нагадує показане нижче, хоча і набагато менше.

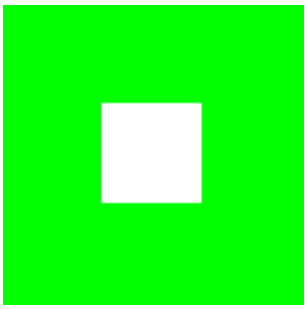


Рисунок 4.6– Приклад 24-бітного файлу "small.bmp"

Кожен рядок в "small.bmp", таким чином, займає (3 пікселі) × (3 байти на піксель) = 9 байт, що не є кратним 4. І таким чином рядок забивають такою кількістю нулів, щоб продовжити довжину рядку до числа кратного 4. Іншими словами, необхідно між 0 і 3 байтами заповнення для кожного рядка в 24-бітному форматі BMP. У разі "small.bmp", необхідні 3 байти нулів, так як (3 пікселі) × (3 байти на піксель) + (3 байти заповнення) = 12 байт, які дійсно кратні 4.

```
0000036: 00ff00 00ff00 00ff00 000000  ............
0000042: 00ff00 ffffff 00ff00 000000  ............
000004e: 00ff00 00ff00 00ff00 000000  ............
```

Рисунок 4.7– Структура 24-бітного файлу "small.bmp"

4.2.3 Розробка програмного модуля LSB

LSB (Least Significant Bit, найменший значущий біт) - суть цього методу полягає в заміні останніх значущих бітів в контейнері (зображення, аудіо або відеозапису) на біти приховуваного повідомлення. Різниця між порожнім і заповненим контейнерами повинна бути не відчутна для органів сприйняття людини. Для шифрування/дешифрування буде використовуватися bmp-файл, який

не містить палітру. В такому bmp-файлі кожні 3 байта визначають 3 кольору пікселя. Схема методу наведена на рис. 4.8.

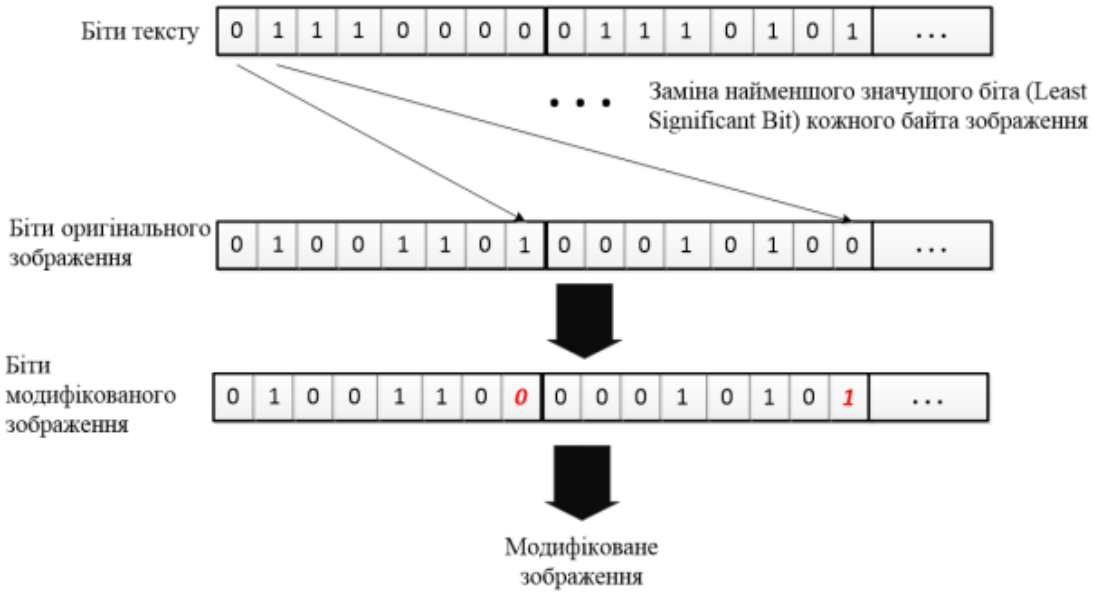


Рисунок 4.8 – Схема методу LSB шифрування

Програмний модуль DES містить наступні функції та методи:

– Метод "private BitArray ByteToBit(byte src)" – метод перетворення байта в біти.

– Метод "private byte BitToByte(BitArray scr)" – метод перетворення бітів в байт.

– Метод "byte [] Symbol = Encoding.GetEncoding(1251).GetBytes" - метод, який записує в піксель 0.0 ознаку зашифрованого файлу.

– Метод "private bool isEncryption(Bitmap scr)" - метод перевірки ознаки, того, що в файлі є інформація записана в bmp файл.

– Метод "private void WriteCountText(int count, Bitmap src)" - метод запису розміру текстової інформації.

– "Метод private int ReadCountText(Bitmap src)" - метод читання розміру текстової інформації.

Розташування в інформації в bmp-зображенні буде наступним:

– Піксель 0,0: ознака того, що в файлі є текстова інформація. За ознаку використовується символ "/".

– Пікселі 0.1 - 0.3: розмір текстової інформації, записаної в файл.

– Пікселі 0.4 і до кінця файлу: текстова інформація.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Загальні питання з охорони праці

В законі України «Про охорону праці» визначається [10], що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

Сутність охорони праці полягає у визначенні можливих небезпечних і шкідливих виробничих факторів, що можуть проявитися при проведенні запланованих для виконання робіт; прогнозуванні моментів прояву зазначених факторів; проведенні необхідних профілактичних заходів.

Державна політика в галузі охорони праці базується на головному принципі – пріоритету життя і здоров'я працівників.

Підвищення рівня промислової безпеки шляхом забезпечення суцільного технічного контролю за станом виробництв, технологій та продукції позитивно впливає на ефективність виробництва. На це вказує те, що травматизм визначає істотну частину непродуктивних втрат робочого часу, а боротьба з травматизмом, крім гуманістичного спрямування, має чітко виражений економічний аспект.

При виконанні роботи безпосередньо на персональних комп’ютерах та з використанням друкувальної, копіювальної та іншої офісної техніки характерні такі небезпечні фактори, як:

– шум і вібрація, джерелом яких є комп’ютери, офісна техніка;

– забруднення атмосфери шкідливими речовинами (озон, оксиди азоту, пил);

– широке використання електричного струму: комп’ютерами, в штучному освітленні;

– електромагнітні поля, інфрачервоне та іонізуюче випромінювання;

– тепловиділення;

– статична електрика.

5.1.1 Вимоги до приміщень

Геометричні розміри приміщення зазначені в табл. 5.1.

Таблиця 5.1 – Розміри приміщення

Найменування	Значення
Довжина, м	6
Ширина, м	5
Висота, м	3
Площа, $м^2$	25
Об’єм, $м^3$	75

Згідно з ДСН 3.3.6.042-99 [11] розмір площі для одного робочого місця оператора персонального комп’ютера має бути не менше 6 кв. м, а об’єм – не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

5.1.2 Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за ДСанПіН 3.3.2.007-98 [12] (табл. 5.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 5.2 - Характеристики робочого місця

Найменування параметра	Фактичне Значення	Нормативне Значення
Висота робочої поверхні, мм	700	680 ÷ 800
Висота простору для ніг, мм	650	не менше 600
Ширина простору для ніг, мм	540	не менше 500
Глибина простору для ніг, мм	660	не менше 650

Продовження таблиці 5.2

Висота поверхні сидіння, мм	420	400 ÷ 500
Ширина сидіння, мм	410	не менше 400
Глибина сидіння, мм	420	не менше 400
Висота поверхні спинки, мм	500	не менше 300
Ширина опорної поверхні спинки, мм	400	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	750	700 ÷ 800

5.2 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

5.2.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 4.3). Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання НПАОП 0.00-7.15-18 [13], яке встановлює вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Основними робочими характеристиками персонального комп'ютера є:

- робоча напруга *U=+220В +-5%;*
- робочий струм *I=2А*;
- споживана потужність *Р=350 Вт*.

Таблиця 5.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна Оцінка	Нормативні Документи
1	2	3	4
Фізичні:			
підвищена або знижена вологість повітря	-//-	3	[15]
підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	3	[14] [14]
Психофізіологічні:			
1	2	3	4
нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- формулювання теми; - пошук інформацію про предметну область; - пошук інформації про наявні аналоги; - проектування структур та алгоритмів; - виконання роботи; - оформлення записки.	4	[13] [12]
фізичні (статичне - сидіння)	порушення умов організації робочого часу (безперервна робота)	2	[13] [12]

Робочі місця мають відповідати вимогам державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [12].

5.2.2 Пожежна безпека

Висока щільність елементів в електронних схемах призводить до значного підвищення температури окремих вузлів (80…100℃). При проходженні електричного струму по провідниках і деталей виділяється тепло, що в умовах їх високої щільності може привести до перегріву, і може служити причиною запалювання ізоляційних матеріалів. Слабкий опір ізоляційних матеріалів дії

температури може викликати порушення ізоляції і привести до короткого замикання між струмоведучими частинами обладнання (шини, електроди).

Для гасіння пожеж в офісному приміщенні пропонується використовувати порошкові або вуглекислотні вогнегасники, так як вони є універсальними.

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), надійно захищені діелектричними щитками та/або сітками з метою недопущення потрапляння працівника під напругу.

В приміщенні наявна затверджена "План-схема евакуації з кабінету (приміщення)".

Горючими матеріалами в приміщенні, де розташовані ЕОМ, є:

– поліамід – матеріал корпусу мікросхем, горюча речовина, температура самозаймання 420 ℃;

– полівінілхлорид – ізоляційний матеріал, горюча речовина, температура запалювання 335 ℃, температура самозаймання 530 ℃;

– склотекстоліт ДЦ – матеріал друкарських плат, важкогорючий матеріал, показник горючості 1.74, не схильний до температурного самозаймання;

– пластикат кабельний №489 – матеріал ізоляції кабелів, горючий матеріал, показник горючості більше 2.1;

– деревина – будівельний і обробний матеріал, з якого виготовлені меблі, горючий матеріал, показник горючості більше 2.1, температура запалювання 255 ℃, температура самозаймання 399 ℃.

Простори усередині приміщень в межах, яких можуть утворюватися або знаходиться пожежонебезпечні речовини і матеріали відповідно до ДБН В.2.5-28:2018 [14] відносяться до категорії "В" (пожежонебезпечної). Це обумовлено тим, що в приміщенні знаходяться тверді горючі та важкозаймисті речовини та матеріали.

Причинами можливого загоряння і пожежі можуть бути:

– несправність електроустановки;

– конструктивні недоліки устаткування;

– коротке замикання в електричних мережах;

– запалювання горючих матеріалів, що знаходяться в безпосередній близькості від електроустановки.

Продуктами згорання, що виділяються на пожежі, є: окис вуглецю; сірчистий газ; окис азоту; синильна кислота; акромін; фосген; хлор і ін. При горінні пластмас, окрім звичних продуктів згорання, виділяються різні продукти термічного розкладання: хлорангідридні кислоти, формальдегіди, хлористий водень, фосген, синильна кислота, аміак, фенол, ацетон, стирол.

5.2.3 Електробезпека

Виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК, укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється – заземлюючий провідник.

5.3 Гігієнічні вимоги до параметрів виробничого середовища

Для зменшення негативного впливу ЕОМ з ВДТ і ПП планують комплекс медико-гігієнічних, адміністративно-технічних і ергономічних заходів:

– контроль за конструкцією, добрим станом і функціями ВДТ;

– створення оптимальних умов для праці у виробничому приміщенні (мікроклімат, освітлення, захист від опромінювання комп’ютера, іонізації повітря тощо);

– відповідність місця праці рекомендаціям ергономіки та гігієни;

– раціональний режим праці;

– підвищення опору організму користувачів комп’ютера до дії несприятливих чинників (антистресова дія, аеробіка та спеціальні фізичні вправи, психологічні та соціальні заходи, профілактичне харчування);

– диспансерне медико-гігієнічне обслуговування з цілеспрямованим проведенням оздоровчих (наприклад, корекція зору) і профілактичних заходів;

– особиста участь працівника у догляді за своїм здоров’ям;

– вентиляція та кондиціонування приміщень установи

Для забезпечення оптимальних умов мікроклімату у приміщеннях з ВДТ використовують системи вентиляції та кондиціювання повітря, а також природне провітрювання.

5.3.1 Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря. В даному приміщенні проводяться роботи, що виконуються сидячи і не потребують динамічного фізичного напруження, то для нього відповідає категорія робіт Іа. Отже оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають ДСН 3.3.6.042-99 [11] і наведені в табл. 4.4:

Таблиця 5.4 – Нормативні параметри мікроклімату для приміщень з ВДТ

Період Року	Категорія Робіт	Температура C^0	Відносна вологість %	Швидкість руху повітря, м/с
Холодна	Легка-1а	22-24	40-60	0,1
Тепла	Легка-1а	23-25	40-60	0,1

5.3.2 Освітлення

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше – 1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} / \frac{1}{10}\right) \cdot S_n, \tag{5.1}$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$$S_n = a \cdot b = 5 \cdot 5 = 25\,\text{м}^2;$$

$$S = \frac{1}{8} \cdot 25 = 3{,}125\ \text{м}^2.$$

Приймаємо 2 вікна площею S=1,6 м^2 кожне.

Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 5 м, ширина 5 м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 5400 лм кожна. Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (5.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \qquad (5.2)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 25 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575;

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400 лм (для ЛБ-80).

Підставивши числові значення у формулу (5.2), отримуємо:

$$n = \frac{300 \cdot 25 \cdot 1{,}1 \cdot 1{,}5}{5400 \cdot 0{,}575 \cdot 2} \approx 2{,}0.$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

5.4 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти), тобто при $V_{приміщення} > 40$ м^3 на одного працюючого допускається природна вентиляція. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в

ДСанПІН 3.3.2.007-98 [12]. Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

5.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

1. Заходи безпеки під час експлуатації персонального комп'ютера та периферійних пристроїв передбачають:

– правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;

– експлуатацію сертифікованого обладнання;

– дотримання заходів електробезпеки;

– забезпечення оптимальних параметрів мікроклімату;

– забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);

– облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря:

а) якщо об'єм приміщення 20 $м^3$, то потрібно подати не менш як 30 $м^3$/год повітря;

б) якщо об'єм приміщення у межах від 20 до 40 $м^3$, то потрібно подати не менш як 20 $м^3$/год повітря;

в) якщо об'єм приміщення становить понад 40 $м^3$, допускається природна вентиляція, у випадку, коли немає виділення шкідливих речовин.

2. Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

– постійно стежити за справним станом електромережі, розподільних щитків, вимикачів, штепсельних розеток, ламповихпатронів, а також мережевих кабелів живлення, за допомогою яких електроприлади під'єднують до електромережі;

– постійно стежити за справністю ізоляції електромережі та мережевих кабелів, не допускаючи їхньої експлуатації з пошкодженою ізоляцією;

– не тягнути за мережевий кабель, щоб витягти вилку з розетки;

– не закривати меблями, різноманітним інвентарем вимикачі, штепсельні розетки;

– не підключати одночасно декілька потужних електропристроїв до однієї розетки, що може викликати надмірне нагрівання провідників, руйнування їхньої ізоляції, розплавлення і загоряння полімерних матеріалів;

– не залишати включені електроприлади без нагляду;

5.5.1 Розрахунок захисного заземлення (забезпечення електробезпеки будівлі)

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом згідно з наказом від 1 липня 2016 року N 204 [15], приміщення в якому проводяться всі роботи відноситься до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 36 В, 220 В, та 360 В. Опір контуру заземлення повинен мати не більше 4 Ом.

Послідовність розрахунку.

1. Визначається необхідний опір штучних заземлювачів $R_{шт.з}$:

$$R_{шт.з} = \frac{R_д \cdot R_{пр.з}}{R_{пр.з} - R_д}, \tag{5.3}$$

де $R_{пр.з}$ – опір природних заземлювачів;

$R_д$ – допустимий опір заземлення.

Якщо природні заземлювачі відсутні, то $R_{шт.з} = R_д$.

Підставивши числові значення у формулу (5.3), отримуємо:

$$R_{шт.з} = \frac{4 \cdot 40}{40 - 4} \approx 4\,Ом.$$

2. Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо ρ=40 Ом·м (табличне значення).

3. Розрахунковий питомий опір ґрунту, $P_{розр}$, Ом·м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $P_{розр.г}$, Ом·м за формулою (5.4):

$$P_{розр.} = \Psi * p \qquad (5.4)$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів I кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $P_{розр.в}$=1,7 і горизонтальних $P_{розр.г}$ =5,5 Ом·м.

$$\text{Р}_{\text{озр.в}} = 1,\ 7 * 40 = 68\ \text{Ом/м};$$
$$\text{Р}_{\text{озр.г}} = \cdot 5,5 * 40 = 220\ \text{Ом/м}.$$

4. Розраховується опір розтікання струму вертикального заземлювача Rв, Ом, за (5.5).

$$R_B = \frac{P_{розр.в}}{2 * \pi * 1_B} * \left(\ln \frac{2 * 1_B}{d_{CT}} + \frac{1}{2} * \ln \frac{4 * t + 1_B}{4 * t - 1_B} \right) \qquad (5.5)$$

де $l_в$ – довжина вертикального заземлювача (для труб – 2-3 м; $l_в$=3 м);

$d_{ст}$ – діаметр стержня (для труб – 0,03-0,05 м; $d_{ст}$=0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за

формулою (5.6):

$$t = h_B + \frac{1_B}{2} \qquad (5.6)$$

де $h_в$ – глибина закладання вертикальних заземлювачів (0,8 м); тоді $t = 0.8 + \frac{3}{2} = 2{,}3м$.

$$R_B = \frac{68}{2 * \pi * 3} * \left(\ln \frac{2 * 3}{0{,}05} + \frac{1}{2} * \ln \frac{4 * 2{,}3 + 3}{4 * 2{,}3 - 3} \right) = 18{,}5Ом .$$

5. Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання ηв за формулою (5.7):

$$n = \frac{2 * R_B}{R_д} = \frac{2 * 18{,}5}{4} = 9{,}25 . \qquad (5.7)$$

6. Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_в$, шт. за формулою (5.8):

$$n_B = \frac{2 * R_B}{R_д * \eta_B} = \frac{2 * 18{,}5}{4 * 0{,}57} = 16{,}2 \approx 16 . \qquad (5.8)$$

7. Визначається довжина з'єднувальної стрічки горизонтального заземлювача lc, м за формулою (5.9):

$$l_c = 1{,}05 * L_B * (n_B - 1) . \qquad (5.9)$$

де $l_в$ – відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м);

$n_в$ – необхідна кількість вертикальних заземлювачів.

$$l_c = 1{,}05 * 3 * (16 - 1) \approx 48 м .$$

8. Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки) $R_г$, Ом за формулою (5.10):

$$R_Г = \frac{P_{розр.г}}{2 * \pi * l_c} * \ln \frac{2 * l_c^2}{d_{ст} * h_Г} \qquad (5.10)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ =0,95b, b = 0,15 м;

$h_г$ – глибина закладання горизонтальних заземлювачів (0,5 м);

l_c – довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м.

$$R_Г = \frac{220}{2 * \pi * 48} * \ln \frac{2 * 48^2}{0{,}95 * 0{,}15 * 0{,}5} = 8{,}1 Ом .$$

9. Визначається коефіцієнт використання горизонтального заземлювача $\eta_с$. відповідно до необхідної кількості вертикальних заземлювачів $n_в$. Коефіцієнт використання з'єднувальної смуги $\eta_с$=0,3 (табличне значення).

10. Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги за формулою (5.11):

$$R_{заг} = \frac{R_B * R_Г}{R_B * \eta_c + R_Г * n_B * \eta_B} \qquad (5.11)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг}$ Ом, а саме:

$$R_{заг} = \frac{18{,}5 * 8{,}1}{18{,}5 * 0{,}3 + 8{,}1 * 16 * 0{,}57} = 1{,}9 \le R_д .$$

Висновки до розділу 5

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуто заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Технічним завданням дипломного проекту було створення програмного модуля для кодування текстової інформації з використанням DES-шифрування та LSB-стеганографії. Так як в процесі розробки використовувався ПК, то аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для персонального комп'ютера, на якому було розроблено модуль.

ВИСНОВКИ

Для вирішення технічного завдання дипломного проекту було проведено дослідження основних алгоритмів шифрування тексту, які швидко та надійно шифрують тестову інформацію. Проведено інтегрування сучасних методів шифрування тексту до зображення, які дозволять приховати наявність зашифрованого тексту у зображенні. При порівнянні алгоритмів та методів основний акцент робився на надійності шифрування тексту та непомітному приховуванні його в зображенні.

Для шифрування текстової інформації було обрано метод DES, через його відносної простоти та високої стійкості до взламу.

Для шифрування графічної інформації було обрано метод LSB, який не вимагає значних витрат часу і сил для реалізації та не викликає підозр.

Результати дослідження дозволили отримати програмний модуль, який надає можливість користувачеві швидко та зручно приховати текстову інформацію для її подальшої передачі (через змінні носії, локальну або глобальну мережу).

Програмний модуль дипломного проекту функціонально не уступає конкурентам галузі та має важливу перевагу – наявність подвійного шифрування.

Для подальшого розвитку можна впровадити додаткові алгоритми шифрування тексту, покращити інтерфейс та інтегрувати модуль на web та android платформи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Порівняння настільних програм для шифрування [Електронний ресурс] // Хабр. URL: https://habr.com/ru/company/cybersafe/blog/252561/

2. Захарченко М. В. Асиметричні методи шифрування в телекомунікаціях: навч. посіб. / М. В. Захарченко, О. В. Онацький, Л. Г. Йона, Т. М. Шинкарчук. – Одеса: ОНАЗ ім. О. С. Попова, 2011. – 184 с. – (Криптографічні методи захисту інформації в телекомунікаційних системах та мережах: модуль 2 з дисципліни „Захист інформації в телекомунікаційних системах та мережах”).

3. Метод вибору оптимального алгоритму криптографічного захисту інформації / В. А. Висоцька, О. Р. Гарасим // Вісн. Нац. ун-ту "Львів. політехніка". - 2010. - № 673. - С. 220-232. - Бібліогр.: 20 назв. - укр.

4. Шифр Цезаря [Електронний ресурс] // Вікіпедія: вільна енциклопедія. URL: https://ru.wikipedia.org/wiki/Шифр_Цезаря

5. ШифрВіженера [Електронний ресурс] // Вікіпедія: вільна енциклопедія. URL: https://ru.wikipedia.org/wiki/Шифр_Виженера

6. Алгоритми шифрування DES та AES [Електронний ресурс] // Интитуит: Національний відкритий університет. URL: https://www.intuit.ru/studies/courses/691/547/lecture/12377

7. Шелест М.Є., Андреев В.І. Комп'ютерна стеганографія та її можливості [Текст] / М.Є. Шелест, В.І. Андреев// Сучасна спеціальна техніка. — 2011. — № 1. — С. 98—104.

8. Про один метод стеганографічного аналізу контейнерів-зображень / І.В. Швідченко // Искусственный интеллект. — 2013. — № 3. — С. 554–563.

9. Шостак Н. В., Безрук В. М., Астраханцев А. А. Вибір переважного алгоритму вбудовування цифрових водяних знаків в відеофайли. - С. 167-173.

10. Закон України "Про охорону праці". Вводиться в дію Постановою ВР № 2695-XII від 14.10.92, ВВР, 1992, № 49, ст.669. - Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/2694-12

11. Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. Постанова N 42 від 01.12.99. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/va042282-99

12. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98. Затверджено Постановою Головного державного санітарного лікаря України 10 грудня 1998 р. N 7. Режим доступу: www. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98

13. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. За № 508/31960. Режим доступу: www. URL: https://zakon.rada.gov.ua/laws/show/z0508-18

14. ДБН В.2.5-28:2018 «Природне і штучне освітлення». Режим доступу: www. URL:http://www.minregion.gov.ua/wp-content/uploads/2018/12/V2528-1.pdf

15. Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом. Наказ від 1 липня 2016 року N 204. Режим доступу: www. URL:http://epicentre.co.ua/dstu/doc28522.html

Додаток А.

Комп’ютерна презентація дипломного проекту

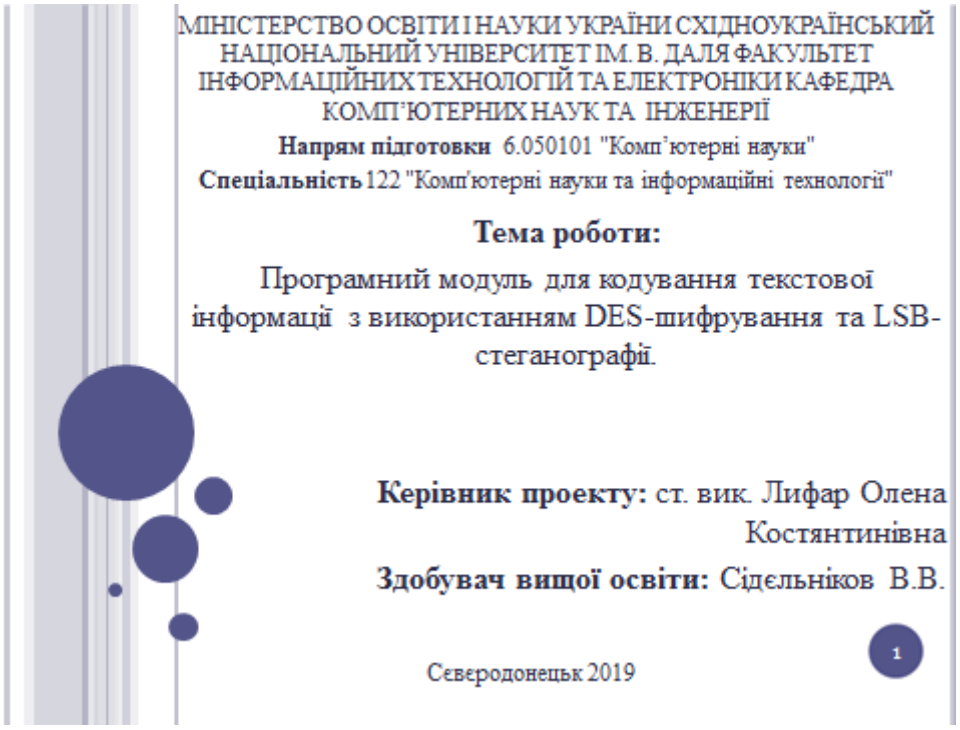


Рисунок А.1 – Титульный слайд

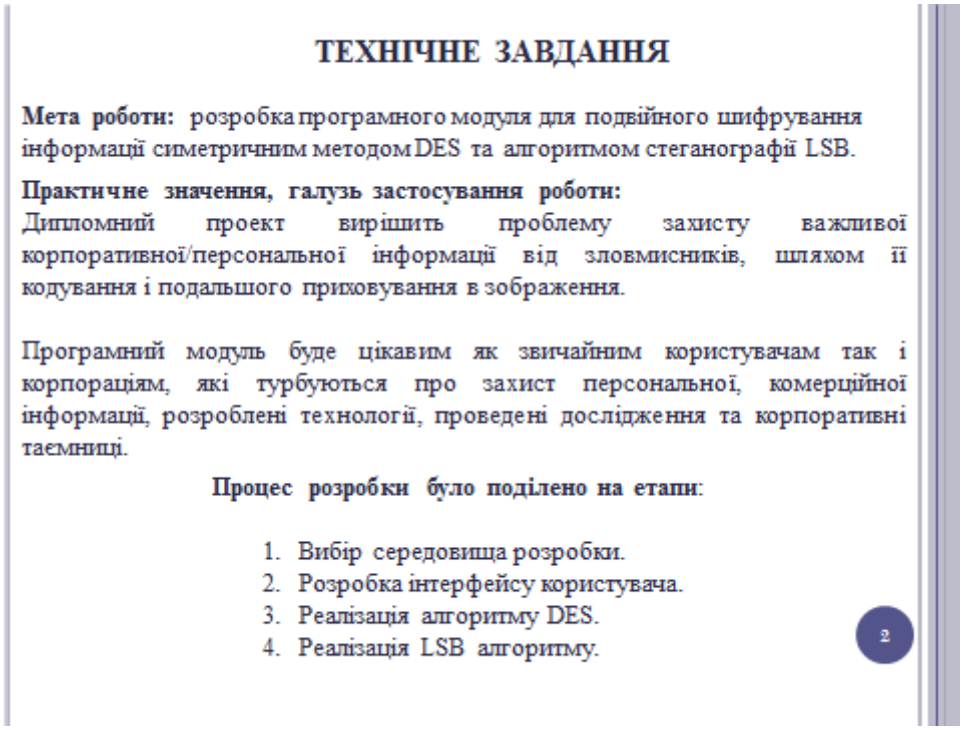


Рисунок А.2 – Слайд "Технічне завдання"

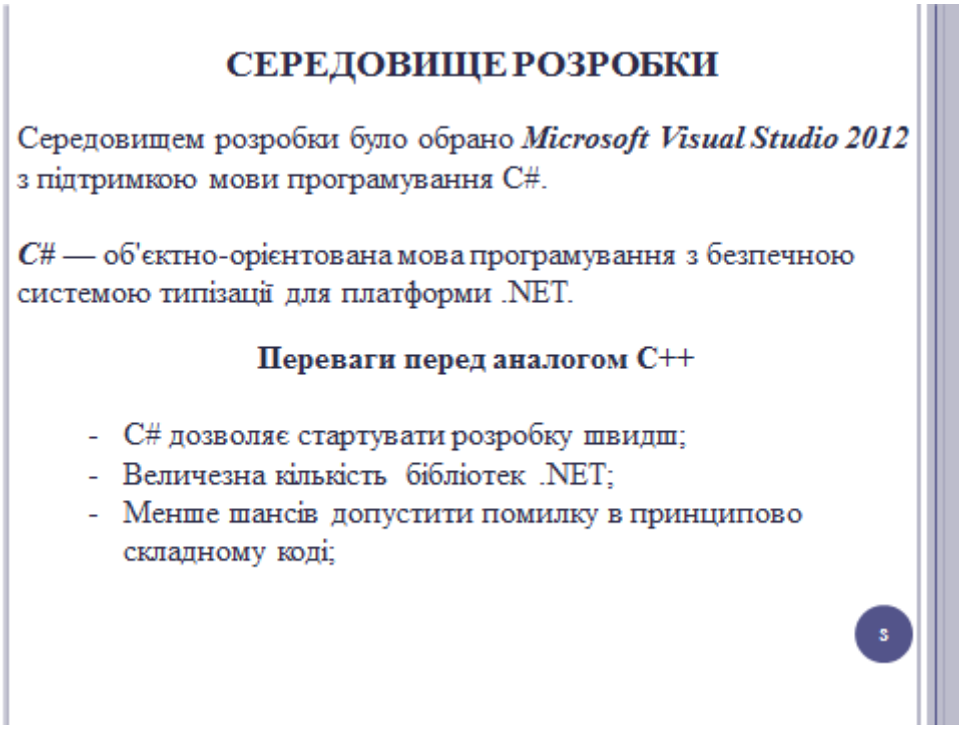


Рисунок А.3 - Слайд "Середовище розробки"

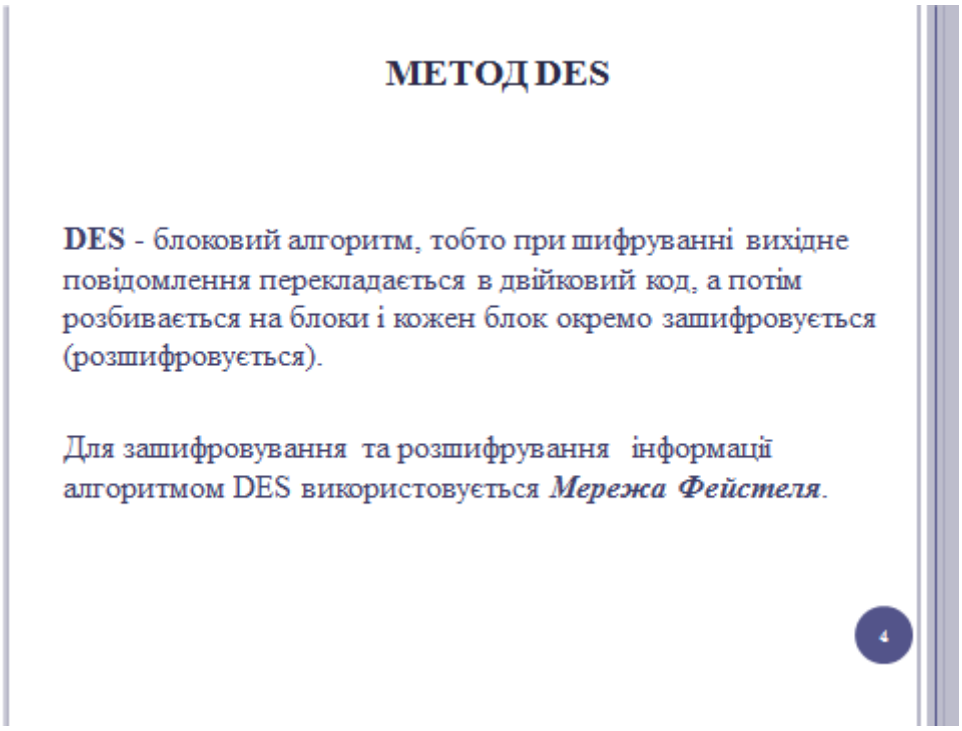
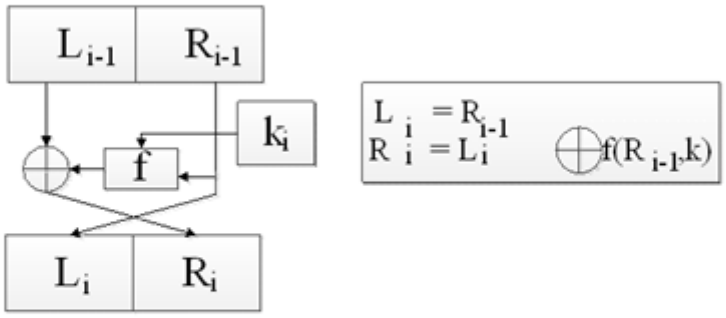


Рисунок А.4 - Слайд "Метод DES"

МЕТОД DES. СХЕМА



Пряме перетворення мережею Фейстеля

Джерело інформації
https://vscode.ru/prog-lessons/algoritm-des.html

5

Рисунок А.5 - Слайд "Метод DES. Схема"

МЕТОД LSB

LSB (Least Significant Bit) — цей метод полягає в заміні останніх значущих бітів у контейнері (зображення) на біти приховуваного повідомлення.

Різниця між порожнім і заповненим контейнерами повинна бути не відчутна для органів сприйняття людини.

6

Рисунок А.6 - Слайд "Метод LSB"

МЕТОД LSB. СХЕМА

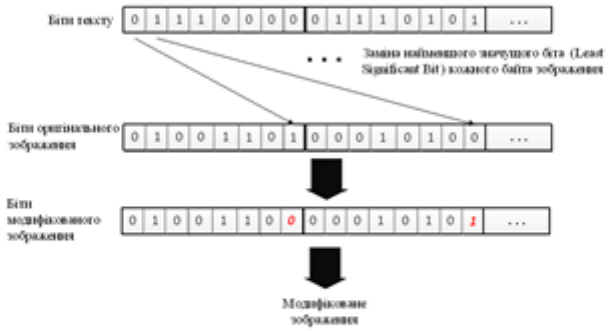


Схема алгоритму LSB методу

Джерело інформації:
https://habr.com/post/140373/

7

Рисунок А.7 - Слайд "Метод LSB. Схема"

ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ

- Шифрування текстової інформації методом DES;
- Приховування тексту в BMP зображення;
- Розшифровування текстової інформації та BMP файлів;
- Можливість збереження зашифрованої або розшифрованої інформації в окремий файл;

8

Рисунок А.8 - Слайд "Функціональні можливості"

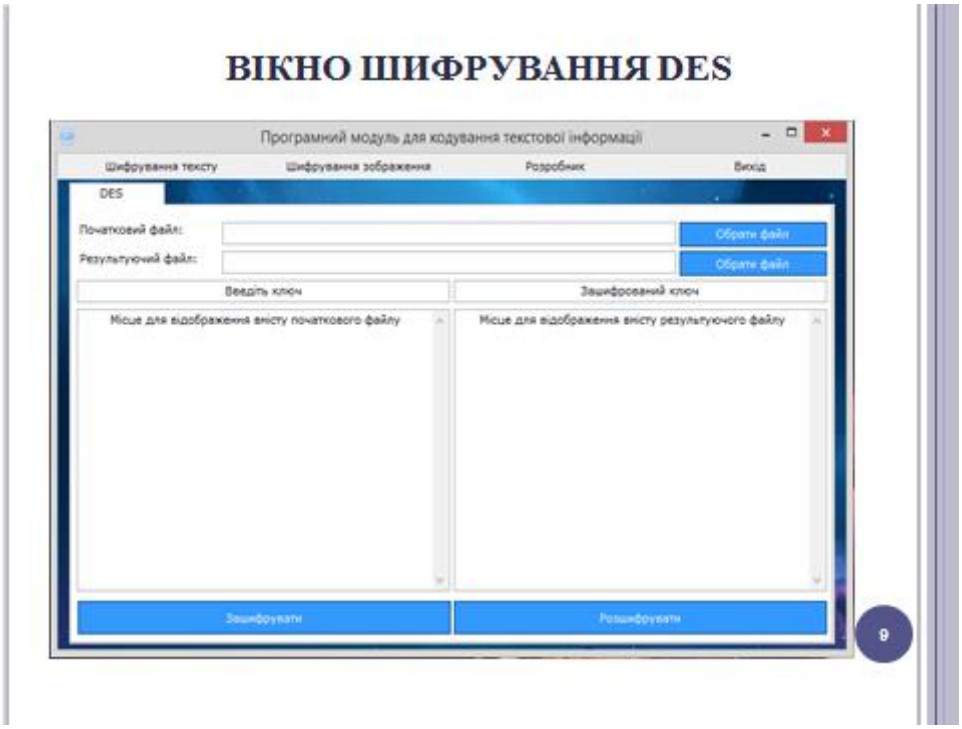


Рисунок А.9 - Слайд "Вікно шифрування DES"

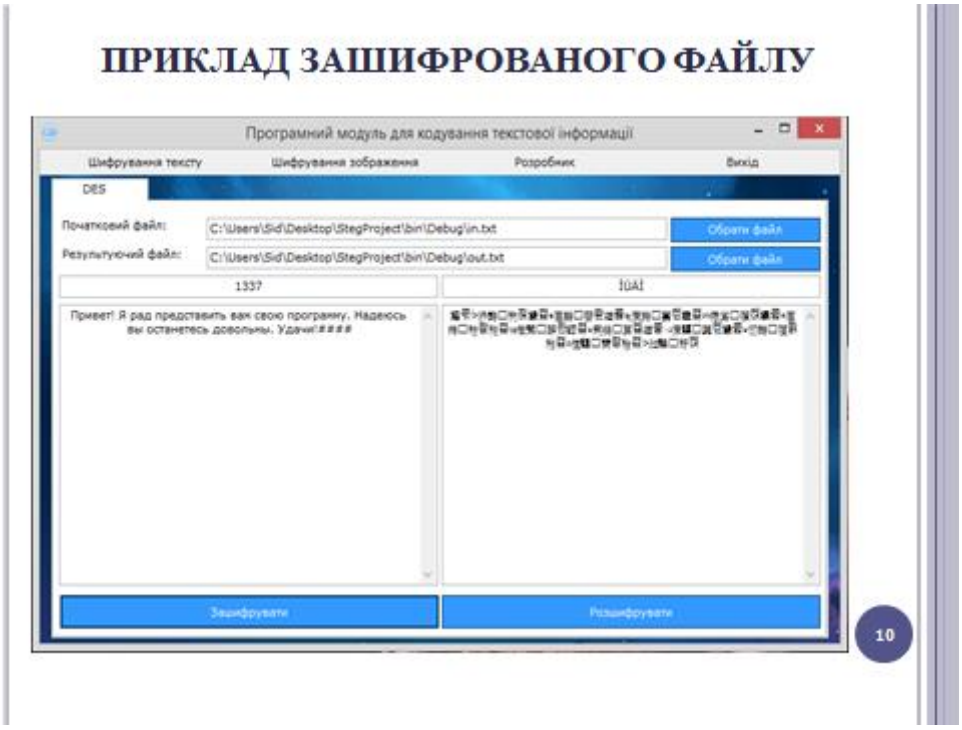


Рисунок А.10 - Слайд "Приклад зашифрованого файлу"

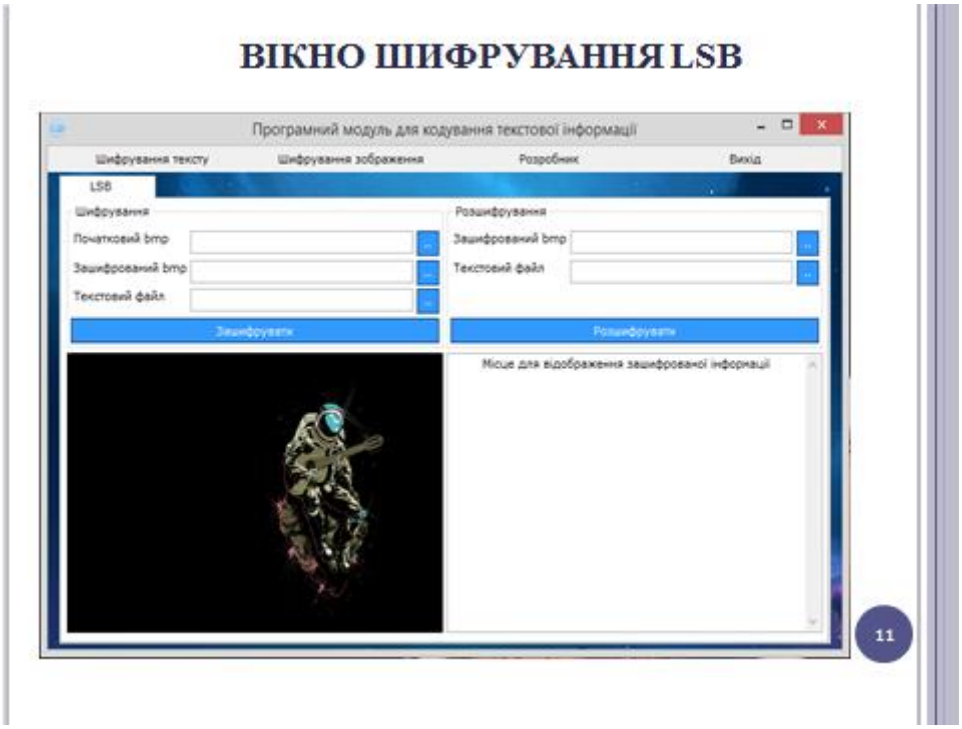


Рисунок А.11 - Слайд "Вікно шифрування LSB"

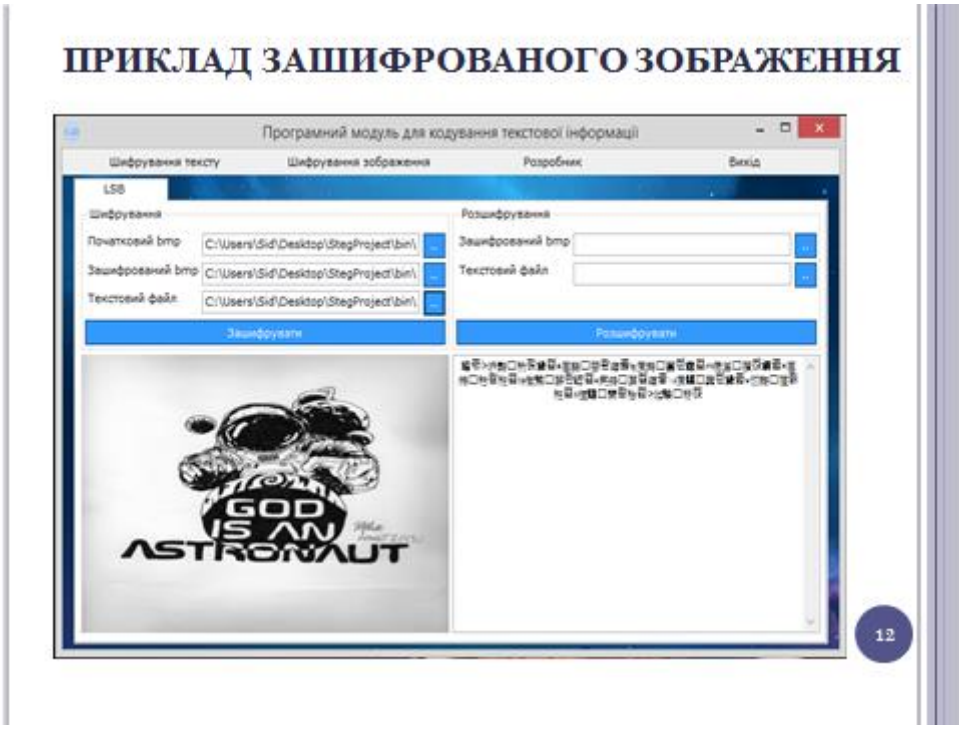


Рисунок А.12 - Слайд "Приклад зашифрованого зображення"

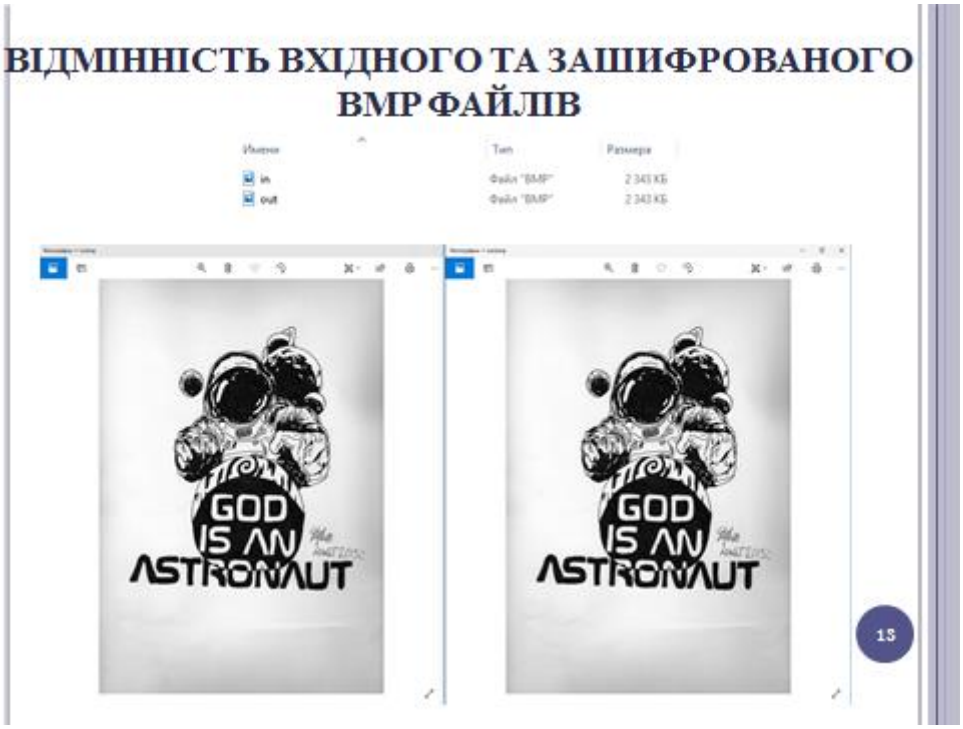


Рисунок А.13 - Слайд "Відмінність вхідного та зашифрованого BMP файлів"

Додаток Б.

Лістинг програмного модуля

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

namespace StegProject
{
    public partial class Form1 : Form
    {
        private const int sizeOfBlock = 128; //в DES размер блока 64 бит, но
поскольку в unicode символ в два раза длинее, то увеличим блок тоже в два раза
        private const int sizeOfChar = 16; //размер одного символа (in Unicode 16
bit)

        private const int shiftKey = 2; //сдвиг ключа

        private const int quantityOfRounds = 16; //количество раундов

        string[] Blocks; //сами блоки в двоичном формате

        public Form1()
        {
            InitializeComponent();
        }

        private BitArray ByteToBit(byte src) {
            BitArray bitArray = new BitArray(8);
            bool st = false;
            for (int i = 0; i < 8; i++)
            {
                if ((src >> i & 1) == 1) {
                    st = true;
                } else st = false;
                bitArray[i] = st;
            }
            return bitArray;
        }

        private byte BitToByte(BitArray scr) {
            byte num = 0;
            for (int i = 0; i < scr.Count; i++)
                if (scr[i] == true)
                    num += (byte)Math.Pow(2, i);
            return num;
        }

        /*Проверяет, зашифрован ли файл,  возвращает true, если символ в первом
пикслеле равен / иначе false */
        private bool isEncryption(Bitmap scr)
        {
```

```
            byte[] rez = new byte[1];
            Color color = scr.GetPixel(0, 0);
            BitArray colorArray = ByteToBit(color.R); //получаем байт цвета и
преобразуем в массив бит
            BitArray messageArray = ByteToBit(color.R); ;//инициализируем
результирующий массив бит
            messageArray[0] = colorArray[0];
            messageArray[1] = colorArray[1];

            colorArray = ByteToBit(color.G);//получаем байт цвета и преобразуем в
массив бит
            messageArray[2] = colorArray[0];
            messageArray[3] = colorArray[1];
            messageArray[4] = colorArray[2];

            colorArray = ByteToBit(color.B);//получаем байт цвета и преобразуем в
массив бит
            messageArray[5] = colorArray[0];
            messageArray[6] = colorArray[1];
            messageArray[7] = colorArray[2];
            rez[0] = BitToByte(messageArray); //получаем байт символа, записанного
в 1 пикселе
            string m = Encoding.GetEncoding(1251).GetString(rez);
            if (m == "/")
            {
                return true;
            }
            else return false;
        }

        /*Записыает количество символов для шифрования в первые биты картинки */
        private void WriteCountText(int count, Bitmap src) {
            byte[] CountSymbols =
Encoding.GetEncoding(1251).GetBytes(count.ToString());
            for (int i = 0; i < 3; i++)
            {
                BitArray bitCount = ByteToBit(CountSymbols[i]); //биты количества
символов
                Color pColor = src.GetPixel(0, i + 1); //1, 2, 3 пикселы
                BitArray bitsCurColor = ByteToBit(pColor.R); //бит цветов текущего
пикселя
                bitsCurColor[0] = bitCount[0];
                bitsCurColor[1] = bitCount[1];
                byte nR = BitToByte(bitsCurColor); //новый бит цвета пиксея

                bitsCurColor = ByteToBit(pColor.G);//бит бит цветов текущего
пикселя
                bitsCurColor[0] = bitCount[2];
                bitsCurColor[1] = bitCount[3];
                bitsCurColor[2] = bitCount[4];
                byte nG = BitToByte(bitsCurColor);//новый цвет пиксея

                bitsCurColor = ByteToBit(pColor.B);//бит бит цветов текущего
пикселя
                bitsCurColor[0] = bitCount[5];
                bitsCurColor[1] = bitCount[6];
                bitsCurColor[2] = bitCount[7];
                byte nB = BitToByte(bitsCurColor);//новый цвет пиксея

                Color nColor = Color.FromArgb(nR, nG, nB); //новый цвет из
полученных битов
                src.SetPixel(0, i + 1, nColor); //записали полученный цвет в
картинку
```

```
            }
        }

        /*Читает количество символов для дешифрования из первых бит картинки*/
        private int ReadCountText(Bitmap src) {
            byte[] rez = new byte[3]; //массив на 3 элемента, т.е. максимум 999
символов шифруется
            for (int i = 0; i < 3; i++)
            {
                Color color = src.GetPixel(0, i + 1); //цвет 1, 2, 3 пикселей
                BitArray colorArray = ByteToBit(color.R); //биты цвета
                BitArray bitCount = ByteToBit(color.R); ; //инициализация
результирующего массива бит
                bitCount[0] = colorArray[0];
                bitCount[1] = colorArray[1];

                colorArray = ByteToBit(color.G);
                bitCount[2] = colorArray[0];
                bitCount[3] = colorArray[1];
                bitCount[4] = colorArray[2];

                colorArray = ByteToBit(color.B);
                bitCount[5] = colorArray[0];
                bitCount[6] = colorArray[1];
                bitCount[7] = colorArray[2];
                rez[i] = BitToByte(bitCount);
            }
            string m = Encoding.GetEncoding(1251).GetString(rez);
            return Convert.ToInt32(m, 10);
        }

        /* Открыть файл для шифрования */
        private void button1_Click(object sender, EventArgs e)
        {
            string FilePic = textBox5.Text;
            if (FilePic == "")
                return;

            string FileText = textBox7.Text ;
            if (FileText == "")
                return;

            FileStream rFile;
            try
            {
                rFile = new FileStream(FilePic, FileMode.Open); //открываем поток
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            Bitmap bPic = new Bitmap(rFile);
         //   pictureBox1.Image = bPic;


            FileStream rText;
            try
            {
                rText = new FileStream(FileText, FileMode.Open); //открываем поток
            }
```

```
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            BinaryReader bText = new BinaryReader(rText, Encoding.ASCII);

            List<byte> bList = new List<byte>();
            while (bText.PeekChar() != -1) { //считали весь текстовый файл для
шифрования в лист байт
                bList.Add(bText.ReadByte());
            }
            int CountText = bList.Count; // в CountText - количество в байтах
текста, который нужно закодировать
            bText.Close();
            rFile.Close();

            //проверяем, поместиться ли исходный текст в картинке
            if (CountText > ((bPic.Width * bPic.Height)) - 4 ) {
                MessageBox.Show("Выбранная картинка мала для размещения выбранного
текста", "Информация", MessageBoxButtons.OK);
                return;
            }

            //проверяем, может быть картинка уже зашифрована
            if (isEncryption(bPic))
            {
                MessageBox.Show("Файл уже зашифрован", "Информация",
MessageBoxButtons.OK);
                return;
            }

            byte [] Symbol = Encoding.GetEncoding(1251).GetBytes("/");
            BitArray ArrBeginSymbol = ByteToBit(Symbol[0]);
            Color curColor = bPic.GetPixel(0, 0);
            BitArray tempArray = ByteToBit(curColor.R);
            tempArray[0] = ArrBeginSymbol[0];
            tempArray[1] = ArrBeginSymbol[1];
            byte nR = BitToByte(tempArray);

            tempArray = ByteToBit(curColor.G);
            tempArray[0] = ArrBeginSymbol[2];
            tempArray[1] = ArrBeginSymbol[3];
            tempArray[2] = ArrBeginSymbol[4];
            byte nG = BitToByte(tempArray);

            tempArray = ByteToBit(curColor.B);
            tempArray[0] = ArrBeginSymbol[5];
            tempArray[1] = ArrBeginSymbol[6];
            tempArray[2] = ArrBeginSymbol[7];
            byte nB = BitToByte(tempArray);

            Color nColor = Color.FromArgb(nR, nG, nB);
            bPic.SetPixel(0, 0, nColor);
            //то есть в первом пикселе будет символ /, который говорит о том, что
картика зашифрована

            WriteCountText(CountText, bPic); //записываем количество символов для
шифрования

            int index = 0;
            bool st = false;
```

```
            for (int i = 4; i < bPic.Width; i++) {
                for (int j = 0; j < bPic.Height; j++) {
                    Color pixelColor = bPic.GetPixel(i, j);
                    if (index == bList.Count) {
                        st = true;
                        break;
                    }
                    BitArray colorArray = ByteToBit(pixelColor.R);
                    BitArray messageArray = ByteToBit(bList[index]);
                    colorArray[0] = messageArray[0]; //меняем
                    colorArray[1] = messageArray[1]; // в нашем цвете биты
                    byte newR = BitToByte(colorArray);

                    colorArray = ByteToBit(pixelColor.G);
                    colorArray[0] = messageArray[2];
                    colorArray[1] = messageArray[3];
                    colorArray[2] = messageArray[4];
                    byte newG = BitToByte(colorArray);

                    colorArray = ByteToBit(pixelColor.B);
                    colorArray[0] = messageArray[5];
                    colorArray[1] = messageArray[6];
                    colorArray[2] = messageArray[7];
                    byte newB = BitToByte(colorArray);

                    Color newColor = Color.FromArgb(newR, newG, newB);
                    bPic.SetPixel(i, j, newColor);
                    index ++;
                }
                if (st) {
                    break;
                }
            }
           // pictureBox1.Image = bPic;

            string sFilePic = textBox6.Text;
            if (sFilePic == "")
                return;

            FileStream wFile;
            try
            {
                wFile = new FileStream(sFilePic, FileMode.Create); //открываем
поток на запись результатов
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла на запись", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            bPic.Save(wFile, System.Drawing.Imaging.ImageFormat.Bmp);
            wFile.Close(); //закрываем поток

            MessageBox.Show("Шифрование прошло успешно!\nРезультат в файле:\n" +
sFilePic, "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        /*Открыть файл для дешифрования */
        private void button2_Click(object sender, EventArgs e)
        {
            string FilePic = textBox10.Text;
```

```
            if (FilePic == "")
                return;

            FileStream rFile;
            try
            {
                rFile = new FileStream(FilePic, FileMode.Open); //открываем поток
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            Bitmap bPic = new Bitmap(rFile);
            if (!isEncryption(bPic)) {
                MessageBox.Show("В файле нет зашифрованной информации",
"Информация", MessageBoxButtons.OK);
                return;
            }

            int countSymbol = ReadCountText(bPic); //считали количество
зашифрованных символов
            byte[] message = new byte[countSymbol];
            int index = 0;
            bool st = false;
            for (int i = 4; i < bPic.Width; i++) {
                for (int j = 0; j < bPic.Height; j++) {
                    Color pixelColor = bPic.GetPixel(i, j);
                    if (index == message.Length) {
                        st = true;
                        break;
                    }
                    BitArray colorArray = ByteToBit(pixelColor.R);
                    BitArray messageArray = ByteToBit(pixelColor.R); ;
                    messageArray[0] = colorArray[0];
                    messageArray[1] = colorArray[1];

                    colorArray = ByteToBit(pixelColor.G);
                    messageArray[2] = colorArray[0];
                    messageArray[3] = colorArray[1];
                    messageArray[4] = colorArray[2];

                    colorArray = ByteToBit(pixelColor.B);
                    messageArray[5] = colorArray[0];
                    messageArray[6] = colorArray[1];
                    messageArray[7] = colorArray[2];
                    message[index] = BitToByte(messageArray);
                    index++;
                }
                if (st) {
                    break;
                }
            }
            string strMessage = Encoding.GetEncoding(1251).GetString(message);

            string sFileText = textBox9.Text;
            if (sFileText == "")
                return;

            FileStream wFile;
            try
            {
```

```
                wFile = new FileStream(sFileText, FileMode.Create); //открываем
поток на запись результатов
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла на запись", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            StreamWriter wText = new StreamWriter(wFile, Encoding.Default);
            wText.Write(strMessage);

            wText.Close();
            wFile.Close(); //закрываем поток

            string s = "";
            StreamReader sr = new StreamReader(sFileText);
            textBox8.Clear();

            while (!sr.EndOfStream)
            {
                string str = sr.ReadLine();
                s += str;
                textBox8.Text += str + Environment.NewLine;

            }

            sr.Close();

            MessageBox.Show("Расшифровка прошла успешно!\nТекст записан в файл:\n"
+ sFileText, "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        //зашифровать
        private void buttonEncrypt_Click(object sender, EventArgs e)
        {
            textBox1.Clear();
            textBox2.Clear();



            if (textBoxEncodeKeyWord.Text.Length > 0)
            {
                string FileText1;
                FileText1 = textBox3.Text;

                if (FileText1 == "")
                    return;

                string s = "";

                string key = textBoxEncodeKeyWord.Text;

                StreamReader sr = new StreamReader(FileText1);

                while (!sr.EndOfStream)
                {
                    string str = sr.ReadLine();
                    s += str;
                    textBox1.Text += str+Environment.NewLine;

                }
```

```
                sr.Close();



                s = StringToRightLength(s);

                CutStringIntoBlocks(s);

                key = CorrectKeyWord(key, s.Length / (2 * Blocks.Length));
                textBoxEncodeKeyWord.Text = key;
                key = StringToBinaryFormat(key);

                for (int j = 0; j < quantityOfRounds; j++)
                {
                    for (int i = 0; i < Blocks.Length; i++)
                        Blocks[i] = EncodeDES_One_Round(Blocks[i], key);

                    key = KeyToNextRound(key);
                }

                key = KeyToPrevRound(key);

                textBoxDecodeKeyWord.Text = StringFromBinaryToNormalFormat(key);

                string result = "";

                for (int i = 0; i < Blocks.Length; i++)
                    result += Blocks[i];

                string FileText2;
                FileText2 = textBox4.Text;

                if (FileText2 == "")
                    return;

                StreamWriter sw = new StreamWriter(FileText2);
                sw.WriteLine(StringFromBinaryToNormalFormat(result));
                sw.Close();

                textBox2.Text = StringFromBinaryToNormalFormat(result);
              //  Process.Start("out1.txt");
                MessageBox.Show("Шифрование прошло успешно!\nРезультаты в
файле:\n" + FileText2, "Информация", MessageBoxButtons.OK,
MessageBoxIcon.Information);

            }
            else
                MessageBox.Show("Введите ключевое слово!");
        }
        //расшифровать
        private void buttonDecipher_Click(object sender, EventArgs e)
        {
            textBox1.Clear();
            textBox2.Clear();
            if (textBoxDecodeKeyWord.Text.Length > 0)
            {
                string FileText1;
                FileText1 = textBox3.Text;

                if (FileText1 == "")
                    return;

                string s = "";
```

```
                string key = StringToBinaryFormat(textBoxDecodeKeyWord.Text);

                StreamReader sr = new StreamReader(FileText1);

                while (!sr.EndOfStream)
                {
                    s += sr.ReadLine();
                }
                textBox1.Text = s;
                sr.Close();

                s = StringToBinaryFormat(s);

                CutBinaryStringIntoBlocks(s);

                for (int j = 0; j < quantityOfRounds; j++)
                {
                    for (int i = 0; i < Blocks.Length; i++)
                        Blocks[i] = DecodeDES_One_Round(Blocks[i], key);

                    key = KeyToPrevRound(key);
                }

                key = KeyToNextRound(key);

                textBoxEncodeKeyWord.Text = StringFromBinaryToNormalFormat(key);

                string result = "";

                for (int i = 0; i < Blocks.Length; i++)
                    result += Blocks[i];

                string FileText2;
                FileText2 = textBox4.Text;

                if (FileText2 == "")
                    return;

                StreamWriter sw = new StreamWriter(FileText2);
                sw.WriteLine(StringFromBinaryToNormalFormat(result));
                sw.Close();

                textBox2.Text = StringFromBinaryToNormalFormat(result);
                MessageBox.Show("Расшифровка прошло успешно!\nРезультаты в
файле:\n" + FileText2, "Информация", MessageBoxButtons.OK,
MessageBoxIcon.Information);
               // Process.Start("out2.txt");
            }
            else
                MessageBox.Show("Введите ключевое слово!");

        }

        //доводим строку до размера, чтобы делилась на sizeOfBlock
        private string StringToRightLength(string input)
        {
            while (((input.Length * sizeOfChar) % sizeOfBlock) != 0)
                input += "#";

            return input;
        }
```

```
//разбиение обычной строки на блоки
private void CutStringIntoBlocks(string input)
{
    Blocks = new string[(input.Length * sizeOfChar) / sizeOfBlock];

    int lengthOfBlock = input.Length / Blocks.Length;

    for (int i = 0; i < Blocks.Length; i++)
    {
        Blocks[i] = input.Substring(i * lengthOfBlock, lengthOfBlock);
        Blocks[i] = StringToBinaryFormat(Blocks[i]);
    }
}

//разбиение двоичной строки на блоки
private void CutBinaryStringIntoBlocks(string input)
{
    Blocks = new string[input.Length / sizeOfBlock];

    int lengthOfBlock = input.Length / Blocks.Length;

    for (int i = 0; i < Blocks.Length; i++)
        Blocks[i] = input.Substring(i * lengthOfBlock, lengthOfBlock);
}

//перевод строки в двоичный формат
private string StringToBinaryFormat(string input)
{
    string output = "";

    for (int i = 0; i < input.Length; i++)
    {
        string char_binary = Convert.ToString(input[i], 2);

        while (char_binary.Length < sizeOfChar)
            char_binary = "0" + char_binary;

        output += char_binary;
    }

    return output;
}

//доводим длину ключа до нужной
private string CorrectKeyWord(string input, int lengthKey)
{
    if (input.Length > lengthKey)
        input = input.Substring(0, lengthKey);
    else
        while (input.Length < lengthKey)
            input = "0" + input;

    return input;
}

//шифрование DES один раунд
private string EncodeDES_One_Round(string input, string key)
{
    string L = input.Substring(0, input.Length / 2);
    string R = input.Substring(input.Length / 2, input.Length / 2);

    return (R + XOR(L, f(R, key)));
}
```

```
        //расшифровка DES один раунд
        private string DecodeDES_One_Round(string input, string key)
        {
            string L = input.Substring(0, input.Length / 2);
            string R = input.Substring(input.Length / 2, input.Length / 2);

            return (XOR(f(L, key), R) + L);
        }

        //XOR двух строк с двоичными данными
        private string XOR(string s1, string s2)
        {
            string result = "";

            for (int i = 0; i < s1.Length; i++)
            {
                bool a = Convert.ToBoolean(Convert.ToInt32(s1[i].ToString()));
                bool b = Convert.ToBoolean(Convert.ToInt32(s2[i].ToString()));

                if (a ^ b)
                    result += "1";
                else
                    result += "0";
            }
            return result;
        }

        //шифрующая функция f. в данном случае это XOR
        private string f(string s1, string s2)
        {
            return XOR(s1, s2);
        }

        //вычисление ключа для следующего раунда шифрования. циклический сдвиг >>
2
        private string KeyToNextRound(string key)
        {
            for (int i = 0; i < shiftKey; i++)
            {
                key = key[key.Length - 1] + key;
                key = key.Remove(key.Length - 1);
            }

            return key;
        }

        //вычисление ключа для следующего раунда расшифровки. циклический сдвиг <<
2
        private string KeyToPrevRound(string key)
        {
            for (int i = 0; i < shiftKey; i++)
            {
                key = key + key[0];
                key = key.Remove(0, 1);
            }

            return key;
        }

        //переводим строку с двоичными данными в символьный формат
        private string StringFromBinaryToNormalFormat(string input)
        {
```

```
            string output = "";

            while (input.Length > 0)
            {
                string char_binary = input.Substring(0, sizeOfChar);
                input = input.Remove(0, sizeOfChar);

                int a = 0;
                int degree = char_binary.Length - 1;

                foreach (char c in char_binary)
                    a += Convert.ToInt32(c.ToString()) * (int)Math.Pow(2, degree--
);

                output += ((char)a).ToString();
            }

            return output;
        }

        private void button3_Click(object sender, EventArgs e)
        {
            string FileText;
            textBox3.Clear();

            OpenFileDialog dText = new OpenFileDialog();
            dText.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*";
            if (dText.ShowDialog() == DialogResult.OK)
            {
                FileText = dText.FileName;
            }
            else
            {
                FileText = "";
                return;
            }

            textBox3.Text = FileText;
        }

        private void button4_Click(object sender, EventArgs e)
        {
            string FileText;
            textBox4.Clear();

            SaveFileDialog dSaveText = new SaveFileDialog();
            dSaveText.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы
(*.*)|*.*";
            if (dSaveText.ShowDialog() == DialogResult.OK)
            {
                FileText = dSaveText.FileName;
            }
            else
            {
                FileText = "";
                return;
            };

            textBox4.Text = FileText;

        }

        private void button5_Click(object sender, EventArgs e)
```

```
        {
            string FilePic;
            textBox5.Clear();
            OpenFileDialog dPic = new OpenFileDialog();
            dPic.Filter = "Файлы изображений (*.bmp)|*.bmp|Все файлы (*.*)|*.*";
            if (dPic.ShowDialog() == DialogResult.OK)
            {
                FilePic = dPic.FileName;
            }
            else
            {
                FilePic = "";
                return;
            }

            FileStream rFile;
            try
            {
                rFile = new FileStream(FilePic, FileMode.Open); //открываем поток
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            Bitmap bPic = new Bitmap(rFile);
            pictureBox1.Image = bPic;

            textBox5.Text = FilePic;

            rFile.Close();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            string FileText;
            textBox7.Clear();
            textBox8.Clear();
            OpenFileDialog dText = new OpenFileDialog();
            dText.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*";
            if (dText.ShowDialog() == DialogResult.OK)
            {
                FileText = dText.FileName;
            }
            else
            {
                FileText = "";
                return;
            }
            textBox7.Text = FileText;

            string s = "";
            StreamReader sr = new StreamReader(FileText);

            while (!sr.EndOfStream)
            {
                string str = sr.ReadLine();
                s += str;
                textBox8.Text += str + Environment.NewLine;

            }
```

```
            sr.Close();

        }

        private void button6_Click(object sender, EventArgs e)
        {
            string FilePic;
            textBox6.Clear();
            SaveFileDialog dPic = new SaveFileDialog();
            dPic.Filter = "Файлы изображений (*.bmp)|*.bmp|Все файлы (*.*)|*.*";
            if (dPic.ShowDialog() == DialogResult.OK)
            {
                FilePic = dPic.FileName;
            }
            else
            {
                FilePic = "";
                return;
            }

            textBox6.Text = FilePic;

        }

        private void button9_Click(object sender, EventArgs e)
        {
            string FilePic;
            textBox10.Clear();
            OpenFileDialog dPic = new OpenFileDialog();
            dPic.Filter = "Файлы изображений (*.bmp)|*.bmp|Все файлы (*.*)|*.*";
            if (dPic.ShowDialog() == DialogResult.OK)
            {
                FilePic = dPic.FileName;
            }
            else
            {
                FilePic = "";
                return;
            }

            FileStream rFile;
            try
            {
                rFile = new FileStream(FilePic, FileMode.Open); //открываем поток
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            Bitmap bPic = new Bitmap(rFile);
            pictureBox1.Image = bPic;

            textBox10.Text = FilePic;

            rFile.Close();
        }

        private void button8_Click(object sender, EventArgs e)
        {
            string sFileText;
            textBox9.Clear();
```

```
            SaveFileDialog dSaveText = new SaveFileDialog();
            dSaveText.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы
(*.*)|*.*";
            if (dSaveText.ShowDialog() == DialogResult.OK)
            {
                sFileText = dSaveText.FileName;
            }
            else
            {
                sFileText = "";
                return;
            };
            textBox9.Text = sFileText;
        }

        bool textboss = true;

        private void textBoxEncodeKeyWord_Enter(object sender, EventArgs e)
        {
            if (textboss == true)
            {

                textBoxEncodeKeyWord.Text = "";
                textboss = false;

            }
        }

        private void textBoxEncodeKeyWord_Leave(object sender, EventArgs e)
        {
            if (textBoxEncodeKeyWord.Text == "")
            {
                textBoxEncodeKeyWord.Text = "Введите ключевое слово";
                textboss = true;
            }
        }



        bool textbos = true;
        private void textBoxDecodeKeyWord_Enter(object sender, EventArgs e)
        {
            if (textbos == true)
            {

                textBoxDecodeKeyWord.Text = "";
                textbos = false;

            }
        }

        private void textBoxDecodeKeyWord_Leave(object sender, EventArgs e)
        {
            if (textBoxDecodeKeyWord.Text == "")
            {
                textBoxDecodeKeyWord.Text = "Скопируйте код из этого поля";
                textbos = true;
            }
        }

        private void textBoxDecodeKeyWord_TextChanged(object sender, EventArgs e)
        {
            textbos = false;
```

```
        }

        private void шифрованиеDesToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            pictureBox2.Hide();
            tabControl1.Show();
            tabPage2.Parent = null;
            tabPage1.Parent =tabControl1;
            button10.Hide();
        }


        private void шифрованиеLBSToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            pictureBox2.Hide();
            tabControl1.Show();
            tabPage1.Parent = null;
            tabPage2.Parent = tabControl1;
            button10.Hide();
        }


        private void выходToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            tabPage1.Parent = null;
            tabPage2.Parent = null;

        }

        private void оПрограммеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Form2 form = new Form2();
            form.ShowDialog();
        }

        bool dexbox = true;

        private void textBox1_Enter(object sender, EventArgs e)
        {
            if (dexbox == true)
            {

                textBox1.Text = "";
                dexbox = false;

            }
        }

        private void textBox1_Leave(object sender, EventArgs e)
        {
            if (textBox1.Text == "")
            {
                textBox1.Text = "Здесь будет отображаться содержание исходного
файла.";
                dexbox = true;
            }
```

```
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            dexbox = false;
        }

        bool dexbox2 = true;

        private void textBox2_Enter(object sender, EventArgs e)
        {
            if (dexbox2 == true)
            {

                textBox2.Text = "";
                dexbox2 = false;

            }
        }

        private void textBox2_Leave(object sender, EventArgs e)
        {
            if (textBox2.Text == "")
            {
                textBox2.Text = "Здесь будет отображаться содержание
результирующего файла.";
                dexbox2 = true;
            }
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
            dexbox2 = false;
        }

        private void button10_Click(object sender, EventArgs e)
        {
            pictureBox2.Hide();
            tabControl1.Show();
            tabPage2.Parent = null;
            tabPage1.Parent = tabControl1;
            button10.Hide();
        }


    }
}
```