Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Тулуши Максима Олексійовича*
(ПІБ)

академічної групи *121-18ск-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення для шифрування та дешифрування текстових повідомлень*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Корнієнко В.І.*			
розділів:				
спеціальний	*проф. Корнієнко В.І.*			
економічний	*проф. Вагонова О.Г.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2021

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-18ск-1* *Тулуши Максима Олексійовича*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення для шифрування та дешифрування текстових повідомлень*

затверджена наказом ректора НТУ «ДП» від « 07 » 06 2021 р. №317 -с

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2021 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2021 р.***

Завдання видав _____ *проф. Корнієнко В.І.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Тулуша М.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2021 р.*

Термін подання кваліфікаційної роботи до ЕК: *11.06.2021 р.*

РЕФЕРАТ

Пояснювальна записка 78 с., 38 рис., 3 дод., 31 джерело.

Об'єкт розробки: додаток для шифрування та дешифрування текстових повідомлень.

Мета кваліфікаційної роботи: підвищення зручності та ефективності шифрування та дешифрування текстових повідомлень.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування застосунку, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження застосунку, описана робота програми.

В економічному розділі визначено трудомісткість розробленого програмного продукту, проведений підрахунок вартості роботи по створенню застосунку та розраховано час на його створення.

Практичне значення полягає у підвищенні надійності передачі або зберігання інформації, для цього було обрано створення додатку для шифрування-розшифрування текстових повідомлень. Також даний додаток може використовувати викладач чи студент для підвищення продуктивності перевірки завдань з криптографії.

Актуальність програмного продукту визначається великим попитом на подібні інструменти.

Список ключових слів: КРИПТОГРАФІЯ, ШИФР ЦЕЗАРЯ, ШИФР ВІЖЕНЕРА ТА ГАМУВАННЯ, ШИФР ПЕРЕСТАНОВКИ, ХЕШУВАННЯ MD5, СТВОРЕННЯ QR-КОДУ, ЦИФРОВИЙ ПІДПИС, DES-ШИФРУВАННЯ, RSA-ШИФРУВАННЯ.

ABSTRACT

Explanatory note 78 pp., 38 figs., 3 apps., 31 sources.

Development object: application for encryption and decryption of text messages.

The purpose of the qualification work: to increase the convenience and efficiency of encryption and decryption of text messages.

The introduction considers the analysis and the current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and purpose of development is defined, the statement of the task is developed, requirements to software realization, technologies and software are set.

The second section analyzes the existing solutions, selects the platform for development, performs design and development of the program, describes the algorithm and structure of the application, defines the input and output data, describes the characteristics of the parameters of hardware, describes the call and download the application, describes the program .

In the economic section, the complexity of the developed software product is determined, the cost of work on creating the application is calculated and the time for its creation is calculated.

The practical value is to increase the reliability of transmission or storage of information was chosen to create an application for encryption-decryption of text messages. This application can also be used by a teacher or student to improve the performance of cryptography tasks.

The relevance of the software product is determined by the high demand for such tools.

Keywords: CRYPTOGRAPHY, CESAR CODE, WIENER'S CODE AND HAMING, CHANGE CODE, MD5 HASHING, QR-CREATION CREATION, DIGITAL SUPPORTS.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ІС – інформаційна система;

ООП – об'єктно-орієнтоване програмування;

ОС – операційна система;

ПЗ – програмне забезпечення;

ІТ – інформаційні технології.

ВСТУП

Проблема захисту інформаційних ресурсів набуває все більш важливого значення, хоча вона є однією із найскладніших задач. В першу чергу, це пояснюється прискореними темпами науково-технічного прогресу, результатом якого є нові технічні та електронні засоби, які становлять небезпеку виникнення каналів витоку інформації. Також одним із факторів, які визначають трудомісткість вирішення завдань захисту інформації, є розширення кола користувачів, що мають доступ до ресурсів комп'ютерної системи і масивів даних, які знаходяться в ній. Для вирішення цієї проблеми потрібна система заходів, головною ціллю якої є попередження від несанкціонованого доступу, наслідком якого може бути втрата, модифікація і витік інформації.

Швидкий розвиток інформаційних технологій привів до нових досягнень в сфері безпеки інформації, яка є дуже важливою для сучасного суспільства. Питання розроблення та впровадження методів захисту інформації є актуальними не лише для криптографії та стеганографії, а й для майже всіх галузей науки, враховуючи високу автоматизацію різних сфер людської діяльності перетворившись на загальнопоширений інструмент передачі та захисту даних,

Сучасна криптографія базується на математичному апараті, що включає теорію ймовірності та абстрактну алгебру. Основним завданням математики в криптографії є забезпечення криптографічної стійкості, тобто здатності протистояти практичному злому. Криптографічна стійкість визначається кількістю затраченого часу і ресурсів, щоб із шифрованого тексту відновити вихідний відкритий текст. Результатом стійкої криптографії є шифротекст, що винятково складно зламати без володіння визначеними інструментами по дешифруванню [3].

Крім того, захист інформації необхідний для забезпечення працездатності самих автоматизованих систем. Не випадково необхідність впровадження

засобів криптографічного захисту інформації є одним з положень Закону України «Про електронні документи і електронний документообіг».

Слід зазначити, що стандарти, які на сьогодні діють в Україні, в області криптографічного захисту інформації не покривають весь спектр необхідних криптоалгоритмів і криптопротоколів, наприклад, відсутні стандарти на асиметричну криптосистему і систему узгодження симетричних ключів. З іншого боку, необхідні криптоалгоритми і криптопротоколи можна побудувати на основі окремих положень вітчизняних і міжнародних стандартів, з подальшою процедурою сертифікації. Це зумовлює до необхідність аналізу вимог і обмежень, встановлених нормативно-правовими документами і вимагає об'єктивної оцінки можливостей організації відносно реалізації відповідних технологій [2].

Щоб підвищити надійність передачі або зберігання інформації було обрано створення додатку для шифрування-дешифрування текстових повідомлень.

Також даний додаток може використовувати викладач чи студент для підвищення продуктивності перевірки завдань з криптографії.

Завдання кваліфікаційної роботи та об'єкт її діяльності безпосередньо пов'язані з освітньою програмою спеціальності 121 «Інженерія програмного забезпечення» та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених компетенцій.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

1.1.1. Шифрування з використанням публічного ключа

Найчастіше використовується реалізації шифрування з використанням публічного ключа, що базуються на алгоритмах, запатентованих RSA Data Security. Тому цей розділ описує підхід RSA до шифрування з використанням публічного ключа.

Шифрування з використанням публічного ключа (також називається асиметричне шифрування) залучає пару ключів: публічний ключ та приватний ключ, що пов'язані з об'єктом, що потрібно аутентифікувати електронно, підписати чи зашифрувати. Кожен публічний ключ є у вільному доступі, а відповідний приватний ключ зберігається в таємниці. Дані зашифровані вашим публічним ключем можуть бути дешифровані лише вашим приватним ключем. Рисунок 1.1. ілюструє спрощений вигляд того, як працює шифрування з використанням публічного ключа.

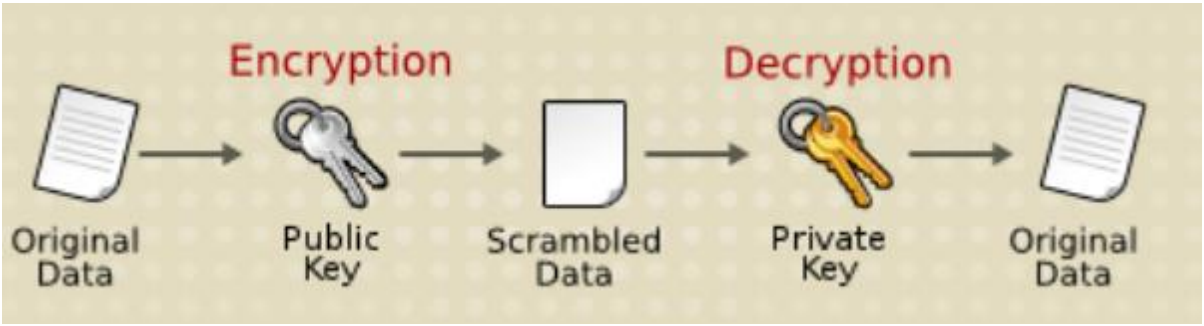

Рис. 1.1. Схема передачі зашифрованої інформації

Схема, що зображена на рис.1.1. дозволяє вільно розповсюджувати публічний ключ і лише ви будете спроможні читати інформацію зашифровану цим ключем. В загальному, щоб відправити зашифровані дані комусь, ви

шифруєте інформацію цим публічним ключем і особа, що отримує зашифровану інформацію дешифрує її відповідним приватним ключем.

В порівнянні з шифруванням симетричним ключем, шифрування з використанням публічного ключа потребує більший обсяг обчислень і тому не завжди використовується для великих обсягів інформації. Однак, є можливість використовувати шифрування з використанням публічного ключа для відправки симетричного ключа, що потім може бути використаний для шифрування додаткових даних. Це підхід, що використовується протоколом SSL.

Зворотній порядок схеми, показаної на рис.1.1. також працює: дані зашифровані вашим приватним ключем можуть бути дешифровані лише вашим публічним ключем. Це не є бажаний шлях шифрування секретної інформації, тому що будь-хто з вашим публічним ключем, який за визначенням є опублікованим, має змогу дешифрувати дані. Тим не менш, шифрування з використанням приватного ключа є корисним, адже це означає, що ви можете використовувати ваш приватний ключ щоб підписати дані вашим цифровим підписом, що є важливою вимогою для електронної комерції та інших комерційних сторін використання криптографії. Програмне забезпечення для клієнта, наприклад Firefox може пізніше використати ваш публічний ключ для підтвердження того, що повідомлення було підписане вашим приватним ключем і не було підроблене з моменту підписання.

1.1.2. Довжина ключа й захищеність шифрування

Процес злому алгоритму шифрування це, по суті, пошук ключа для доступу до зашифрованих даних в звичайному тексті. Для симетричних алгоритмів, злом зазвичай означає спробу визначити ключ, що використовувався при шифруванні тексту. Для алгоритмів з використанням публічного ключа, процес злому алгоритму означає отримання секретної інформації, що була поширена між двома абонентами.

Один з методів злому симетричного алгоритму шифрування є просто спроба підбору кожного відомого ключа у рамках цілого алгоритму, допоки не знайдеться вірний. Для алгоритмів з використанням публічного ключа, так як одна половина ключів є в доступі завчасно, інша половина (приватний ключ) може бути розрахована використовуючи публічний, через комплекс складних математичних розрахунків. Ручний пошук ключа для злому алгоритму називається методом грубої сили (brute force attack).

Злом алгоритму знайомить з ризиком перехоплення чи навіть розкриття особистості та підробки підтвердження. Також повідомляє нас про ризик перехоплення та розкриття приватної інформації.

Міцність ключа алгоритму вираховується шляхом знаходження найшвидшого методу для злому цього ж алгоритму і порівняння цього способу з методом грубої сили.

Для симетричних ключів, міцність шифрування часто описується в вигляді розміру чи довжини ключів, що використовуються для виконання шифрування: загалом, довші ключі надають кращий рівень захищеності. Довжина ключа вимірюється у бітах. Наприклад, 128-бітні ключі, що використовуються для шифру з симетричними ключами, RC4, що підтримується протоколом SSL надають значно кращий криптографічний захист ніж 40-бітні ключі, що використовуються тим самим шифром. Грубо кажучи, 128-бітне RC4 шифрування є в $3x10^{26}$ разів захищеним, ніж 40-бітне RC4 шифрування. Ключ шифрування вважається повністю захищеним, якщо найкращий відомий метод атаки з метою злому не є швидшим ніж спроба атаки методом грубої сили, для тестування кожного можливого ключа.

Різні шифри можуть потребувати ключі різноманітної довжни для отримання одного рівня захищеності. Шифр RSA, що використовується для шифрування з використанням публічного ключа, наприклад, може використовувати лише підмножину всіх можливих значень для ключа даної довжини, все через природу математичної проблеми (задачі), на якій він базується. Інші шифри, такі як ті, що використовуються для симетричного

шифрування, можуть використовувати всі можливі значення для ключа даної довжини, а не підмножину цих значень.

Так як, зламати ключ шифрування RSA є досить незначним завданням, то шифрування RSA з використанням публічного ключа має використовувати дуже довгі ключі, як мінімум 1024 біти, щоб бути впевненим, що рівень захисту досить високий, з криптографічної точки зору. Але з іншого боку, шифрування з використанням симетричного ключа може отримати приблизно той же рівень захисту з використанням всього лише 80-бітного ключа для більшості алгоритмів [4].

1.1.3. Аналітичний огляд існуючих та перспективних рішень об’єкту проектування

Програми для шифрування служать відмінним засобом для захисту даних від інших користувачів і від вірусів. Як правило програми для шифрування створюють зашифровані сховища даних, відкрити які можна тільки при наявності пароля. Одна програма може підтримувати шифрування за кількома алгоритмам - на вибір користувача. У функціонал також може входити повне або часткове шифрування розділів жорсткого диска. При шифруванні розділу, всі файли на ньому паче не будуть доступні, тому якщо це системний розділ, то при його шифруванні операційна системи з нього не завантажиться. Крім того, деякі програми мають додаткові функції, які дозволяють, наприклад безповоротно видаляти файли [5].

Програма TrueCrypt. Це одна з найвідоміших програм для шифрування даних, повністю відповідна ідеї того, що дійсно надійна програма для шифрування не може бути проприєтарною. TrueCrypt поширюється під вільною ліцензією з відкритим вихідним кодом. Вона підтримує алгоритми шифрування AES, Serpent і Twofish і кілька різних хеш-функцій для створення ключів. Дані шифруються «на льоту», і поміщаються в спеціальний контейнер, який можна підключати як зовнішній диск, а також копіювати або переносити на інші носії.

Крім того, є можливість створювати зашифровані розділи на жорстких дисках або на USB-флешках.

Ключові особливості та функції створює віртуальний зашифрований диск в файл і встановлює його як реальний диск, доступний системі і всіх програм; шифрує весь пристрій зберігання або розділ, наприклад, USB накопичувач або жорсткий диск; можливо шифрування диска, на якому встановлено Windows. Для завантаження, в такому випадку, буде вимагатися введення пароля; шифрування є автоматичним і в реальному часі. Безперервно і прозоро; забезпечує два рівня ймовірної захисту. Це допоможе в разі, якщо противник змушує вас показати пароль: прихований розділ і прихована операційна система, TrueCrypt не може бути ідентифікований (обсяги даних не можна відрізнити від випадкових даних). таким чином, забезпечується надійний захист даних і неможливість ідентифікації TrueCrypt на диску або накопичувачі; алгоритми шифрування: AES-256, Serpent і Twofish. Режим роботи: XTS. зашифрувати можна змінний накопичувач, розділ жорсткого диска або тільки вибрані дані. Для даних створюється віртуальний диск і далі працювати з даними можна безпосередньо, не піклуючись про шифрування [5].

WinMend Folder Hidden - це програма, яка дозволяє "приховувати" файли і папки на локальних і зовнішніх носіях. Призначена вона, в першу чергу, для домашнього використання і не претендує на звання ультимативного засоби шифрування особливо важливих даних. Так що розробники не рекомендує її як рішення для комерційних організацій і підприємств з підвищеним рівнем "секретності".

Користуватися WinMend Folder Hidden дуже просто. Для того, щоб "приховати" файл або папку просто перетягніть його / її у вікно програми. У цьому ж вікні можна швидко змінювати статус файлів і папок з "прихованого" на "видимий", сортувати об'єкти за різними параметрами і здійснювати над ними масові операції. Файли на USB-накопичувачах, які ви приховали за допомогою даної програми, залишаться такими і при підключенні до іншого комп'ютера. Для того щоб їх побачити, необхідно буде встановити WinMend

Folder Hidden і ввести майстер-пароль. Цей майстер-пароль налаштовується в програмі при першому запуску і запитується при кожному наступному вході в неї. Системи відновлення пароля в програмі не передбачено, так що обов'язково запам'ятайте (або запишіть) пароль, щоб не позбутися доступу до файлів. WinMend Folder Hidden працює з будь-якими файловими системами і типами носіїв. "Заховані" з її допомогою дані неможливо "побачити" з сторонніх програм і під іншими обліковими записами. Програма є абсолютно безкоштовною і не містить реклами.

Ключові особливості та функції: дозволяє приховувати файли і папки на локальних і знімних носіях; дає можливість швидко змінювати статус об'єктів зі "прихованого" на "видимий" і назад; вхід в програму здійснюється за обраним "майстер-паролем"; працює з будь-якими файловими системами і типами носіїв; є абсолютно безкоштовною, не містить реклами [5].

PGP Desktop - це одна з найпотужніших програм для захисту і шифрування даних. Вона може зашифрувати ділову переписку, електронну пошту, текстову інформацію, а також файли і цілі жорсткі диски.

Через особливості асиметричного шифрування для роботи з програмою спочатку необхідно створити ключову пару, що складається з відкритого і приватного ключа, які в подальшому будуть використовуватися для шифрування, а також для установки і верифікації цифрових підписів. Серед основних налаштувань потрібно вказати алгоритми шифрування і хешування, термін «життя» і довжину ключа, а також ввести кодову фразу, яка буде використана для доступу до контейнера з ключами. Все це програма попросить зробити при першому запуску. Після установки PGP Desktop інтегрується з провідником і додає нові пункти меню в контекстне меню Windows. З їх допомогою можна швидко зашифрувати файли або каталоги. Вибрані елементи містяться в контейнер з розширенням. PGP. До речі, в налаштуваннях програми можна активувати стиснення контейнерів, в цьому випадку вони будуть займати менше місця і отримають додатковий рівень захисту від криптоаналізу. Всі дії з контейнерами записуються в журнал, з якого користувач в будь-який

момент може отримати дані про те, хто і коли створював або відкривав зашифровані архіви.

Крім усього іншого, програма може створювати віртуальні шифровані диски, які можна підключити і працювати з ними так само, як і зі звичайними дисками, а також шифрувати розділи і затирати вільний простір, роблячи неможливим відновлення видалених даних. Крім того, після установки на робочому столі з'явиться кошик PGP Shredder, який служить для безповоротного видалення: файли, поміщені в нього, будуть знищені, а простір, який вони займали, заб'ється нулями. Незважаючи на настільки широкий функціонал, працювати з PGP Desktop досить просто, інтерфейс не викличе складнощів в освоєнні, майстри настройки допоможуть виконати основні операції, а на випадок виникнення питань існує розгорнута система допомоги.

Ключові особливості та функції надійне шифрування даних асиметричними алгоритмами; можливість встановлювати і верифікувати цифрові підписи і використовувати різні алгоритми хешування; зручні інструменти для управління ключами; можливість безпечного обміну файлами і каталогами; ведення журналу всіх операцій з захищеними контейнерами; можливість шифрування електронної пошти, листування в AOL Instant Messenger, і будь-яких текстових даних у відкритих вікнах; наявність інструментів для безповоротного видалення даних; зручний інтерфейс, в якому всі інструменти «розкладені» по розділах [5].

1.2. Призначення розробки та галузь застосування

Основним призначенняи розробки та галуззю застосування додатку, що розробляється в кваліфікаційній роботі є шифрування та дешифрування текстових повідомлень. Даний додаток розробляється з метою підвищення зручності та ефективності шифрування та дешифрування текстових повідомлень різними методами.

Зазначена розробка може бути використана будь-якими користувачами та не потребує спеціальних знать та навичок.

Також даний додаток можна використати в навчальному процесі при вивченні відповідних тем з криптографічного захисту інформації.

1.3. Підстава для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Отже, підставами для розробки та виконання кваліфікаційної роботи є: освітня програма спеціальності 121 «Інженерія програмного забезпечення» та навчальний план та графік навчального процесу та наказ ректора Національного технічного університету «Дніпровська політехніка» № 317-с від 07.06.2021 р. та завдання на кваліфікаційну роботу на тему: «Розробка програмного забезпечення для шифрування та дешифрування текстових повідомлень».

1.4. Постановка завдання

Основною метою кваліфікаційної роботи ставилося підвищення зручності та ефективності шифрування та дешифрування текстових повідомлень різними методами за рахунок розробки відповідного програмного додатку. Отже для досягнення мети в роботі ставляться наступні задачі:

- аналіз існуючих аналогічних програм для шифрування / дешифрування;
- аналіз методів шифрування/дешифрування;
- розробка структури та алгоритму роботи програми;
- вибір засобів для програмной реалізації додатку;
- програмна реалізація та тестування додатку;
- оформлення відповідної документації.

Також ставляться додаткові вимоги та завдання:

- програма повинна не бути складною та мати відкритий код;
- мати зрозумілий та визначений інтерфейс;
- мати в своєму арсеналі найпопулярніші системи шифрування.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Основними функціями додатку що розробляється є шифрування та дешифрування текстових повідомлень різними методами. Таким чином до функціональних характеристик ставляться такі вимоги:

- повинна мати в своєму арсеналі популярні методи шифрування/дешифрування та працювати у відповідному режимі;
- програма повинна надавати можливість користувачу працювати з текстовими файлами та/чи введеними в системі повідомленнями;
- програма повинна бути не складною та мати відкритий код;
- мати зручний та зрозумілий інтерфейс;
- бути невимогливою до складу програмно-апаратних засобів та кваліфікації користувача.

1.5.2 Вимоги до інформаційної безпеки

Для уникнення некоректної роботи програми необхідно реалізувати:

- контроль вхідних даних;
- обробку виняткових ситуацій;
- виведення повідомлень про помилки;
- можливість повторного введення даних;
- можливість безперервної роботи протягом не менше 120 годин (5 діб);

– забезпечення збереження та неушкодженого стану даних, що зберігаються в базі даних, у випадку відмови застосунку.

Особливих чи додаткових вимог до інформаційної безпеки додатку не висувається.

1.5.3 Вимоги до складу та параметрів технічних засобів

Програма не є вимогливою до складу та параметрів технічних засобів та може завантажуватись ПК чи ноутбуках під управлінням ОС Windows.

Системні вимоги:

- Графічний інтерфейс (наявність монітора).
- ОС Windows XP,7,8,10.
- Джерелом введення інформації є клавіатура.
- Результати роботи виводяться на екран та заносяться у базу даних. (достатне місце для збереження інформації).
- Обов’язкове збереження даних перед закінченням роботи (достатне місце для збереження інформації).

1.5.4 Вимоги до інформаційної та програмної сумісності

Програма не є вимогливою до інформаційної та програмної сумісності.

Для нормального функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде функціонувати система, відповідало наступним вимогам:

- ✓ операційна система сімейства Windows (7, 8, 10).

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Розроблений в кваліфікаційній роботі програмний продукт в першу чергу призначений для підвищення надійності передачі або зберігання інформації за рахунок створення та використання додатку для шифрування-дешифрування текстових повідомлень. Також функціональним призначенням додатку є забезпечення роботи програми в реживі відповідно до обраного метода шифрування/дешифрування та надання можливості користувачу працювати з текстовими файлами та/чи введеними в системі повідомленнями.

Також даний додаток може використовувати викладач чи студент для підвищення продуктивності перевірки завдань з криптографії в освітньому процесі.

2.2. Опис застосованих математичних методів

В роботі було використано відповідні математичні методи для втілення наступних функцій:

- Шифр Цезаря.
- Шифр Віженера та гамування.
- Шифр перестановки.
- Хешування MD5.
- Створення QR-коду.
- Цифровий підпис.
- DES-шифрування.
- RSA-шифрування.

2.3. Опис використаної архітектури та шаблонів проектування

При розробці додатку було використано технологію .NET та архітектуру MVC. MVC (Model-view-controller) Модель–представлення–контролер — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону - гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

2.4. Опис використаних технологій та мов програмування

Для створення додатку мною була обрана мова програмування C#. C# - це об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET що дозволяє створювати додатки з об'єктно-орієнтованого підходу, що є максимально оптимальним рішенням.

Microsoft .NET - програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java [12].

Рис. 2.1. Стек технологій .NET

Кожна бібліотека (збірка) в .NET має свідчення про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок.

В якості середовища розробки було обране інтегроване середовище розробки Microsoft Visual Studio. Тому що це середовище максимально надійне і стабільне, є найбільший досвід програмування саме в цьому середовищі, та тому що середовище дає змогу розробляти додатки за допомогою технології Windows Forms[5].

2.5. Опис структури програми та алгоритмів її функціонування

За допомогою шифрування Цезаря можна як шифрувати так і розшифровувати повідомлення.

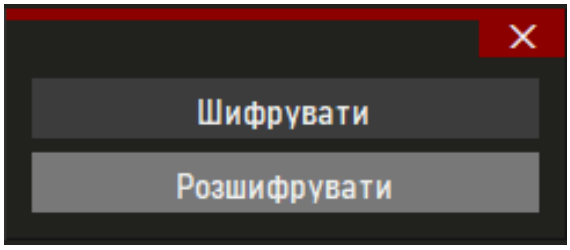

Рис. 2.2. Шифрування або розшифровування шифром Цезаря

Шифр Цезаря або шифр зсуву — симетричний моноалфавітний алгоритм шифрування, в якому кожна буква відкритого тексту заміняється на ту, що віддалена від неї в алфавіті на сталу кількість позицій [6].

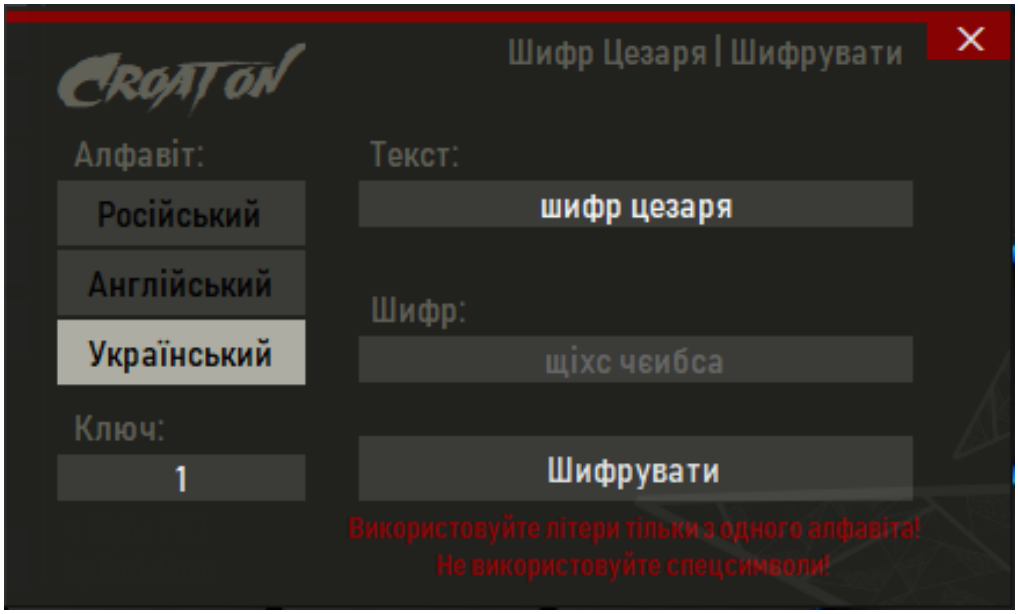

Рис. 2.3. Шифрування повідомлення

У програмі “Croaton” шифром Цезаря можна зашифрувати або розшифрувати повідомлення на російській, англійській і українській мовах.

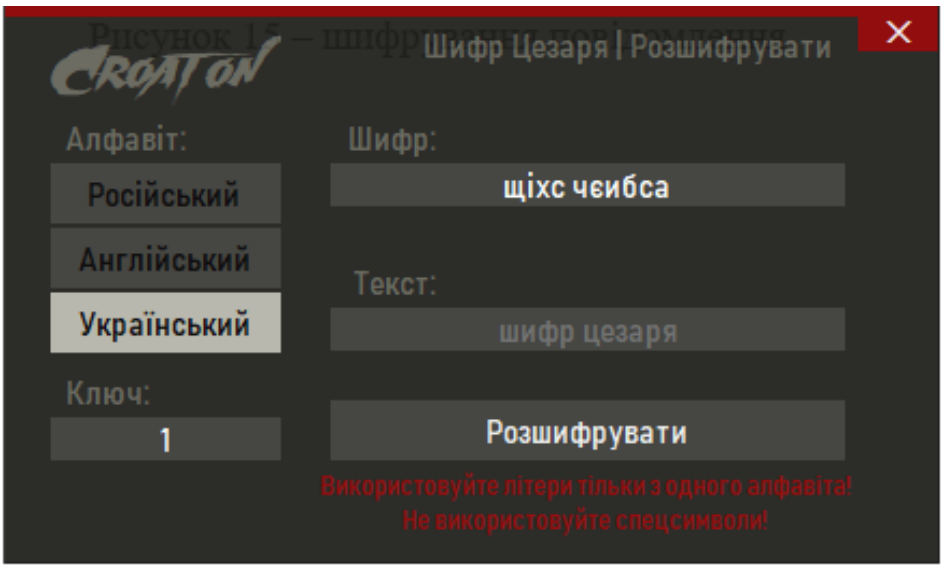

Рис. 2.4. Розшифрування повідомлення

Лістинг даного шифрування представлено в додатку А.

Шифр Віженера — поліалфавітний шифр, який за ключ використовує слово [7].

Шифр XOR або гамування — алгоритм шифрування, у якому як ключ використовується ключове слово та може бути записаний формулою. Ключове слово повторюється поки не отримано гаму, рівну довжині повідомлення [8].

Для роботи з цими типами шифру для зручності використання різного об’єму повідомлення потрібно записувати текс у файл in.txt що знаходиться у кореневій папці з програмою.

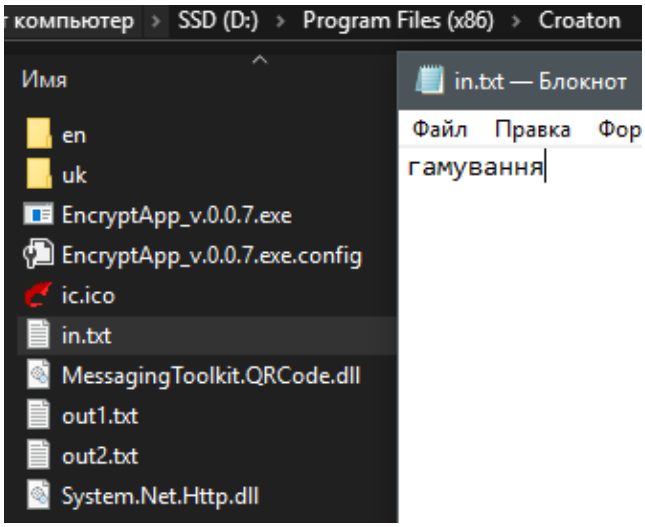

Рис. 2.5. Коренева папка з файлом in.txt

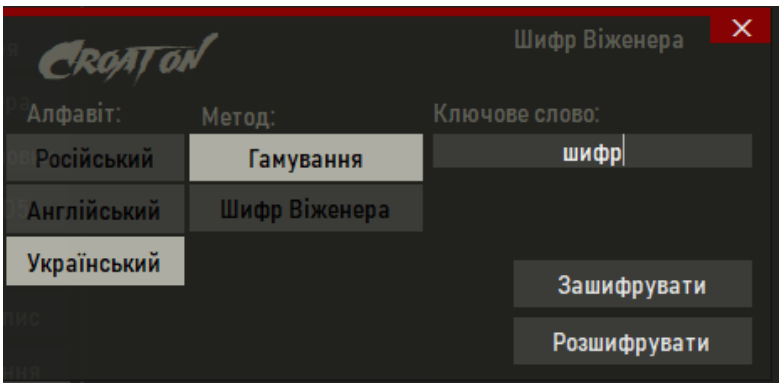

Рис. 2.6. Вікно шифрування

При шифруванні відбувається запис зашифрованого повідомлення у файл out1.txt, а запис розшифрованого повідомлення у файл out2.txt. Тобто для розшифрування, помістіть зашифроване повідомлення у файл out1.txt.

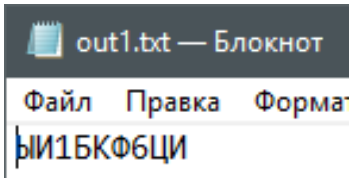

Рис. 2.7. Зашифроване повідомлення гамуванням

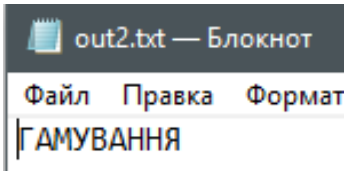

Рис. 2.8. Розшифроване повідомлення гамуванням

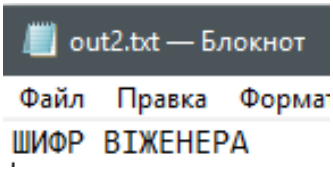

Рис. 2.9. Зашифроване повідомлення шифром Віженера

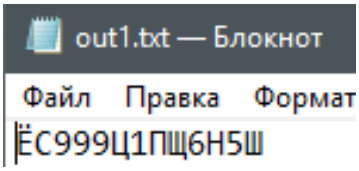

Рис. 2.10. Розшифроване повідомлення шифром Віженера

Лістинг представлено в додатку А.

Перестановочний шифр — алгоритм шифрування, який полягає у перестановці знаків відкритого тексту згідно з певним правилом, яке є ключем [9].

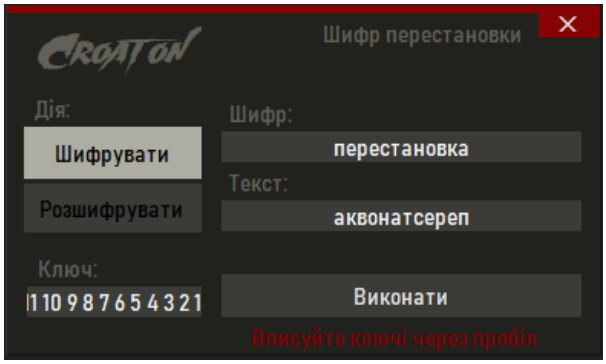

Рис. 2.11. Шифрування перестановкою

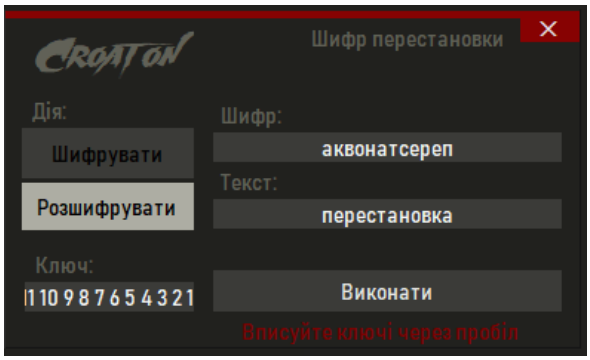

Рис. 2.12. Розшифрування перестановкою

Хешування MD5 . MD5 (Message Digest 5) - 128-бітний алгоритм хешування. Призначений для створення «відбитків» або «дайджестів» повідомлень довільної довжини. Хешування являє собою асиметричний алгоритм, який неможливо розшифрувати у зворотному напрямк [10]. Використання MD5 передбачає собою створення хешу пароля, схему прикладу зображено на рис. 2.13.

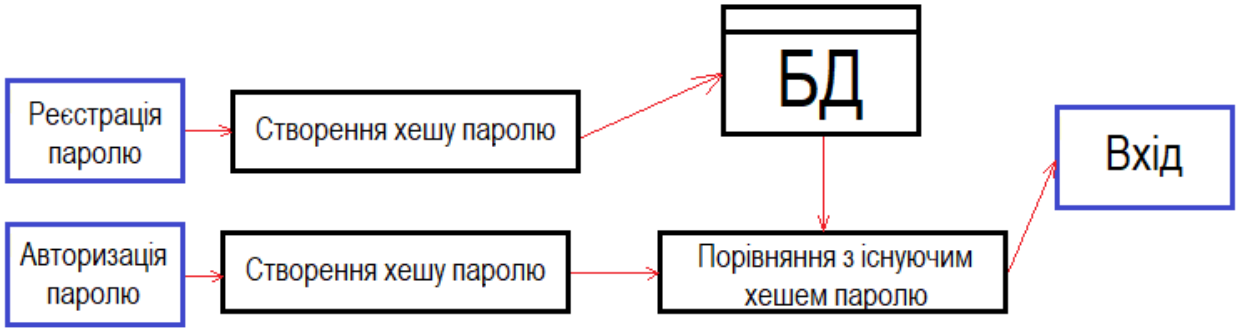

Рис. 2.13 Приклад використання хешу

Для реалізації хешування використовувалася системний простір імен .NET Framework “System.Security.Cryptography”. Лістинг представлено в додатку А.

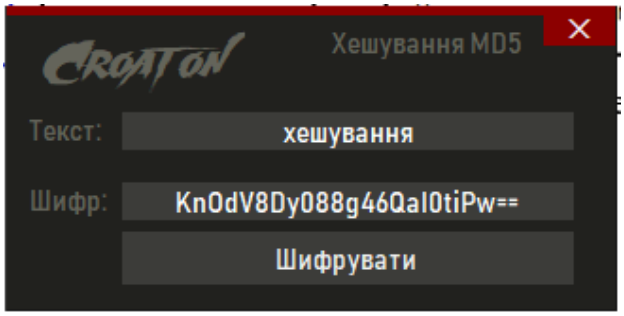

Рис. 2.14. Хешування MD5

Створення QR-коду. QR-код (англ. quick response — швидкий відгук) — матричний код (двовимірний штрих-код), розроблений і представлений японською компанією «Denso-Wave» в 1994 році.

Основна перевага QR-коду — це легке розпізнавання сканувальним обладнанням (в тому числі й фотокамерою мобільного телефона), що дає можливість використання в торгівлі, на виробництві, та логістиці[11].

Програма “Croaton” має в своєму арсеналі створення, збереження та розпізнавання QR-коду.

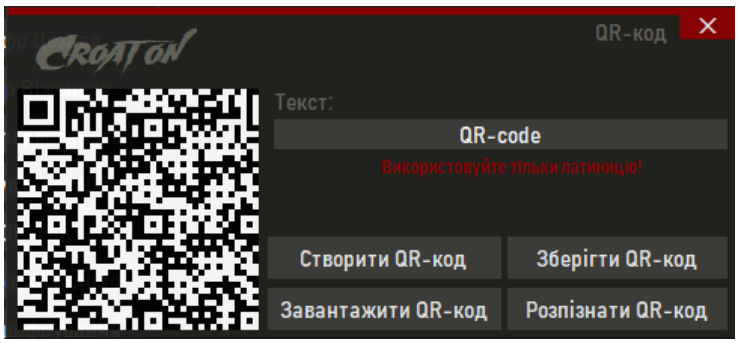

Рис. 2.15. Створення QR-коду

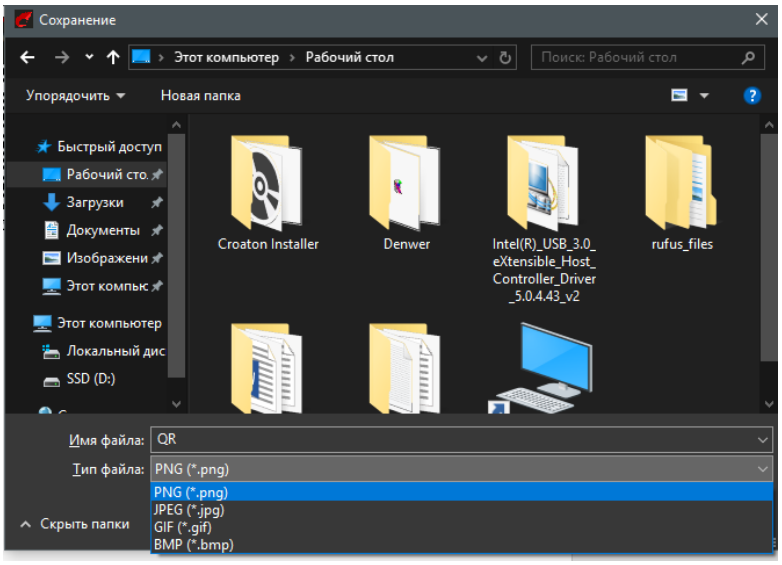

Рис. 2.16. Збереження QR-коду у форматах png, jpeg, gif, bmp

Для роботи з QR-кодом було використано не систему бібліотеку MessagingToolkit.QRCode.dll. Детальний лістинг в додатку А.

Електронний цифровий підпис (ЕЦП) (англ. digital signature) — вид електронного підпису, отриманого за результатом криптографічного перетворення набору електронних даних, який додається до цього набору або логічно з ним поєднується і дає змогу підтвердити його цілісність та ідентифікувати підписувача. Електронний цифровий підпис накладається за допомогою особистого ключа та перевіряється за допомогою відкритого ключа [12].

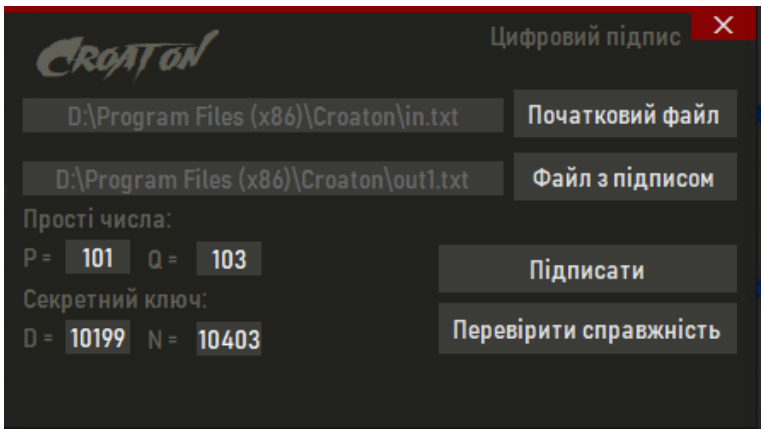

Рис. 2.17. Процес підписання файлу in.txt

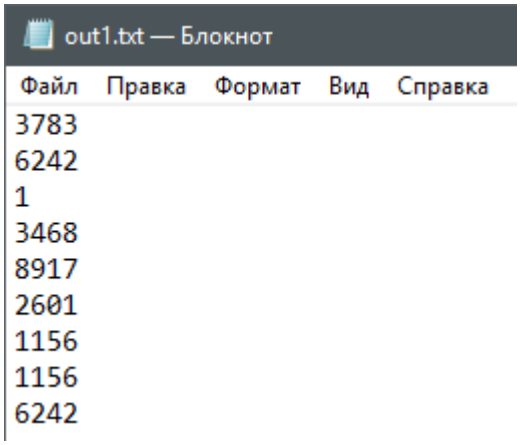

Рис. 2.18. Створення підпису прив’язаного до файлу

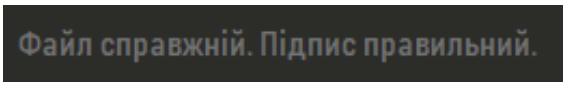

Рис. 2.19 Перевірка файлу з раніше створеним підписом

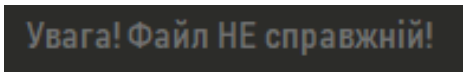

Рис. 2.20. Повідомлення при перевірці файлу з раніше створеним підписом але не правильним секретним ключем

DES-шифрування. DES (англ. Data Encryption Standard) — це симетричний алгоритм шифрування певних даних , стандарт шифрування

прийнятий урядом США із 1976 до кінця 1990-х, з часом набув міжнародного застосування [13].

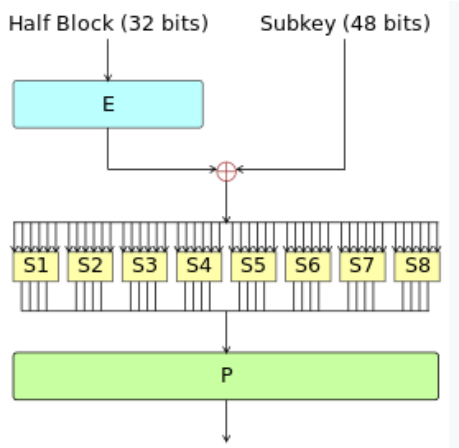

Рис. 2.21. Схема алгоритму блочного шифрування

DES є блочним шифром - дані шифруються блоками по 64 біти - 64 бітний блок явного тексту подається на вхід алгоритму, а 64-бітний блок шифрограми отримується в результаті роботи алгоритму. Крім того, як під час шифрування, так і під час дешифрування використовується один і той самий алгоритм (за винятком дещо іншого шляху утворення робочих ключів) [13].

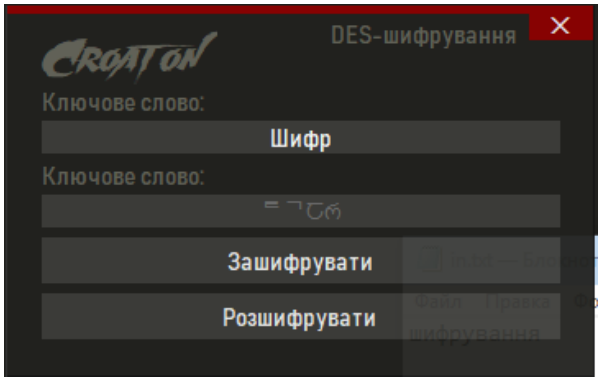

Рис. 2.22. Робота DES-шифрування

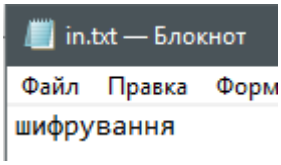

Рис. 2.23. Текст який було зашифровано за допомогою DES

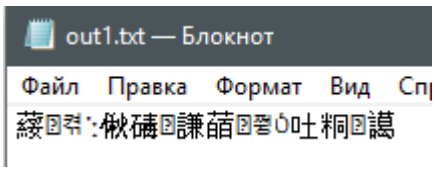

Рис. 2.24. Зашифроване повідомлення

RSA-шифрування. RSA (абревіатура від прізвищ Rivest, Shamir та Adleman) - криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел. RSA став першим алгоритмом такого типу, придатним і для шифрування, і для цифрового підпису. Алгоритм застосовується до великої кількості криптографічних застосунків [14].

Рис. 2.25. Схематичне зображення RSA-шифрування

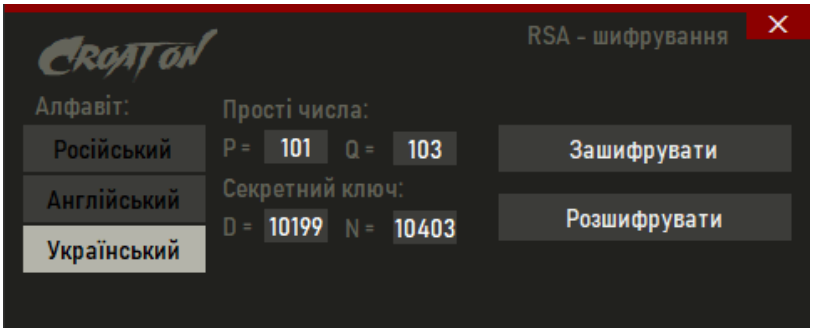

Рис. 2.26. RSA-шифрування

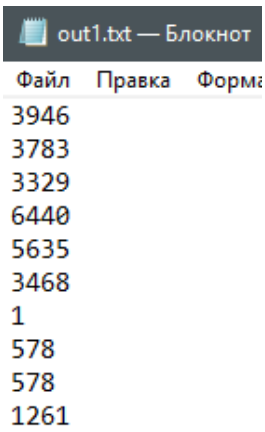

Рис. 2.27. Створений RSA шифр у файлі out1.txt

2.6. Обгрунтування та організація вхідних та вихідних даних програми

Вхідними даними є введені користувачем дані або обрані управляючі команди. Вихідними даними є дані, що отримує користувач від додатку.

2.7. Опис роботи розробленого програмного продукту

2.7.1. Використані технічні засоби

При розробці системи було використано Ноутбук Dell Vostro 15 3501 з наступними характеристиками: екран 15.6" WVA (1920x1080) Full HD, глянсовий з антивідблисковим покриттям / Intel Core i3-1005G1 (1.2 — 3.4 ГГц) / RAM 8 ГБ / SSD 256 ГБ / Intel UHD Graphics / без ОД / LAN / Wi-Fi / Bluetooth / вебкамера .

2.7.2. Використані програмні засоби

В якості середовища розробки було обране інтегроване середовище розробки Microsoft Visual Studio. Тому що це середовище максимально надійне і стабільне, є найбільший досвід програмування саме в цьому середовищі, та тому що середовище дає змогу розробляти додатки за допомогою технології Windows Forms [5].

2.7.3. Виклик та завантаження програми

Для забезпечення захисту передачі текстових даних або повідомлень або зберігання їх в окремих таблицях БД можна використовувати шифрування, що є більш надійним ніж збереження БД “під ключем” (БД захищена паролем користувача).

Додаток для шифрування має назву “Croaton”. Ця назва частково символічна і віддає дань індійському племені, яке використовувало це слово для пошуку своїх груп, та свого поселення. Це було свого роду шифром.

Для того щоб полегшити використання застосунку, було розроблено інсталяційний пакет Windows Installer.

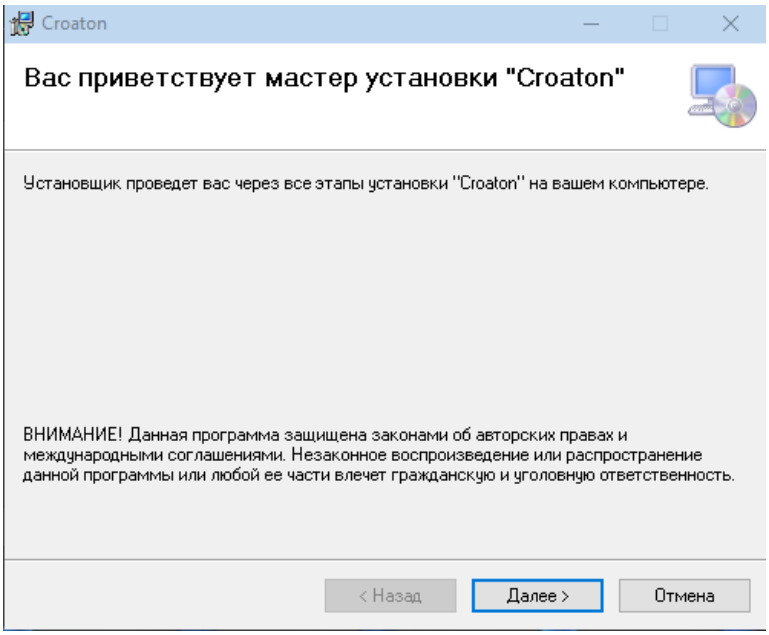

Рис. 2.28. Вікно при відкритті інсталятора програми

Для більш надійного використання додатку, раджу встановлювати додаток на не системний диск. Щоб запобігти враження програми вірусами, та для більш стабільної роботи.

Є думка, що кожен додаток повинен бути легкий у використанні, і не приносити зайвих проблем. Кожен додаток повинен при першій потребі

користувача бути видаленим. Якщо, додаток не запускається, або ви підозрюєте те що файли застосунку були пошкоджені або бажаєте видалити застосунок, запутіть інсталятор, та виберіть потрібний пункт меню.

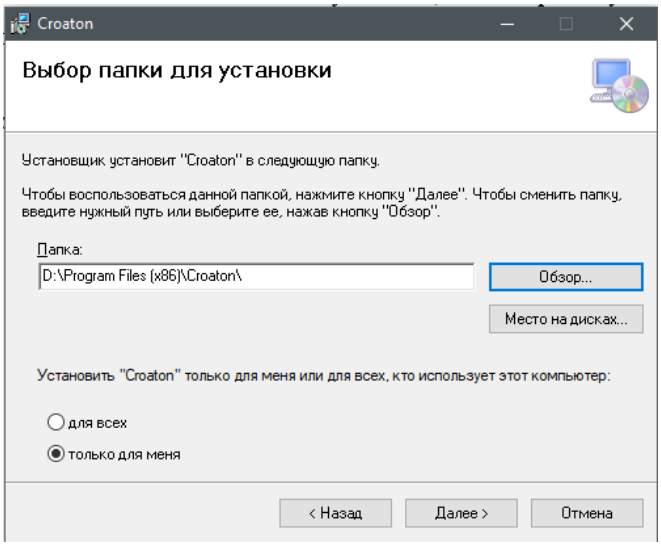

Рис. 2.29. Шлях встановлення програми

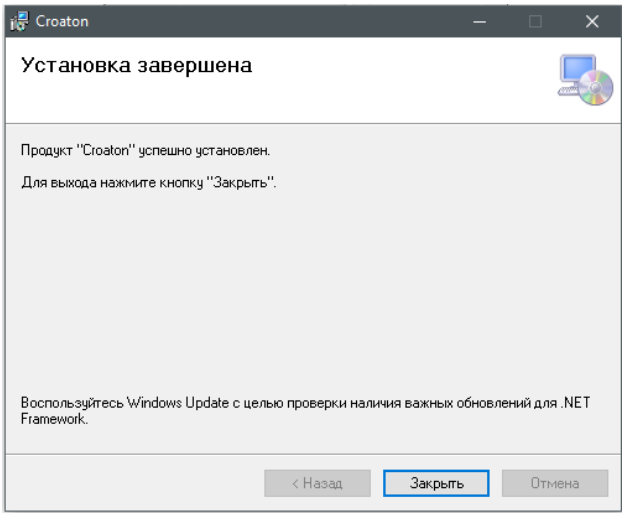

Рис. 2.30. Завершення інсталяції програми

Даний додаток може використовуватися на ОС Windows 7 та вище.

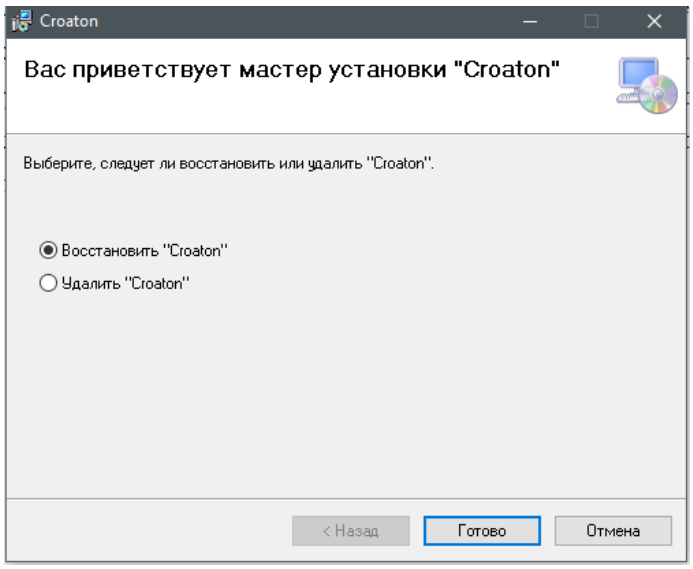

Рис. 2.31. Деінсталяція або відновлення додатку

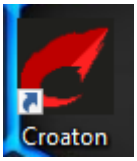

Рис. 2.32. Ярлик програми на “Робочому столі”

Рис. 2.33. Ярлик програми у меню “Пуск”

Після інсталяції додатку, користувач має доступ до програми за допомогою ярлика, або за допомогою меню “Пуск”.

2.7.4. Опис інтерфейсу користувача

Програма розроблена в червоно-сірій цвітовій гамі. Основний шрифт – Arial Nova. Для більш стабільного запуску додатку (щоб програма не зависала на початковому вікні) була розроблена форма завантаження програми.

Рис. 2.34. Вікно завантаження програми

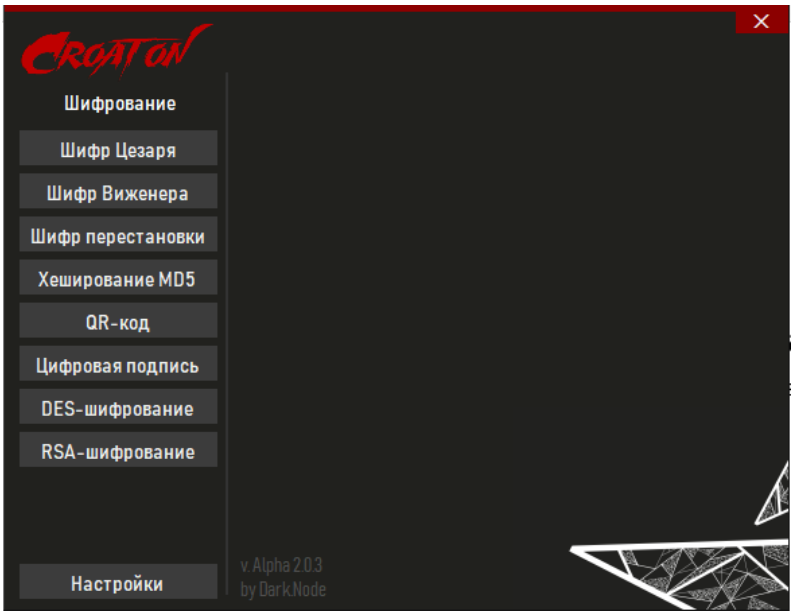

Рис. 2.35. Головне меню застосунку

У вкладці “Налаштування” можна змінити мову застосунку, та продивитися інструкцію користувача, в якій описано як правильно користуватися програмою.

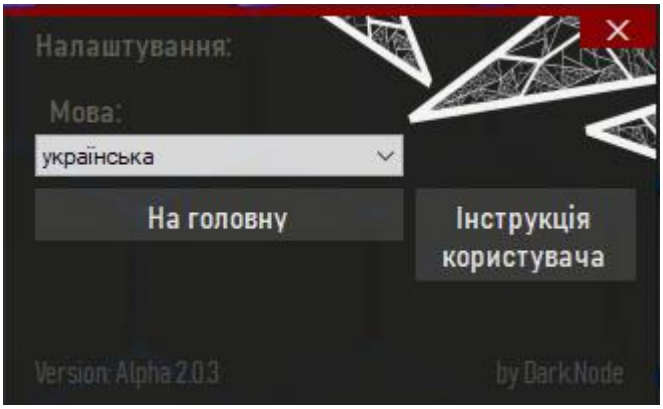

Рис. 2.36. Меню налаштування

Програма “Croaton” повністю переведена на 3 мови: англійська, українська, російська. Лістинг створення головного меню та коду переключення мов, знаходиться у додатку А.

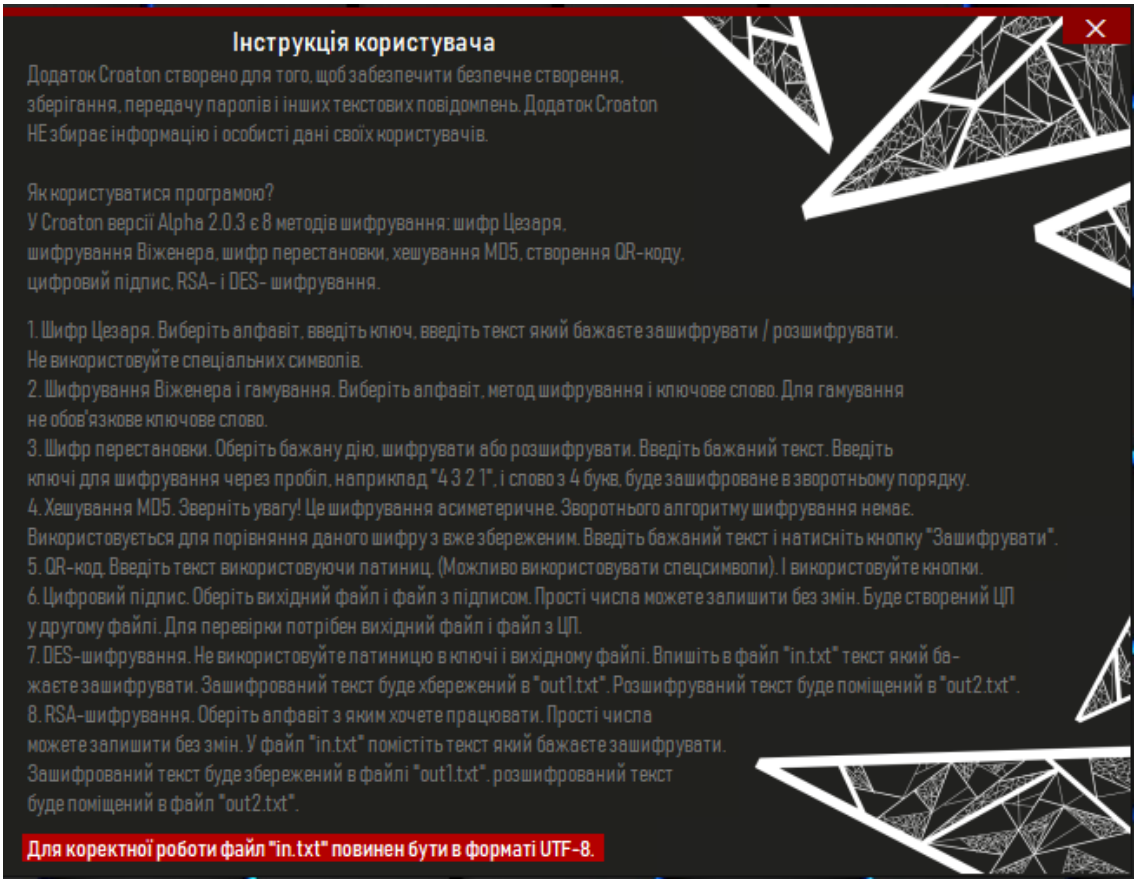

Рис. 2.37. Інструкція користувача

Щоб запобігти випадкового виходу с програми, з’явиться спеціальне діалогове вікно.

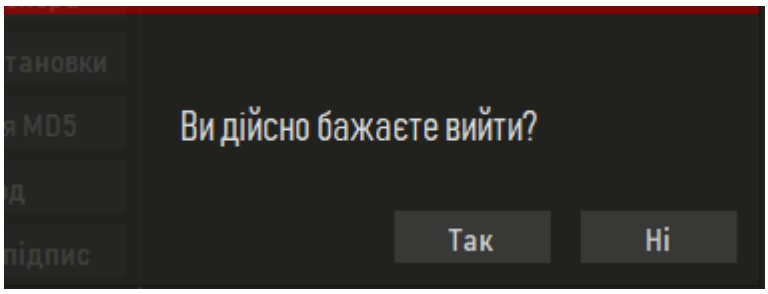

Рис. 2.38. Діалогове вікно

Роботу додатку при шифруванні та розшифруванні різними методами більш детально розглянуто в розділі 2.5. «Опис структури програми та алгоритмів її функціонування».

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми – 1400;
2. коефіцієнт складності програми – 1,8;
3. коефіцієнт корекції програми в ході її розробки – 0,07;
4. годинна заробітна плата програміста – 75 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,3;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,3;
7. вартість машино-години ЕОМ – 20 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмногму забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p),$$

де q - передбачуване число операторів (1400);

C - коефіцієнт складності програми (1,8);

p - коефіцієнт корекції програми в ході її розробки (0,07).

Звідси умовне число операторів в програмі:

$$Q = 1{,}8 \cdot 1400 \cdot (1 + 0{,}07) = 2696{,}4$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}\ \text{, людино-годин,}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 3 до 5 років він складає 1,2.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1.3). З урахуванням коефіцієнта кваліфікації k = 1,3, отримуємо витрати праці на вивчення опису завдання:

$$t_u = (2696{,}4 \cdot 1{,}3) / (75 \cdot 1{,}3) = 35{,}95 \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин,}\quad (3.2)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.2), отримаємо:

$$t_a = 2696{,}4 / (20 \cdot 1{,}3) = 103{,}7 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20\ldots25)\cdot k}\ \text{, людино-годин.}$$

$$t_n = 2696{,}4 / (25 \cdot 1,\ 3) = 82{,}96 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5)\cdot k}\ \text{, людино-годин.}$$

$$t_{отл} = 2696{,}4 / (5 \cdot 1{,}3) = 414{,}83 \text{ чел.-ч.}$$

- за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5 \cdot t_{отл}\ \text{, людино-годин.}$$

$$t^k_{отл} = 1{,}5 \cdot 414{,}83 = 622{,}245 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_д = t_{др} + t_{до}\ ,\ \text{людино-годин},$$

де $t_{др}$ - трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k}\ ,\ \text{людино-годин},$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0{,}75 \cdot t_{др}\ ,\ \text{людино-годин}.$$

Підставляючи відповідні значення, отримаємо:

$t_{др}$ = 2696,4 / (15 · 1,3) = 138,28 людино-годин.

$t_{до}$ = 0,75 · 138,28 = 103,71 людино-годин.

$t_д$ = 138,28 + 103,71 = 241,99 людино-годин.

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

t = 50 + 35,95 + 103,7 + 82,96 + 414,83 + 241,99 = 929,43 людино-годин.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ:

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.}$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,}$$

де: t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 60 грн / год, отримуємо:

$$З_{ЗП} = 929{,}43 \cdot 75 = 69\,707{,}25 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (3.3)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мч}$ - вартість машино-години ЕОМ, грн/год (20 грн/год).

Підставивши в формулу (3.3) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 414{,}83 \cdot 20 = 8296{,}6 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$K_{ПО} = 69\ 707{,}25 + 8296{,}6 = 78\ 003{,}85 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.}$$

де B_k - число виконавців (дорівнює 1);

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

Звідси витрати на створення програмного продукту:

$$T = 929{,}43 / 1 \cdot 176 = 5{,}28 \text{ міс.}$$

Висновок. Програмне забезпечення призначене для шифрування повідомлень різними методами. Розроблене програмне забезпечення є мобільним додатком органайзером. Вартість даного програмного забезпечення становить майже 78 тис. грн. і не вимагає додаткових витрат як при розробці, так і при впровадженні програми. Очікуваний час розробки становить 5,28 місяця. Цей термін пов'язаний зі значним числом операторів, і включає час на дослідження і розробку алгоритму вирішення поставленого завдання, програмування по готовому алгоритму, налагодження програми і підготовку документації.

ВИСНОВКИ

В даній кваліфікаційній роботі була створена програма для шифрування та дезшифровування текстових повідомлень “Croaton”.

В роботі розглядаються основні принципи шифрування, основи криптографії, порівняльна характеристика програм аналогів, створення додатків на платформі .NET.

Також в роботі було розроблено програму для шифрування та рдешифровування текстових повідомлень “Croaton”, було створено та описано інсталятор програми для більш зручного використання.

Були описані типи шифрування такі як: шифрування Цезаря, шифрування перестановкою, шифрування Віженера та гамування, хешування MD5, QR-код, цифровий підпис, DES-шифрування, RSA-шифрування. У додатках наведено лістинг всіх типів шифрування.

Для зручності користувача було також створено інструкцію користувача, та меню зміни мови у застосунку.

Позитивні сторони проекту: добре підібрана кольорова гама робить його більш приємним для сприйняття. Вибір мови дозволяє поліпшити розуміння користувачу. Інсталяційний пакет, забезпечує зручність у використанні. Відритий програмний код, дозволяє модифікувати додаток іншими розробниками.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Основи КЗІ [Електронний ресурси]. – Режим доступу: http://www.immsp.kiev.ua/postgraduate/Biblioteka_trudy/Osnovy_KZI_Gulak.G.M_Muchatchev.V.A_2011.pdf

2. Аналіз методів криптографічного захисту інформації [Електронний ресурси]. – Режим доступу: http://ir.nmu.org.ua/bitstream/handle/123456789/149264/6-7.pdf?sequence=1&isAllowed=y

3. Розроблення методів шифрування електронних документів за допомогою АТЕФ-функцій [Електронний ресурси]. – Режим доступу: http://vlp.com.ua/files/19_18.pdf

4. .NET Framework [Електронний ресурси]. – Режим доступу: https://uk.wikipedia.org/wiki/.NET_Framework

5. Шифр Цезаря [Електронний ресурси]. – Режим доступу: https://uk.wikipedia.org/wiki/Шифр_Цезаря

6. Шифр Віженера [Електронний ресурси]. – Режим доступу: https://uk.wikipedia.org/wiki/Шифр_Віженера

7. Шифр XOR [Електронний ресурси]. – Режим доступу: https://uk.wikipedia.org/wiki/Шифр_XOR

8. Перестановочний шифр [Електронний ресурси]. – Режим доступу: https://uk.wikipedia.org/wiki/Перестановочний_шифр

9. MD5 [Електронний ресурси]. – Режим доступу: https://uk.wikipedia.org/wiki/MD5

10. QR-код [Електронний ресурси]. – Режим доступу: https://uk.wikipedia.org/wiki/QR-код

11. Електронний цифровий підпис [Електронний ресурси]. – Режим доступу: https://uk.wikipedia.org/wiki/Електронний_цифровий_підпис

12. Data Encryption Standard [Електронний ресурси]. – Режим доступу: https://uk.wikipedia.org/wiki/Data_Encryption_Standard

13. RSA [Електронний ресурси]. – Режим доступу: https://uk.wikipedia.org/wiki/RSA

14. Агуров Павел C#. Сборник рецептов / Павел Агуров. - М.: "БХВ-Петербург", 2012. - 432 с.

15. Албахари Джозеф C# 3.0. Справочник / Джозеф Албахари , Бен Албахари. - М.: БХВ-Петербург, 2013. - 944 с.

16. Альфред В. Ахо Компиляторы. Принципы, технологии и инструментарий / Альфред В. Ахо и др. - М.: Вильямс, 2015. - 266 с.

17. Бишоп Дж. C# в кратком изложении / Дж. Бишоп, Н. Хорспул. - М.: Бином. Лаборатория знаний, 2013. - 472 с.

18. Вагнер Билл C# Эффективное программирование / Билл Вагнер. - М.: ЛОРИ, 2013. - 320 с.

19. Зиборов В.В. Visual C# 2012 на примерах / В.В. Зиборов. - М.: БХВ-Петербург, 2013. - 480 с.

20. Ишкова Э.А. Самоучитель C#. Начала программирования / Э.А. Ишкова. - М.: Наука и техника, 2013. - 496 с.

21. Лотка Рокфорд C# и CSLA .NET Framework. Разработка бизнес-объектов / Рокфорд Лотка. - М.: Вильямс, 2010. - 816 с.

22. Подбельский В.В. Язык C#. Базовый курс / В.В. Подбельский. - М.: Финансы и статистика, Инфра-М, 2011. - 384 с.

23. Рихтер Джеффри. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C# / Джеффри Рихтер. - М.: Питер, 2013. - 928 с.

24. Троелсен Эндрю. Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. - 486 с.

25. Фримен Адам. ASP.NET MVC 3 Framework с примерами на C# для профессионалов / Адам Фримен , Стивен Сандерсон. - М.: Вильямс, 2011. - 672 с.

26. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп’ютерні науки / Л.М. Коротенко ,

О.С. Шевцова; Нац. гірн. ун-т. – Д : НТУ «Дніпровська політехніка», 2018. – 65 с.

27. Середня заробітна плата Junior FE developer в Україні станом на початок 2021 року. https://dou.ua/lenta/articles/salary-report-devs-june-2020/

28. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : ДСТУ 3008-95. – Чинний від 1996–01–01. – К. : Держстандарт України, 1996. – 39 с.

29. ДСТУ 2394-94 Інформація та документація. Комплектування фонду, бібліографічний опис, аналіз документів. Терміни та визначення. – Чинний від 01.01.1995. — Київ : Держстандарт України, 1994. – 88 с.

30. Мальчук Е.В. HTML и CSS. Самоучитель / Е. В. Мальчук – М.: Вильямс, 2008. – 416 с.

31. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності «Комп'ютерні системи» / О.Г. Вагонова, О.Б. Нікітіна, Н.Н. Романюк; М-во освіти і науки України, ДВНЗ «Нац. гірн. ун-т». – Д.: НГУ, 2013. – 11 с.

ДОДАТОК А

КОД ПРОГРАМИ

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

namespace EncryptApp_v._0._0._7
{
   public partial class Form1 : Form
   {
      public Form1() //Метод переключення мови
      {
         if (!String.IsNullOrEmpty(Properties.Settings.Default.Language))
         {
            System.Threading.Thread.CurrentThread.CurrentUICulture =
System.Globalization.CultureInfo.GetCultureInfo(Properties.Settings.Default.Langua
ge);
System.Threading.Thread.CurrentThread.CurrentCulture =
System.Globalization.CultureInfo.GetCultureInfo(Properties.Settings.Default.Langua
ge);
         }
         InitializeComponent();
      }

      protected override void WndProc(ref Message message) //Метод який дозволяє
переміщувати форму за вільну область
      {
         if (message.Msg == 0x201)
         {
            base.Capture = false;
            message = Message.Create(base.Handle, 0xA1, new IntPtr(2), IntPtr.Zero);
```

```
            }
            base.WndProc(ref message);
        }
private void panel1_Paint(object sender, PaintEventArgs e){}
private void label1_Click(object sender, EventArgs e){}
private void label2_Click(object sender, EventArgs e){}
private void button1_Click(object sender, EventArgs e)
{Form2 newForm = new Form2(); newForm.ShowDialog();}
private void Form1_Load(object sender, EventArgs e){}
private void button3_Click(object sender, EventArgs e)
{Form5 newForm = new Form5();newForm.Show();}
private void button4_Click(object sender, EventArgs e)
{Form6 newForm = new Form6(); newForm.Show();}
private void button5_Click(object sender, EventArgs e)
{Form7 newForm = new Form7(); newForm.Show();}
private void button6_Click(object sender, EventArgs e)
{Form8 newForm = new Form8(); newForm.Show();}
private void button7_Click(object sender, EventArgs e)
{Form9 newForm = new Form9();newForm.Show();}
private void button8_Click(object sender, EventArgs e)
{Form10 newForm = new Form10(); newForm.Show();}


        private void button9_Click(object sender, EventArgs e)
        {
            Form11 newForm = new Form11();
            newForm.Show();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            Form12 newForm = new Form12();
            newForm.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form13 newForm = new Form13();
```

```
            newForm.Show();
        }

        private void label1_Click_1(object sender, EventArgs e) //Відкриття сторінки у браузері для зворотнього зв'язку
        {
            System.Diagnostics.Process.Start("https://www.instagram.com/dark.n0de/");
        }

        private void label3_Click(object sender, EventArgs e)
        {}
////////////////////////////////////////////////////////////////////////////////
```

Код для вибіру мови у ComboBox

```
private void Form13_Load(object sender, EventArgs e)
        {
            comboBox1.DataSource = new System.Globalization.CultureInfo[] {
            System.Globalization.CultureInfo.GetCultureInfo("ru"),
            System.Globalization.CultureInfo.GetCultureInfo("uk"),
            System.Globalization.CultureInfo.GetCultureInfo("en")
            };
            comboBox1.DisplayMember = "NativeName";
            comboBox1.ValueMember = "Name";

            if (!String.IsNullOrEmpty(Properties.Settings.Default.Language))
            {
                comboBox1.SelectedValue = Properties.Settings.Default.Language;
            }

        }
```

Код для збереження мови при закритті додатку

```
Private void Form13_FormClosing(object sender, FormClosingEventArgs e)
        {
            Properties.Settings.Default.Language = comboBox1.SelectedValue.ToString();
            Properties.Settings.Default.Save();
        }
```

Код для закриття програми

```
private void button2_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }


    public partial class Form3 : Form
    {
        public Form3()
        {InitializeComponent();}

        protected override void WndProc(ref Message message)
        {
            if (message.Msg == 0x201)
            {
                base.Capture = false;
                message = Message.Create(base.Handle, 0xA1, new IntPtr(2), IntPtr.Zero);
            }
            base.WndProc(ref message);
        }

        private void button1_Click(object sender, EventArgs e)
        {Form3 ifrm = new Form3(); this.Close();}

        private void Form3_Load(object sender, EventArgs e)
        {}
        private char[] alf;

        void Russian()
        {
alf = new[] { 'а', 'б', 'в', 'г', 'д', 'е', 'ё', 'ж', 'з', 'и', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у',
'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ь', 'ы', 'ъ', 'э', 'ю', 'я' };}

        void English()
        {
 alf = new[] { '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' };}
```

```
        void Ukrainian()
        {
alf = new[] { 'а', 'б', 'в', 'г', 'ґ', 'д', 'е', 'є', 'ж', 'з', 'и', 'і', 'ї', 'й', 'к', 'л', 'м', 'н', 'о', 'п', 'р',
'с', 'т', 'у', 'ф', 'х', 'ц', 'ч', 'ш', 'щ', 'ь', 'ю', 'я' };}
        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {
            RadioButton radioButton1 = (RadioButton)sender;
            if (radioButton1.Checked)
            {
                Russian();
            }
        }

         void radioButton2_CheckedChanged(object sender, EventArgs e)
        {
            RadioButton radioButton2 = (RadioButton)sender;
            if (radioButton2.Checked)
            {
              English();
            }
        }

        private void radioButton3_CheckedChanged(object sender, EventArgs e)
        {
            RadioButton radioButton3 = (RadioButton)sender;
            if (radioButton3.Checked)
            {
                Ukrainian();
            }
        }

        private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
        {
                char number = e.KeyChar;
            if (!Char.IsDigit(number) && number != 8)
            {
                e.Handled = true;
            }
        }
```

```
private void textBox2_TextChanged(object sender, EventArgs e)
{}

private void textBox2_KeyPress(object sender, KeyPressEventArgs e)
{
   char number = e.KeyChar;
   if (!(e.KeyChar <= 47 || e.KeyChar >= 58))
   {
      e.Handled = true;
   }
}

private void textBox3_KeyPress(object sender, KeyPressEventArgs e)
{
   char number = e.KeyChar;
   e.Handled = true;

}

private void textBox1_TextChanged(object sender, EventArgs e)
{}

private void textBox3_TextChanged(object sender, EventArgs e)
{}

private void button2_Click(object sender, EventArgs e)
{

         string text = textBox2.Text;
         var array_input = text.ToCharArray();
   try
   {
      int key = Convert.ToInt32(textBox1.Text);
      char[] array_output = array_input;


   if (alf == null)
   {
```

```
    textBox2.Text = MyStrings.Error9.ToString();
}
else
{

    while (key >= alf.Length)
    {
        key = key - alf.Length;
    }

    for (int i = 0; i < array_input.Length; i++)
    {
        for (int j = 0; j < alf.Length; j++)
        {
            if (array_input[i] == alf[j])
            {
                if (j + key >= alf.Length)
                {
                    int num = (j + key) - alf.Length;
                    array_output[i] = alf[num];
                    break;
                }
                else if (j + key < alf.Length)
                {
                    array_output[i] = alf[j + key];
                    break;
                }
            }
        }
    }
    string str = new string(array_output);
    textBox3.Text = str.ToString();

}

}
catch
{
    textBox1.Text = MyStrings.Error11.ToString();
```

```
        }

    }
    private void textBox3_ReadOnlyChanged(object sender, EventArgs e)
    {

    }

    private void label8_Click(object sender, EventArgs e)
    {

    }
  }
}
```

Код для дешифрування

```
private void button2_Click(object sender, EventArgs e)
    {

        try
        {
            string text = textBox2.Text;
            var array_input = text.ToCharArray();
            int key = Convert.ToInt32(textBox1.Text);
            char[] array_output = array_input;


            if (alf == null)
            {
                textBox1.Text = MyStrings.Error9.ToString();
            }
            else
            {
                while (key >= alf.Length)
                {
                    key = key - alf.Length;
                }

                for (int i = 0; i < array_input.Length; i++)
                {
```

```
                for (int j = 0; j < alf.Length; j++)
                {
                    if (array_input[i] == alf[j])
                    {
                        if (j - key >= alf.Length)
                        {
                            int num = (j - key) - alf.Length;
                            array_output[i] = alf[num];
                            break;
                        }
                        else if (j - key < alf.Length && j - key >= 0)
                        {
                            array_output[i] = alf[j - key];
                            break;
                        }
                        else if (j - key <= 0)
                        {
                            array_output[i] = alf[alf.Length + (j - key)];
                            break;
                        }
                    }
                }
            }
            string str = new string(array_output);
            textBox3.Text = str.ToString();
        }
    }
    catch
    {
        textBox1.Text = MyStrings.Error11.ToString();
    }


using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
```

```
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace EncryptApp_v._0._0._7
{
   public partial class Form6 : Form
   {
      public char[] characters;
      public Form6()
      {
         InitializeComponent();
      }

      private int N; //довжина алфавіту
      private void button1_Click(object sender, EventArgs e)
      {
         this.Close();
      }
////////////////////////////////////////////////////////////////////////////
      private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
      {
         char number = e.KeyChar;
         e.Handled = true;
      }
////////////////////////////////////////////////////////////////////////////

      private void button2_Click(object sender, EventArgs e)
      {
         if (radioButton1.Checked)
         {
            textBox1.Clear();
            if (characters == null)
            {
               textBox1.Text = MyStrings.Error9.ToString();
            }
            else
```

```
            {
                N = characters.Length;
                string s;

                StreamReader sr = new StreamReader("in.txt");
                StreamWriter sw = new StreamWriter("out1.txt");

                while (!sr.EndOfStream)
                {
                    s = sr.ReadLine();


sw.WriteLine(Encode(s, Generate_Pseudorandom_KeyWord(s.Length, 100)));
                }
                Process.Start("out1.txt");
                sr.Close();
                sw.Close();
            }
        }
        else
        {
            textBox1.Clear();
            if (characters == null)
            {
                textBox1.Text = MyStrings.Error9.ToString();
            }
            else
            {
                N = characters.Length;
                if (textBox2.Text.Length > 0)
                {
                    string s;

                    StreamReader sr = new StreamReader("in.txt");
                    StreamWriter sw = new StreamWriter("out1.txt");

                    while (!sr.EndOfStream)
                    {
                        s = sr.ReadLine();
```

```
                    sw.WriteLine(Encode(s, textBox2.Text));
                }
                Process.Start("out1.txt");
                sr.Close();
                sw.Close();
            }
            else
            {
                textBox1.Text = MyStrings.Error6.ToString();
            }
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        if (radioButton1.Checked)
        {
            textBox1.Clear();
            if (characters == null)
            {
                textBox1.Text = MyStrings.Error9.ToString();
            }
            else
            {
                N = characters.Length;
                string s;
                StreamReader sr = new StreamReader("out1.txt");
                StreamWriter sw = new StreamWriter("out2.txt");

                while (!sr.EndOfStream)
                {
                    s = sr.ReadLine();
                    sw.WriteLine(Decode(s,
Generate_Pseudorandom_KeyWord(s.Length, 100)));
                }
                Process.Start("out2.txt");
                sr.Close();
                sw.Close();
```

```
        }
    }
    else if (radioButton2.Checked)
    {
        textBox1.Clear();
        if (characters == null)
        {
            textBox1.Text = MyStrings.Error9.ToString();
        }
        else
        {
            N = characters.Length;
            if (textBox2.Text.Length > 0)
            {
                string s;

                StreamReader sr = new StreamReader("out1.txt");
                StreamWriter sw = new StreamWriter("out2.txt");

                while (!sr.EndOfStream)
                {
               s = sr.ReadLine();
                sw.WriteLine(Decode(s, textBox2.Text));
                }
                Process.Start("out2.txt");
                sr.Close();
                sw.Close();
            }
            else
            {
                textBox1.Text = MyStrings.Error6.ToString();
            }
        }
    }
}
private string Encode(string input, string keyword)
{
    input = input.ToUpper();
    keyword = keyword.ToUpper();
```

```
            string result = "";
            int keyword_index = 0;
            foreach (char symbol in input)
            {
                int c = (Array.IndexOf(characters, symbol) +
                Array.IndexOf(characters, keyword[keyword_index])) % N;
                result += characters[c];
                    keyword_index++;
                if ((keyword_index + 1) == keyword.Length)
                    keyword_index = 0;
            }
            return result;
        }
        //розшифрувати
        private string Decode(string input, string keyword)
        {
            input = input.ToUpper();
            keyword = keyword.ToUpper();
            string result = "";
            int keyword_index = 0;
            foreach (char symbol in input)
            {
                int p = (Array.IndexOf(characters, symbol) + N -
                    Array.IndexOf(characters, keyword[keyword_index])) % N;
                result += characters[p];
                keyword_index++;
                if ((keyword_index + 1) == keyword.Length)
                    keyword_index = 0;
            }
            return result;
        }
private string Generate_Pseudorandom_KeyWord(int lenght, int startSeed)
        {
        Random rand = new Random(startSeed);
            string result = "";
        for (int i = 0; i < lenght; i++)
        result += characters[rand.Next(0, characters.Length)];
          return result;
        }
```

```
//////////////////////////////////////////////////////////////////////////////////
        private void radioButton5_CheckedChanged(object sender, EventArgs e)
        {
            RadioButton radioButton1 = (RadioButton)sender;
            if (radioButton1.Checked)
            {
                characters = new[] { 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И','Й', 'К', 'Л', 'М',
'Н', 'О', 'П', 'Р', 'С','Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ь', 'Ы', 'Ъ','Э', 'Ю', 'Я', ' ', '1', '2',
'3', '4', '5', '6', '7','8', '9', '0' };
            }
        }

        private void radioButton4_CheckedChanged(object sender, EventArgs e)
        {
            RadioButton radioButton1 = (RadioButton)sender;
            if (radioButton1.Checked)
            {
                characters = new[] { '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', ' ', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0' };
            }
        }

        private void radioButton3_CheckedChanged_1(object sender, EventArgs e)
        {
            RadioButton radioButton1 = (RadioButton)sender;
            if (radioButton1.Checked)
            {characters = new[] { 'А', 'Б', 'В', 'Г', 'ґ', 'Д', 'Е', 'Є', 'Ж', 'З', 'И', 'І', 'Ї','Й', 'К',
'Л', 'М', 'Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш','Щ', 'Ь', 'Ю', 'Я', ' ', '1', '2', '3',
'4', '5', '6', '7', '8', '9', '0' };
            }
        }
//////////////////////////////////////////////////////////////////////////////////////////

    }
}
```

```
public partial class Form7 : Form
{
    Transposition t;
    public Form7()
    {
        InitializeComponent();
        t = new Transposition();
    }

    protected override void WndProc(ref Message message)
    {
        if (message.Msg == 0x201)
        {
            base.Capture = false;
            message = Message.Create(base.Handle, 0xA1, new IntPtr(2), IntPtr.Zero);
        }
        base.WndProc(ref message);
    }
    private void button2_Click(object sender, EventArgs e)
    {
        t.SetKey(textBox1.Text);
        if (radioButton1.Checked) textBox3.Text = t.Encrypt(textBox2.Text);
        else textBox3.Text = t.Decrypt(textBox2.Text);}
    private void button1_Click(object sender, EventArgs e)
    {Form7 ifrm = new Form7();this.Close();}
    private void textBox1_TextChanged(object sender, EventArgs e){}
    private void textBox1_KeyPress(object sender, KeyPressEventArgs e){}}}
```

Файл Transposition.cs

```
class Transposition
{
    private int[] key = null;
    public void SetKey(int[] _key)
    {key = new int[_key.Length];
for (int i = 0; i < _key.Length; i++)
key[i] = _key[i];}

public void SetKey(string[] _key){
```

```
key = new int[_key.Length];
try{for (int i = 0; i < _key.Length; i++)
key[i] = Convert.ToInt32(_key[i]);}
catch{}}
        public void SetKey(string _key){}
        public string Encrypt(string input)
        {for (int i = 0; i < input.Length % key.Length; i++)
input += input[i];
string result = "";
            for (int i = 0; i < input.Length; i += key.Length)
            {
               char[] transposition = new char[key.Length];
               try
               {for (int j = 0; j < key.Length; j++)
               transposition[key[j] - 1] = input[i + j];
for (int j = 0; j < key.Length; j++)
 result += transposition[j];}

 catch {}}
return result;}

        public string Decrypt(string input)
        {
        string result = "";
            try
            {
               for (int i = 0; i < input.Length; i += key.Length)
               {
                  char[] transposition = new char[key.Length];
                  for (int j = 0; j < key.Length; j++)
                     transposition[j] = input[i + key[j] - 1];
                  for (int j = 0; j < key.Length; j++)
                     result += transposition[j];
               }
            }
            catch {}
return result;
        }
    }
```

```
public partial class Form8 : Form
{
    public Form8()
    {
        InitializeComponent();
    }

    protected override void WndProc(ref Message message)
    {
        if (message.Msg == 0x201)
        {
            base.Capture = false;
            message = Message.Create(base.Handle, 0xA1, new IntPtr(2), IntPtr.Zero);
        }
        base.WndProc(ref message);
    }
    public string GetHash(string input)
    {
        var md5 = MD5.Create();
        var hash = md5.ComputeHash(Encoding.UTF8.GetBytes(input));

        return Convert.ToBase64String(hash);
    }
    private void Form8_Load(object sender, EventArgs e)
    {

    }

    private void button2_Click(object sender, EventArgs e)
    {
        textBox1.Text = GetHash(textBox2.Text);
    }

    private void textBox2_TextChanged(object sender, EventArgs e)
    {
```

```
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form3 ifrm = new Form3();
            this.Close();
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }
    }



    public partial class Form9 : Form{
        public Form9()
        {InitializeComponent();}
        protected override void WndProc(ref Message message)
        {if (message.Msg == 0x201){
                base.Capture = false;
                message = Message.Create(base.Handle, 0xA1, new IntPtr(2),
IntPtr.Zero);}
            base.WndProc(ref message);}
        private void Form9_Load(object sender, EventArgs e){}
        private void button1_Click(object sender, EventArgs e)
        {Form9 ifrm = new Form9(); this.Close();}
        private void button2_Click(object sender, EventArgs e)
        {string qrtext = textBox2.Text; //считуємо текст з TextBox'a
QRCodeEncoder encoder = new QRCodeEncoder(); //створюємо об'єкт класу
QRCodeEncoder
 Bitmap qrcode = encoder.Encode(qrtext); // кодуємо слово, отримане з TextBox
(qrtext) в змінну qrcode. класу Bitmap(клас, який використовується для роботи з
зображенням)
pictureBox1.Image = qrcode as Image; // pictureBox виводе qrcode як
зображення.}
```

```
private void button3_Click(object sender, EventArgs e){SaveFileDialog save = new
SaveFileDialog(); // save буде апитувати у користувача, де він хоче зберігти
файл.
save.Filter = "PNG|*.png|JPEG|*.jpg|GIF|*.gif|BMP|*.bmp"; //створюємо фільтр,
який визначає, в яких форматах ми можемо зберігти нашу інформацію.
if (save.ShowDialog() == System.Windows.Forms.DialogResult.OK) //якщо
користувач натискає  кнопку
"Зберігти".{pictureBox1.Image.Save(save.FileName); }}
        private void button4_Click(object sender, EventArgs e)
        {OpenFileDialog load = new OpenFileDialog();
           if (load.ShowDialog() == System.Windows.Forms.DialogResult.OK)
           {pictureBox1.ImageLocation = load.FileName; }}
        private void button5_Click(object sender, EventArgs e)
        {QRCodeDecoder decoder = new QRCodeDecoder();
    try{string mass = decoder.decode(new QRCodeBitmapImage(pictureBox1.Image
as Bitmap));
 textBox1.Text = mass.ToString();}
catch {textBox1.Text = MyStrings.Error12.ToString();}}
private void textBox1_TextChanged(object sender, EventArgs e){}}}



   public partial class Form10 : Form
   {char[] characters = new char[] { '#', '1', '2', '3', '4', '5', '6', '7', '8', '9', '0', '-' };
 public Form10(){InitializeComponent();}
 private void Form10_Load(object sender, EventArgs e){}
 protected override void WndProc(ref Message message){
 if (message.Msg == 0x201){base.Capture = false;
 message = Message.Create(base.Handle, 0xA1, new IntPtr(2), IntPtr.Zero);}
base.WndProc(ref message);}
        //перевірка: просте або ні число?
        private bool IsTheNumberSimple(long n)
        {if (n < 2) return false; if (n == 2) return true; for (long i = 2; i < n; i++)
  if (n % i == 0) return false; return true;}
        //зашифрувати
  private List<string> RSA_Endoce(string s, long e, long n)
 {List<string> result = new List<string>(); BigInteger bi;
 for (int i = 0; i < s.Length; i++){int index = Array.IndexOf(characters, s[i]);
 bi = new BigInteger(index); bi = BigInteger.Pow(bi, (int)e);
 BigInteger n_ = new BigInteger((int)n); bi = bi % n_;
```

```
 result.Add(bi.ToString());} return result;}
        //розшифрувати
private string RSA_Dedoce(List<string> input, long d, long n)
{string result = ""; try{BigInteger bi;foreach (string item in input){
bi = new BigInteger(Convert.ToDouble(item));
bi = BigInteger.Pow(bi, (int)d); BigInteger n_ = new BigInteger((int)n); bi = bi % n_;
int index = Convert.ToInt32(bi.ToString()); result += characters[index].ToString();}}
catch{textBox7.Text = MyStrings.Error1.ToString();}return result;}
        //розраховуємо d. d поинно бути взаємно простим с m
private long Calculate_d(long m){long d = m - 1;for (long i = 2; i <= m; i++)
if ((m % i == 0) && (d % i == 0)) //якщо мають загальні дільники
 {d--;i = 1;}return d;
}
//розрахунок параметру e
private long Calculate_e(long d, long m){long e = 10;
while (true){if ((e * d) % m == 1)break;elsee++;}return e;}
private void button4_Click(object sender, EventArgs e)
{if ((textBox1.Text.Length > 0) && (textBox5.Text.Length > 0) &&
(textBox2.Text.Length > 0) && (textBox3.Text.Length > 0)){long p =
Convert.ToInt64(textBox1.Text);long q = Convert.ToInt64(textBox5.Text);if
(IsTheNumberSimple(p) && IsTheNumberSimple(q))
{string hash = File.ReadAllText(textBox2.Text).GetHashCode().ToString();long n =
p * q;long m = (p - 1) * (q - 1);long d = Calculate_d(m);long e_ = Calculate_e(d, m);
List<string> result = RSA_Endoce(hash, e_, n);
StreamWriter sw = new StreamWriter(textBox3.Text);foreach (string item in
result)sw.WriteLine(item);sw.Close();textBox6.Text = d.ToString();textBox4.Text =
n.ToString();Process.Start(textBox3.Text);}else{textBox7.Text =
MyStrings.Error2.ToString();}}else {textBox7.Text =
MyStrings.Error3.ToString();}}
private void button2_Click(object sender, EventArgs e)
{OpenFileDialog ofd = new OpenFileDialog();
ofd.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
if (ofd.ShowDialog() == DialogResult.OK){textBox2.Text = ofd.FileName;}}
private void button3_Click(object sender, EventArgs e){
OpenFileDialog ofd = new OpenFileDialog();
ofd.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
if (ofd.ShowDialog() == DialogResult.OK){textBox3.Text = ofd.FileName;}}
private void button5_Click(object sender, EventArgs e){
```

```
if ((textBox6.Text.Length > 0) && (textBox4.Text.Length > 0) &&
(textBox2.Text.Length > 0) && (textBox3.Text.Length > 0)){long d =
Convert.ToInt64(textBox6.Text);long n =
Convert.ToInt64(textBox4.Text);List<string> input = new List<string>()
StreamReader sr = new StreamReader(textBox3.Text);while (!sr.EndOfStream)
{input.Add(sr.ReadLine());}sr.Close();string result = RSA_Dedoce(input, d, n);
string hash = File.ReadAllText(textBox2.Text).GetHashCode().ToString();if
(result.Equals(hash))
{textBox7.Text = MyStrings.Message1.ToString();}else
{textBox7.Text = MyStrings.Message2.ToString();}}
 else{textBox7.Text = MyStrings.Error4.ToString();}
      private void button1_Click(object sender, EventArgs e)
      {Form10 ifrm = new Form10();this.Close();}
      private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
      {
         char number = e.KeyChar;
         if (!Char.IsDigit(number) && number != 8)
         {e.Handled = true;}}
      private void textBox5_KeyPress(object sender, KeyPressEventArgs e)
  {char number = e.KeyChar;
  if (!Char.IsDigit(number) && number != 8)
 {e.Handled = true;}}
      private void textBox6_KeyPress(object sender, KeyPressEventArgs e)
   {char number = e.KeyChar;
   if (!Char.IsDigit(number) && number != 8){e.Handled = true;}}
     private void textBox4_KeyPress(object sender, KeyPressEventArgs e)
      {char number = e.KeyChar;
         if (!Char.IsDigit(number) && number != 8)
         {e.Handled = true;}}}
```

```
public partial class Form11 : Form
{public Form11(){InitializeComponent();}
private const int sizeOfBlock = 128; //в DES розмір блоку 64 біт, але так як в
unicode символ у два рази довше, то збільшимо блок теж  у два рази
private const int sizeOfChar = 16; //розмір одного символа (in Unicode 16 bit)
private const int shiftKey = 2; //зсув ключа
private const int quantityOfRounds = 16; //кількість раундів
```

```
string[] Blocks; // блоки у двійковому форматі
private void Form11_Load(object sender, EventArgs e){}
private string StringToRightLength(string input){
while (((input.Length * sizeOfChar) % sizeOfBlock) != 0)input += "#";return input;}
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]);}}
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);}
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;}
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;}
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)));}
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);}
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;}
 private string f(string s1, string s2){return XOR(s1, s2);}
 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;}
private string KeyToPrevRound(string key){
for (int i = 0; i < shiftKey; i++){ key = key + key[0]; key = key.Remove(0, 1);}
return key;}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 button2_Click(object sender,
EventArgs e){if (textBox2.Text.Length > 0){string s = "";string key = textBox2.Text;
 StreamReader sr = new StreamReader("in.txt");тwhile (!sr.EndOfStream)
 {s += sr.ReadLine();}sr.Close();s = StringToRightLength(s);
 CutStringIntoBlocks(s);key = CorrectKeyWord(key, s.Length / (2 *
Blocks.Length)); textBox2.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);
            textBox3.Text = StringFromBinaryToNormalFormat(key);
            string result = "";
            for (int i = 0; i < Blocks.Length; i++)
               result += Blocks[i];
            try
            {
               StreamWriter sw = new StreamWriter("out1.txt");
               sw.WriteLine(StringFromBinaryToNormalFormat(result));
               sw.Close();

               Process.Start("out1.txt");
            }
            catch
            {
               textBox3.Text = MyStrings.Error5.ToString();
            }
         }
         else
         {
            textBox3.Text = MyStrings.Error6.ToString();
```

```
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        if (textBox3.Text.Length > 0)
        {
            string s = "";
            string key = StringToBinaryFormat(textBox3.Text);
            StreamReader sr = new StreamReader("out1.txt");
            while (!sr.EndOfStream)
            {
                s += sr.ReadLine();
            }
            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);
textBox2.Text = StringFromBinaryToNormalFormat(key);
string result = "";for (int i = 0; i < Blocks.Length; i++)
result += Blocks[i]; StreamWriter sw = new StreamWriter("out2.txt");
sw.WriteLine(StringFromBinaryToNormalFormat(result));
sw.Close();Process.Start("out2.txt");}else {textBox3.Text =
MyStrings.Error6.ToString();}}
private void button1_Click(object sender, EventArgs e){this.Close();}}



  public partial class Form12 : Form
  {
    public char[] characters;
    public Form12()
    {
      InitializeComponent();
    }
```

```
        private void button4_Click(object sender, EventArgs e)
        {
            if ((textBox1.Text.Length > 0) && (textBox5.Text.Length > 0))
            {
                long p = Convert.ToInt64(textBox1.Text);
                long q = Convert.ToInt64(textBox5.Text);
                if (IsTheNumberSimple(p) && IsTheNumberSimple(q))
                {
                    string s = "";
                    StreamReader sr = new StreamReader("in.txt");
                    while (!sr.EndOfStream)
                    {
                        s += sr.ReadLine();
                    }
                    sr.Close();
                    s = s.ToUpper();
                    long n = p * q;
                    long m = (p - 1) * (q - 1);
                    long d = Calculate_d(m);
                    long e_ = Calculate_e(d, m);
                    List<string> result = RSA_Endoce(s, e_, n);
                    StreamWriter sw = new StreamWriter("out1.txt");
                    foreach (string item in result)
                        sw.WriteLine(item);
                    sw.Close();
                    textBox6.Text = d.ToString();
                    textBox4.Text = n.ToString();
                    Process.Start("out1.txt");
                }else {textBox2.Text = MyStrings.Error2.ToString();}}
else{textBox2.Text = MyStrings.Error7.ToString();}}
        private void button2_Click(object sender, EventArgs e)
        {if ((textBox6.Text.Length > 0) && (textBox4.Text.Length > 0))
            {long d = Convert.ToInt64(textBox6.Text);
                long n = Convert.ToInt64(textBox4.Text);
                List<string> input = new List<string>();
                StreamReader sr = new StreamReader("out1.txt");
                while (!sr.EndOfStream)
                {input.Add(sr.ReadLine());}
                sr.Close();
```

```
            string result = RSA_Dedoce(input, d, n);
            StreamWriter sw = new StreamWriter("out2.txt");
            sw.WriteLine(result);
            sw.Close();
Process.Start("out2.txt");}else{textBox2.Text = MyStrings.Error8.ToString();}}
        private bool IsTheNumberSimple(long n)
        {
            if (n < 2)
                return false;
            if (n == 2)
                return true;
            for (long i = 2; i < n; i++)
                if (n % i == 0)
                    return false;
            return true;
        }

        private List<string> RSA_Endoce(string s, long e, long n)
        {
            List<string> result = new List<string>();
            BigInteger bi;
            try
            {
                textBox2.Clear();
                for (int i = 0; i < s.Length; i++)
                {
                    int index = Array.IndexOf(characters, s[i]);
                    bi = new BigInteger(index);
                    bi = BigInteger.Pow(bi, (int)e);
                    BigInteger n_ = new BigInteger((int)n);
                    bi = bi % n_;
                    result.Add(bi.ToString());}}catch
{textBox2.Text = MyStrings.Error9.ToString();}return result;}
        private string RSA_Dedoce(List<string> input, long d, long n)
        {string result = "";
            BigInteger bi;
            try
            {textBox2.Clear();
                foreach (string item in input)
```

```
                {
bi = new BigInteger(Convert.ToDouble(item));bi = BigInteger.Pow(bi,
(int)d);BigInteger n_ = new BigInteger((int)n);bi = bi % n_;int index =
Convert.ToInt32(bi.ToString());
result += characters[index].ToString();}}catch {textBox2.Text =
MyStrings.Error10.ToString();}return result;private long Calculate_d(long m)
{long d = m - 1;
for (long i = 2; i <= m; i++)
if ((m % i == 0) && (d % i == 0)) //якщо мають загальні дільники
{d--;i = 1;}return d;}private long Calculate_e(long d, long m){long e = 10;while
(true){if ((e * d) % m == 1)break;elsee++;}return e;}
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{RadioButton radioButton1 = (RadioButton)sender;
if (radioButton1.Checked)
{characters = new[] { '#', 'А', 'Б', 'В', 'Г', 'Д', 'Е', 'Ё', 'Ж', 'З', 'И', 'Й', 'К', 'Л', 'М', 'Н',
'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ь', 'Ы', 'Ъ', 'Э', 'Ю', 'Я', ' ', '1', '2','3',
'4', '5', '6', '7','8', '9', '0' };}}private void radioButton2_CheckedChanged(object
sender, EventArgs e){RadioButton radioButton2 = (RadioButton)sender;if
(radioButton2.Checked){characters = new[] { '#', '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', ' ', '1', '2', '3', '4', '5',
'6', '7', '8', '9', '0'};}}
 private void radioButton3_CheckedChanged(object sender, EventArgs e)
{RadioButton radioButton3 = (RadioButton)sender;
if (radioButton3.Checked)
{characters = new[] {  '#', 'А', 'Б', 'В', 'Г', 'ґ', 'Д', 'Е', 'Є', 'Ж', 'З', 'И', 'І', 'Ї','Й', 'К', 'Л',
'М', 'Н', 'О', 'П', 'Р', 'С', 'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ь', 'Ю', 'Я', ' ', '1', '2', '3', '4',
'5', '6', '7', '8', '9', '0' };}}
private void button1_Click(object sender, EventArgs e)
{this.Close();}
protected override void WndProc(ref Message message)
{if (message.Msg == 0x201)
{base.Capture = false;message = Message.Create(base.Handle, 0xA1, new IntPtr(2),
IntPtr.Zero);}base.WndProc(ref message);}}
```

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
«Розробка програмного забезпечення для шифрування та дешифрування текстових повідомлень»
студента групи 121-18ск-1 Тулуши Максима Олексійовича

Керівник економічного розділу
Зав. каф. ПЕП та ПУ, д.е.н. **О. Г. Вагонова**

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Тулуша.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Тулуша.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
Diplom Тулуша.zip	Архів. Містить коди програми і откомпільовану програму.
Презентація	
Презентація Тулуша.ppt	Презентація кваліфікаційної роботи.