

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Кафедра програмних засобів

КУРСОВИЙ ПРОЕКТ

з дисципліни: «Об'єктно-орієнтоване програмування»
на тему: «Візуалізація криптографічних алгоритмів для Blockchain технології. RSA (Rivest–Shamir–Adleman)»

Студентів 1 курсу КНТ-219сп групи
спеціальності Комп'ютерні науки
освітня програма (спеціалізація)

Комп'ютерні науки

Ковальов Р.В.,

Резніченко А.С.,

Рогова Є. В.,

Тихоновська Т.В.

Керівник доцент Миронова Н.О.

Національна шкала _____

Кількість балів: _____ Оцінка: ECTS _____

Члени комісії

(підпис)

(підпис)

(підпис)

Табунщик Г.В.

(прізвище та ініціали)

Миронова Н.О.

(прізвище та ініціали)

Каплієнко Т.І.

(прізвище та ініціали)

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА
ПОЛІТЕХНІКА»

Кафедра програмних засобів

Дисципліна Об'єктно-орієнтоване програмування

Спеціальність Комп'ютерні Науки

Курс 1 Група КНТ-219сп Семестр I

ЗАВДАННЯ
на курсовий проект (роботу) студентам

Ковальову Роману Віталійовичу, Резніченко Андрію Сергійович, Роговій Єлизаветі Віталіївні, Тихоновській Тетяні Володимирівні

1. Тема проекту (роботи): Візуалізація криптографічних алгоритмів для Blockchain технології. RSA (Rivest–Shamir–Adleman)

2. Термін здачі студентом закінченого проекту (роботи): _____

3. Вихідні дані до проекту: Реалізувати криптографічного алгоритму RSA (Rivest–Shamir–Adleman)

вхідні дані: Зображення формату bmp, png, jpeg, jpg, txt

вихідні дані: програма повинна забезпечувати відображення шифрованої інформації та її збереження в відповідному форматі до вхідного і відображення дешифрованої інформації – її збереження в відповідному форматі до вхідного

4. ЗМІСТ розрахунково-пояснювальної записки (перелік питань, що їх належить розробити): 1 Аналіз предметної області

2 Аналіз програмних засобів

3 Основні рішення з реалізації компонентів системи

4 Посібник програміста

5 Інструкція користувача

Висновки, Додаток А Текст програми,

Додаток Б Інтерфейс програми,

Додаток В Слайди презентації

5. Дата видачі завдання: вересень 2019р

Календарний план

№ пор.	Назва етапів курсового проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1.	Аналіз індивідуального завдання	1-2 тиждень	
2.	Аналіз програмних засобів, що будуть використовуватись в роботі	3-4 тиждень	
3.	Аналіз структур даних, що необхідно використати в курсовій роботі	4-5 тиждень	
4.	Вивчення можливостей програмної реалізації структур даних та інтерфейсу користувача	5-6 тиждень	
5.	Оформлення відповідних пунктів пояснювальної записки	4 тиждень	розділи 1,2 ПЗ
6.	Проміжний контроль	8 тиждень	
7.	Аналіз вимог до апаратних засобів	9 тиждень	
8.	Розробка програмного забезпечення	10-15 тиждень	
9.	Оформлення, відповідних пунктів пояснювальної записки	9-16 тиждень	розділи 3,4,5 ПЗ
10.	Захист курсової роботи	17 тиждень	

Студент _____ Ковальов Р.В.

Студент _____ Резніченко А.С.

Студентка _____ Рогова Є.В.

Студентка _____ Тихоновська Т.В.

Керівник _____ Миронова Н.О.

“ _____ ” _____ 2019р

ЗМІСТ

Реферат.....	6
Перелік умовних позначень, символів, одиниць, скорочень і термінів	7
Вступ	8
1 Аналіз предметної області	9
1.1 Огляд існуючих методів вирішення завдання	9
1.2 Огляд існуючих програмних засобів	10
1.2.1 Алгоритм шифрування RSA (Rivest–Shamir–Adleman)	11
1.3 Постановка завдання роботи	13
2 Аналіз програмних засобів	14
2.1 Огляд особливостей мови програмування	14
2.2 Огляд особливостей обраного середовища програмування	14
2.3 Основні компоненти Windows Forms	15
2.4 Використані класи мови C#	16
2.5 Висновки з розділу	20
3 Основні рішення з реалізації компонентів системи	21
3.1 Основні рішення щодо уявлення даних системи.....	21
3.2 Основні розроблені алгоритми.....	24
3.4 Особливості реалізації системи.....	25
3.5 Результати тестування системи.....	26
3.6 Висновки з розділу	26
4 Посібник програміста.....	28
4.1 Призначення та умови застосування програми	28
4.2 Характеристики програми	28
4.3 Звертання до програми.....	29
4.4 Вхідні та вихідні дані	29
4.5 Повідомлення	30
5 Інструкція користувача	31
5.1 Призначення програми.....	31
5.2 Умови виконання програми.....	31

5.3 Як запустити програму.....	31
5.4 Виконання програми	32
5.5 Повідомлення користувачу.....	38
Висновки.....	40
Перелік посилань	41
Додаток А	42
Додаток Б.....	77
Додаток В.....	81

РЕФЕРАТ

ПЗ: 89 стор., 22 рис., 48 додатки, 10 джерел.

Метою даного курсового проекту є розробка візуалізації криптографічних алгоритмів для Blockchain технології. RSA (Rivest–Shamir–Adleman).

Було виконано аналіз предметної області, розглянуто аналогічні існуючі методи та програмні засоби для вирішення завдання.

Для реалізації програмного продукту використовувалася мова програмування C# та середа розробки JetBrains Rider.

При виконанні курсового проекту було розглянуто особливості мови програмування C# та компілятору. Також приділяється увага розробці інтерфейсу користувача, який було розроблено за допомогою Windows Forms. Також приділяється акцент розробці графічного інтерфейсу користувача, файловому вводу-виводу.

RSA, BLOCKCHAIN, LSB, КРИПТОГРАФІЯ, СОРТУВАННЯ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Blockchain	Ланцюжок з формованих блоків транзакцій, який побудований за певними правилами.
RSA	Криптографічний алгоритм з відкритим ключем, що базується на обчислювальній складності задачі факторизації великих цілих чисел.
LSB	Суть цього методу полягає в заміні останніх значущих бітів в контейнері (зображення, аудіо або відеозапису) на біти приховуваного повідомлення.
Криптографія	Наука про математичні методи забезпечення конфіденційності, цілісності і автентичності інформації.
ООП	Об'єктно-орієнтоване програмування - одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою

ВСТУП

Технологія Blockchain часто асоціюється у спільноті з «Інтернетом цінностей». З її допомогою будь-яка людина здатна розмістити в мережі дані, після чого інші користувачі зможуть отримати до неї простий доступ, де б вони не знаходилися. Щоб отримати доступ до розподіленої бази даних, людина повинна володіти закритим ключем, який створений на основі криптографічних алгоритмів. Такий ключ відкриє доступ тільки до тих блоків, які «належать» вам. А якщо ви передаватимете іншій особі закритий ключ, то делегуєте таким чином доступ до даних цих блоків.

Завдяки цьому алгоритму встановлюються довірчі відносини і реалізується принцип автентичності особистості. Нікому не можна змінити ланцюжок блоків без володіння потрібними ключами.

Таким чином, актуальність роботи полягає в практичному підході до роботи із текстом і зображенням, де введена інформація зчитується і додається в масив, що шифрується за відповідним ключем і дешифрується за другим ключем відповідним першому повертаючи початкову інформацію.

Об'єктом дослідження є процес шифрування і дешифрування за створеними ключами.

Метою даного курсового проекту є розробка програмного забезпечення візуалізації для криптографічних алгоритмів Blockchain технології.

Досягнення поставленої мети вимагає розв'язання таких завдань:

- дослідити та розглянути аналоги реалізацій алгоритму RSA;
- обрати метод роботи з зображенням;
- організувати відображення результатів роботи алгоритму і можливість їх збереження.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих методів вирішення завдання

Прикладна програма (застосунок) - користувацька комп'ютерна програма, що дає змогу вирішувати конкретні прикладні задачі користувача.

Види прикладних програм:

- застосунки підприємств та організацій. Наприклад, фінансове управління, система взаємовідносин із споживачами, система постачання. До цього типу відносяться також відомче ПЗ підприємств малого бізнесу, а також окремі підрозділи великого підприємства (Керування транспортними витратами, служба ІТ підтримки);
- застосунки інфраструктури підприємства. Забезпечує загальні потреби у підтримці ПЗ підприємств. Це бази даних, сервери електронної пошти, керування мережею та безпекою;
- застосунки інформаційного робітника. Обслуговує потреби індивідуальних користувачів у створенні та керуванні інформацією. Це, як правило, керування часом, ресурсами, документацією, наприклад, текстові редактори, електронні таблиці, програми-клієнти для електронної пошти та блогів, персональні інформаційні системи та медіа редактори;
- застосунки для доступу до змісту. Використовується для доступу до тих чи інших програм або ресурсів без їх редагування (може включати й функцію редагування). Призначено для груп та індивідуальних користувачів цифрового контенту. Це, наприклад, медіаплеєри, веб-браузери, допоміжні браузері та інше;
- освітні застосунки за змістом близько до ПЗ для медіа та розваг, але на відміну від нього має чіткі вимоги по тестуванню знань користувача та відслідковуванню прогресу у вивченні того чи іншого матеріалу. Багато освітніх програм включають функції спільного користування та багатостороннього співробітництва. Способи використання освітніх програмних засобів у навчальному процесі вивчає технопедагогіка;
- імітаційні програми. Використовуються для симуляції фізичних або абстрактних систем у наукових, освітніх або інших цілях;
- інструментальні застосунки у галузі медіа. Забезпечують потреби користувачів, що виробляють друковані або електронні медіа ресурси для інших

Цей програмний застосунок містить зручний, інтуїтивно зрозумілий інтерфейс користувача і окрім можливості шифрувати і дешифрувати інформацію, а й створення цифрового підпису. Користувач отримує на виході громіздкі ключі і їх можливо перенести в десятковий формат, що є ще одною перевагою застосунку. Але користувачу необхідно вводити адресу файлу повністю в відповідне поле, що не зручно при швидкій роботі. Також не має можливості прослідкувати в самій формі за результатом шифрування і дешифрування.

Іншим аналогом є «ImageCryptography»[4]. Інтерфейс програми наведений на рисунку 1.2.

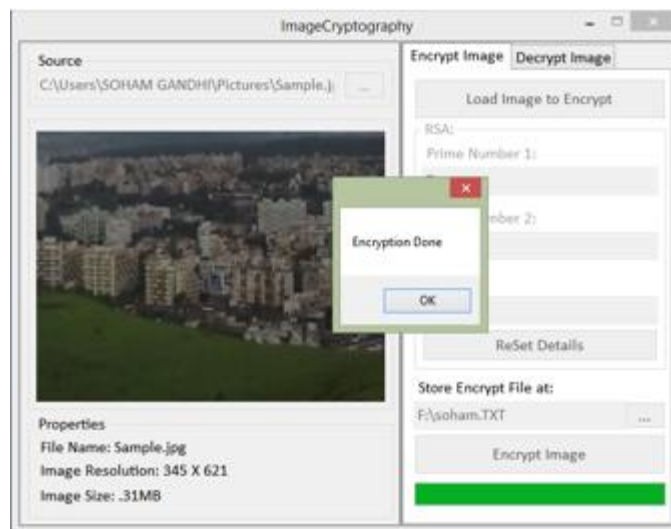


Рисунок 1.2 – Програмний застосунок «ImageCryptography»

В даному прикладі перевагами додатку є можливість побачити початкове зображення та інформацію про нього, також поле прогресу, що дає зрозуміти скільки ще часу триватиме процес шифрування чи дешифрування, і звичайно можливість обрати потрібне зображення за допомогою діалогового вікна. Про те до ряду недоліків додатку належить використання зображень великого розміру, що уповільнює роботу, занесення шифрованого зображення до файлу у вигляді масиву, що не дає змоги візуально перевірити результат роботи алгоритму.

1.2.1 Алгоритм шифрування RSA (Rivest–Shamir–Adleman)

Алгоритм RSA складається з 4 етапів: генерації ключів, шифрування, розшифрування та розповсюдження ключів.

Безпека алгоритму RSA побудована на принципі складності факторизації цілих чисел. Алгоритм використовує два ключі — відкритий (public) і секретний (private), разом відкритий і відповідний йому секретний ключі утворюють пари ключів (keypair). Відкритий ключ не потрібно зберігати в таємниці, він використовується для шифрування даних. Якщо повідомлення було зашифровано відкритим ключем, то розшифрувати його можна тільки відповідним секретним ключем.

Розповсюдження ключів. Для того, щоб Боб міг відправити свої секретні повідомлення, Аліса передає свій відкритий ключ (n, e) Бобу через надійний, але не обов'язково секретний маршрут. Секретний ключ d ніколи не розповсюджується.

Генерація ключів. Для того, щоб згенерувати пари ключів виконуються такі дії:

1. Вибираються два великі прості p і q приблизно 512 біт завдовжки кожне
2. Обчислюється їх добуток $n=pq$
3. Обчислюється функція Ейлера $\phi(n)=(p-1)(q-1)$
4. Вибирається ціле число $1 < e < \phi(n)$ та e взаємно просте з $\phi(n)$
5. За допомогою розширеного алгоритму Евкліда знаходиться d , таке, що $ed=1 \pmod{\phi(n)}$

Число n називається модулем, а e і d — відкритою й секретною експонентами (англ. encryption and decryption exponents), відповідно. Пари чисел (n, e) є відкритою частиною ключа, а (n, d) — секретною. Числа p і q після генерації пари ключів можуть бути знищені, але в жодному разі не повинні бути розкриті.

Шифрування. Припустимо, що Боб хотів би відправити повідомлення M Алісі. Спочатку він перетворює M в ціле число m так, щоб $0 \leq m < n$ за допомогою узгодженого оборотного протоколу, відомого як схеми доповнення. Потім він обчислює зашифрований текст c , використовуючи відкритий ключ Аліси e , за допомогою рівняння:

$$c = m^e \bmod n$$

Це може бути зроблено досить швидко, навіть для 500-бітних чисел, з використанням модульного зведення в ступінь. Потім Боб передає c Алісі.

Розшифрування. Для розшифрування повідомлення Боба m Алісі потрібно обчислити таку рівність:

$$m = c^d \bmod n$$

Неважко переконатися, що при розшифруванні відновиться вихідне повідомлення:

$$C^d = (m^e)^d = m^{ed} \pmod{n}$$

З умови $ed \equiv 1 \pmod{\phi(n)}$ випливає, що $ed = k\phi(n) + 1$ для деякого цілого k , отже $m^{ed} = m^{k\phi(n)+1} \pmod{n}$.

Згідно з теоремою Ейлера: $m^{\phi(n)} \equiv 1 \pmod{n}$ тому:

$$m^{k\phi(n)+1} \equiv m \pmod{n}$$

$$c^d \equiv m \pmod{n}$$

1.3 Постановка завдання роботи

Створити програмний продукт, який шляхом взаємодії з користувачем шифрувати і дешифрувати зображення за створеними ключами.

Система повинна забезпечувати:

- завантаження зображення або тексту;
- зчитати зображення або текст в масив;
- шифрувати елементи масиву;
- вивести шифровану інформацію;
- завантажити шифроване зображення або текст;
- зчитати шифроване зображення або текст в масив;
- дешифрувати елементи масиву;
- вивести дешифровану інформацію.

2 АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ

2.1 Огляд особливостей мови програмування

C# - об'єктно-орієнтована мова програмування. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки. Ось лише кілька функцій мови C #, що забезпечують надійність і стійкість додатків: прибирання сміття автоматично звільняє пам'ять, зайняту знищеними і невикористовуваними об'єктами; обробка виключень надає структурований і розширюваний спосіб виявляти і обробляти помилки; сувора типізація мови не дозволяє звертатися до неініціалізованих змінних, виходити за межі індексованих масивів або виконувати неконтрольоване приведення типів.

Основними перевагами мови є:

- а) багато синтаксичного цукру. Синтаксичний цукор - це такі конструкції, які створені для полегшення написання і розуміння коду (особливо якщо це код іншого програміста) і не грають ролі при компіляції;
- б) гарний розвиток завдяки підтримки Microsoft;
- в) повністю ООП. Об'єктно-орієнтоване програмування - одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою;
- г) величезна кількість вже готових класів на всі випадки життя;
- г) є справжні (а не тільки вкладені) багатовимірні масиви і опціонально - перевірка переповнення.[5]

2.2 Огляд особливостей обраного середовища програмування

JetBrains Rider – кросплатформене інтегроване середовище розробки програмного забезпечення. Функціональність Rider полягає у:

- а) аналіз коду. Rider надає більше 2200 інспекцій коду і автоматизованих виправлень для усунення виявлених проблем як в індивідуальному, так і масовому

порядку. Механізм аналізу помилок по всьому рішенням буде шукати помилки в кодової базі і повідомляти про них, навіть якщо проблемний файл не відкритий в редакторі;

б) редагування коду. Розумний редактор Rider надає різні види автодоповнення і шаблонів, автоматично вставляє парні дужки і імпортує відсутні простору імен. Підказки та іконки на полях допомагають легко переміщатися по ієрархії успадкування, контекстні дії роблять розробку зручною та ефективною;

в) налагодження та інші інструменти. Вбудований відладчик для додатків на .NET Framework, Mono і .NET Core підтримує покрокове виконання, дозволяє обчислювати вирази на льоту, запускати програму від поточної виконуваної рядки до рядка з курсором, відстежувати і змінювати значення змінних. Крім того, Rider включає в себе браузер NuGet, дозволяє переглядати трасування стека, підтримує різні системи контролю версій і бази даних;

г) навігація та пошук. Ви можете миттєво переходити до будь-якого файлу, типу або члену у вашій кодової базі, а також швидко знаходити потрібні налаштування і дії IDE. Від будь-якого символу в коді ви можете миттєво перейти до базових і похідних символів, реалізацій та перевизначення, а також до місць використання.[1]

2.3 Основні компоненти Windows Forms

Для створення програмного застосунку було використано такі компоненти Windows Forms:

- Form – основний клас для роботи з формою.
- PictureBox – елемент, що дозволяє виводити зображення на форму. Він дозволяє відобразити файли у форматі bmp, jpeg, jpg, png, а також метафайли зображень та іконки.
- MessageBox – елемент, що дозволяє показати користувачу повідомлення.
- Label – елемент для відображення тексту на формі, доступного тільки для читання.
- TextBox – елемент для вводу та редагування тексту.
- OpenFileDialog и SaveFileDialog – класи для відкриття та збереження файлів

2.4 Використані класи мови C#

Клас `MemoryStream` безпосередньо розширено з класу `Stream`, це потік (stream), дані якого зберігаються (store) в пам'яті. `Stream` - клас, який імітує потік byte (байт), збудовані в ряд. Клас `Stream` надає базові методи роботи з потоками даних, а саме метод читання та запису байт або масиву байт.[2]

Основні методи класу, що часто застосовуються наведені в таблиці 2.1.

Таблиця 2.1 – Методи класу `MemoryStream`.

Методи класу	Опис
<code>MemoryStream()</code>	Ініціалізує новий екземпляр класу <code>MemoryStream</code> розширюється продуктивністю, ініціалізованої нульовим значенням.
<code>ReadByte()</code>	Зчитує байт з поточного потоку.
<code>ToArray()</code>	Записує вміст потоку в масив байтів незалежно від властивості <code>Position</code> .
<code>ToString()</code>	Повертає рядок, що представляє поточний об'єкт.

Клас використано для розкладу зображення на масив байт і роботи з ним для безпосередньо шифрування зображення. Також він використовується для зворотного процесу – збору зашифрованого масиву в зображення. Відповідна частина коду наведена на рисунку 2.1 та 2.2.


```

var ms = new MemoryStream();
if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Jpeg))
{
    image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
}
if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Png))
{
    image.Save(ms, System.Drawing.Imaging.ImageFormat.Png);
}
if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Bmp))
{
    image.Save(ms, System.Drawing.Imaging.ImageFormat.Bmp);
}
bytes = ms.ToArray();

```

Рисунок 2.1 – Розклад обраного зображення на масив байт

```

byte[] b = mas.Select(i => (byte)i).ToArray();
var imageMemoryStream = new MemoryStream(b);
Image imgFromStream = Image.FromStream(imageMemoryStream);

```

Рисунок 2.2 – Створення зображення з масиву байт

BigInteger структура надає тип BigInteger - це незмінний тип, який представляє довільно велике ціле число. Цей тип дозволяє зберігати цілі числа довільної (будь-який) довжини і вирішувати математичні операції з ними. При зведенні числа в ступінь в даному випадку виходять дуже великі числа, які не поміщаються ні в один зі стандартних типів. Тому для їх зберігання використовується екземпляр класу BigInteger.[2]

Основні методи класу, що часто застосовуються наведені в таблиці 2.2.

Таблиця 2.2 – Методи класу BigInteger.

Методи класу	Опис
1	2
Add(BigInteger, BigInteger)	Складає два значення BigInteger і повертає результат.

Продовження табл. 2.2

1	2
<code>BigInteger(Byte[])</code>	Ініціалізує новий екземпляр структури <code>BigInteger</code> , використовуючи значення в масиві типу <code>byte</code> .
<code>BigInteger(Double)</code>	Ініціалізує новий екземпляр структури <code>BigInteger</code> , використовуючи значення з плаваючою комою подвійної точності.
<code>Pow(BigInteger, Int32)</code>	Зводить значення <code>BigInteger</code> в задану ступінь.
<code>ToArray()</code>	Перетворює значення <code>BigInteger</code> в масив байтів.
<code>ToString()</code>	Перетворює числове значення поточного об'єкта <code>BigInteger</code> в еквівалентну йому строкове представлення.

Клас `List <T>` є універсальним еквівалентом класу `ArrayList`. Він реалізує `List <T>` універсальний інтерфейс за допомогою масиву, розмір якого динамічно збільшується в міру необхідності. Елементи можна додавати в `List <T>` за допомогою методів `Add` або `AddRange`. Являє строго типізований список об'єктів, доступних за індексом. Підтримує методи для пошуку за списком, виконання сортування і інших операцій зі списками.

Основні методи класу, що часто застосовуються наведені в таблиці 2.3.

Таблиця 2.3 – Методи класу `List <T>`.

Методи класу	Опис
<code>List<T>()</code>	Ініціалізує новий екземпляр класу <code>List <T></code> , який порожній і початкову ємність за замовчуванням.
<code>Add(T)</code>	Додає об'єкт в кінець черги <code>List <T></code> .
<code>ToArray()</code>	Копіює елементи списку <code>List <T></code> в новий масив.
<code>ToString()</code>	Повертає рядок, що представляє поточний об'єкт.

Клас Image представляє елемент керування, що відображає зображення. Клас Image дозволяє завантажувати наступні типи зображень :. bmp,. gif,. ico,. jpg,. png. При відображенні багатокадрового зображення відображається тільки перший кадр. Анімація багатокадрових зображень не підтримуються елементом управління Image. До завантаження вмісту зображення ActualWidth і ActualHeight елемента управління матимуть нульове значення, так як вміст зображення використовується для визначення кінцевого розміру і розташування елемента керування. Для елемента керування фіксованого розміру можна задати властивості Width і (або) Height.

Основні методи класу, що часто застосовуються наведені в таблиці 2.4.

Таблиця 2.4 – Методи класу Image.

Методи класу	Опис
Equals(Object)	Визначає, чи є зазначений DependencyObject еквівалентом поточного DependencyObject.
Save(String, ImageFormat)	Зберігає об'єкт Image в зазначений файл у вказаному форматі.
Save(String)	Зберігає об'єкт Image в зазначений файл або потік.
RawFormat	Повертає формат файлу цього об'єкта Image.
FromStream	Створює об'єкт Image із зазначеного потоку даних.

Клас OpenFileDialog показує діалогове вікно, що дозволяє користувачеві відкрити файл. Цей клас не успадковується. Клас SaveFileDialog Запитує у користувача розташування для збереження файлу. Цей клас не успадковується.

Основні методи класів, що часто застосовуються наведені в таблиці 2.5.

Таблиця 2.5 – Методи класу OpenFileDialog та SaveFileDialog.

Методи класу	Опис
OpenFileDialog()	Ініціалізує екземпляр класу OpenFileDialog.
SaveFileDialog()	Ініціалізує новий екземпляр цього класу.
Filter	Повертає або задає поточний рядок фільтра імен файлів, яка визначає варіанти, доступні в поле діалогового вікна "Зберегти як файл типу" або "Файли типу".
ShowDialog()	Запускає загальне діалогове вікно з заданим за замовчуванням власником.

2.5 Висновки з розділу

У даному розділі було описано особливості мови програмування, середі розробки та обраного компілятора. Також розглянули аналоги існуючих програм для алгоритму RSA. Було розглянуто основні компоненти Windows Forms, які були використані для розробки програмного застосунку.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Основні рішення щодо уявлення даних системи

В ході роботи над програмою курсового проекту було розроблено класи: Form1, ПроАлгоритм, ПроПрограму, Інструкція, Про_авторів, EncodeDecode, RSAkey, Program, ImageDate.

Клас Form1 є основним та містить в собі дані головної форми програми, забезпечує взаємодію з користувачем. Дані та методи класу наведені в табл. 3.1.

Таблиця 3.1 – Дані та методи класу Form1.

Поля та методи класу	Опис
1	2
public:	
public Form1()	Завантаження вмісту призначеного для користувача інтерфейсу.
private:	
private void Зберегти_Click(object sender, EventArgs e)	Кнопка збереження шифрованої інформації.
private void Дешифрування_Click(object sender, EventArgs e)	Кнопка дешифрування інформації.
private void ЗберегтиДешифр_Click(object sender, EventArgs e)	Кнопка збереження дешифрованої інформації.
private void ГенеруванняКлючів_Click(object sender, EventArgs e)	Кнопка генерування ключів.
private void ШифруватиІнф_Click(object sender, EventArgs e)	Кнопка шифрування інформації.
private void ЗавантажитиФайл_Click(object sender, EventArgs e)	Кнопка завантаження файлу.

Продовження табл. 3.1

1	2
private void ШифрФайл_Click(object sender, EventArgs e)	Кнопка для завантаження файлу, що містить шифрований текст .
private void Ключі_Click(object sender, EventArgs e)	Кнопка меню «Ключі».
private void Шифрувати_Click(object sender, EventArgs e)	Кнопка меню «Шифрувати».
private void Дешифрувати_Click(object sender, EventArgs e)	Кнопка меню «Дешифрувати».
private void ЗавантажитиЗображення_Click(object sender, EventArgs e)	Кнопка для завантаження зображення і створення масиву байт з нього.
private void ШифрЗоб_Click(object sender, EventArgs e)	Кнопка для завантаження зашифрованого масиву та додаткової інформації до нього.

Класи ПроАлгоритм, ПроПрограму, Інструкція, Про_авторів призначені для надання користувачу більш детальної інформації про програму в окремій формі для користувача. Функції класів наведені в табл. 3.2.

Таблиця 3.2 – Функції класів ПроАлгоритм, ПроПрограму, Інструкція, Про_авторів.

Функції класів	Опис
public Інструкція()	Відображення інформації стосовно призначення проекту.
public ПроАлгоритм()	Відображення інформації про алгорит, що використовується.
public ПроПрограму()	Відображення інструкції для користувача.
public Про_авторів()	Відображення інформації стосовно авторів проекту.

Клас EncodeDecode призначений для реалізації функцій шифрування і дешифрування даних методом RSA. Функції класів наведені в табл. 3.3.

Таблиця 3.3 – Функції класу EncodeDecode.

Функції класів	Опис
public List<string> RSA_Endoce(string s, long e, long n)	Функція призначена для шифрування обраних даних.
public string RSA_Dedoce(List<string> input, long d, long n)	Функція призначена для дешифрування обраних даних.

Клас RSAkey призначений для створення допоміжних змінних, що необхідні для позначення ключів методом RSA. Функції класів наведені в табл. 3.4.

Таблиця 3.4 – Функції класу RSAkey.

Функції класів	Опис
private bool IsTheNumberSimple(long n)	Функція перевірка чи є число простим.
private long Calculate_e()	Функція визначення параметру e.
private long Calculate_d(long d, long m)	Функція визначення параметру d.

Клас ImageDate призначений для роботи з функціями, що необхідні для перетворення зображення в масив байт і навпаки. Функції класів наведені в табл. 3.5.

Таблиця 3.5 – Функції класу ImageDate.

Функції класів	Опис
public byte[] MasByte(Image image)	Функція призначена для перетворення зображення в масив байт.
public Image RetIm(int[] mas)	Функція призначена для перетворення числового масиву в зображення.

Клас Program, що має метод саме з нього програма починає свою роботу, там же ви і побачите відкриття вікна: Application.Run (new Form1 ()) – що запускає першу форму.

3.2 Основні розроблені алгоритми

Під час розробки було реалізовано наступні алгоритми:

- файлове зчитування та запис у файл;
- відображення візуальних та текстових елементів;
- завантаження зображення або тексту;
- розбиття зображення на байтовий масив та переведення цього масиву в інший формат;
- застосування алгоритму криптографічного шифрування RSA (Rivest–Shamir–Adleman).

Для розбиття файлу на байтовий масив використовується клас C# - MemoryStream, що перетворює зображення потік байт і тим самим допомагає записати його до байтового масиву. Далі байтовий масив для зручності конвертується в масив цілих чисел. З нього обирається двадцять довільних значень. Це зроблено для прискорення роботи алгоритму, та можливості зберегти зашифроване зображення. Після чого, випадково обрані значення передаються у викликану функцію шифрування у вигляді рядку.

Додавання нових зображень у програму або тексту. При роботі алгоритму шифрування враховано можливість не заповнених полів, що є обов'язковими, тому тоді коли вони є вільними з'являється відповідне діалогове вікно, що повідомляє які поля мають бути заповненими для подальшої роботи.

Алгоритм RSA розбито на 5 функцій, що виконують певний крок його роботи.

Крок перший - для створення ключів вводяться два простих числа, на основі яких генеруватимуться ключі. Одна з функцій перевіряє введені числа на простоту, тобто число має ділитися на себе та одиницю.

Крок другий - необхідно вирахувати функцію Ейлера, що знаходиться шляхом перемноження введених значень менших на одиницю.

Крок третій - обирається відкрита експонента, що також є простим числом. Для цього створено функцію в якій на основі запропонованого масиву випадково обирається таке число.

Крок четвертий - функція вираховує секретну експоненту за формулою: відкрита експоненти в ступені мінус одиниці перемножується з модулем результату функції Ейлера.

Ключі створено. Відкритий ключ матиме значення обраної відкритої експоненти та добутку введених простих чисел. Закритий ключ матиме значення секретної експоненти та добутку введених простих чисел. Шифрування відбувається за наступною формулою: кожне значення введеної інформації зводиться в ступінь введеної експоненти та перемножується з модулем добутку простих чисел. Дешифрування відбувається за тією самою формулою, проте необхідно вказувати значення другого ключа.

3.4 Особливості реалізації системи

Особливістю реалізації системи є зрозумілий для користувача інтерфейс. Будь-яка людина зможе просто користуватися розробленим додатком на інтуїтивному рівні, завантажувати потрібне йому зображення або текст, перетворювати зображення в масив байт та застосовувати алгоритм шифрування і дешифрування. Також особливістю є шифрування не всього масиву байт створеного з зображення, а обраних навмання байт, що є додатковою інформацією для дешифрування. Відповідну частину коду з довільним обранням позицій масиву байт для шифрування наведено на рис. 3.1.

```
var ran = new Random();
int[] mas = new int[20];
int[] g = new int[20];

for (int i = 0; i < 20; i++)
{
    g[i] = ran.Next(100, mas.Length);
    mas[i] = mas[g[i]];
}

strImg = string.Join(" ", mas);

result = RSA_Endoce(strImg, e_, n);
```

Рисунок 3.1 – Обрання довільного масиву байт для шифрування

3.5 Результати тестування системи

У ході тестування програми усі помилки було виправлено. На даному етапі програма функціонує правильно та злагоджено. На рис. 3.2 зображено повідомлення, яке виникає при не заповнених полях для генерації ключів.

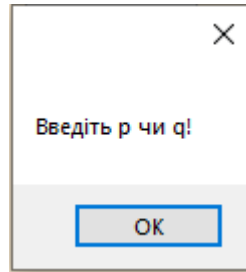


Рисунок 3.2 – Повідомлення не заповнених полів простих чисел

На рис. 3.3 зображено повідомлення, яке виникає при не заповнених полях ключів для шифрування чи дешифрування.

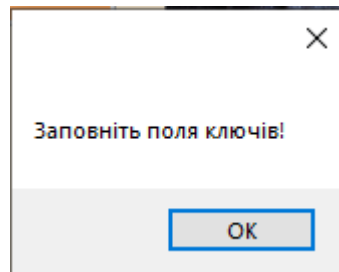


Рисунок 3.3 – Повідомлення не заповнених полів ключів

3.6 Висновки з розділу

В ході роботи над програмою курсового проекту було розроблено клас Form1, та використано клас MemoryStream та структуру BigInteger, клас List <T>, клас Image, класи OpenFileDialog та SaveFileDialog.

При виконанні курсової роботи було розроблено декілька алгоритмів (завантаження зображення, завантаження файлу, відображення візуальних та текстових елементів) які відповідають певним функціональним частинам програми, що допомагає створенню зрозумілого користувацької інтерфейсу. При розробці інтерфейсу програми були

використані такі візуальні компоненти Windows Forms: TextBox, Label, PictureBox, panel, ToolStripMenuItem.

У ході тестування системи усі помилки було виправлено.

4 ПОСІБНИК ПРОГРАМІСТА

4.1 Призначення та умови застосування програми

Призначення програми – забезпечення візуалізації шифрування зображень та тексту, які завантажуються користувачем.

Умовою застосування для коректної роботи є використання останніх версій ПЗ, що необхідно для роботи програми. Для отримання повного функціоналу цілком достатньо обирати зображення розміром не більше 400 КБ, текст може бути довільного розміру, але дія алгоритму буде повільнішою.

4.2 Характеристики програми

Програма виконана за допомогою мови програмування високого рівня C# в середовищі розробки JetBrains Rider. Проект (рис. 4.1) файли класу форми, ресурсів та проекту.

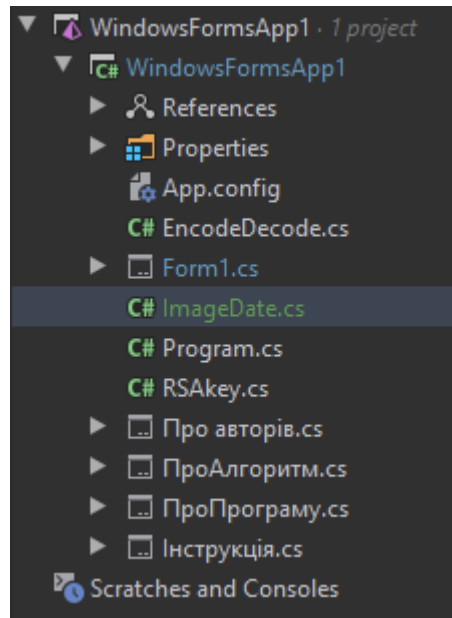
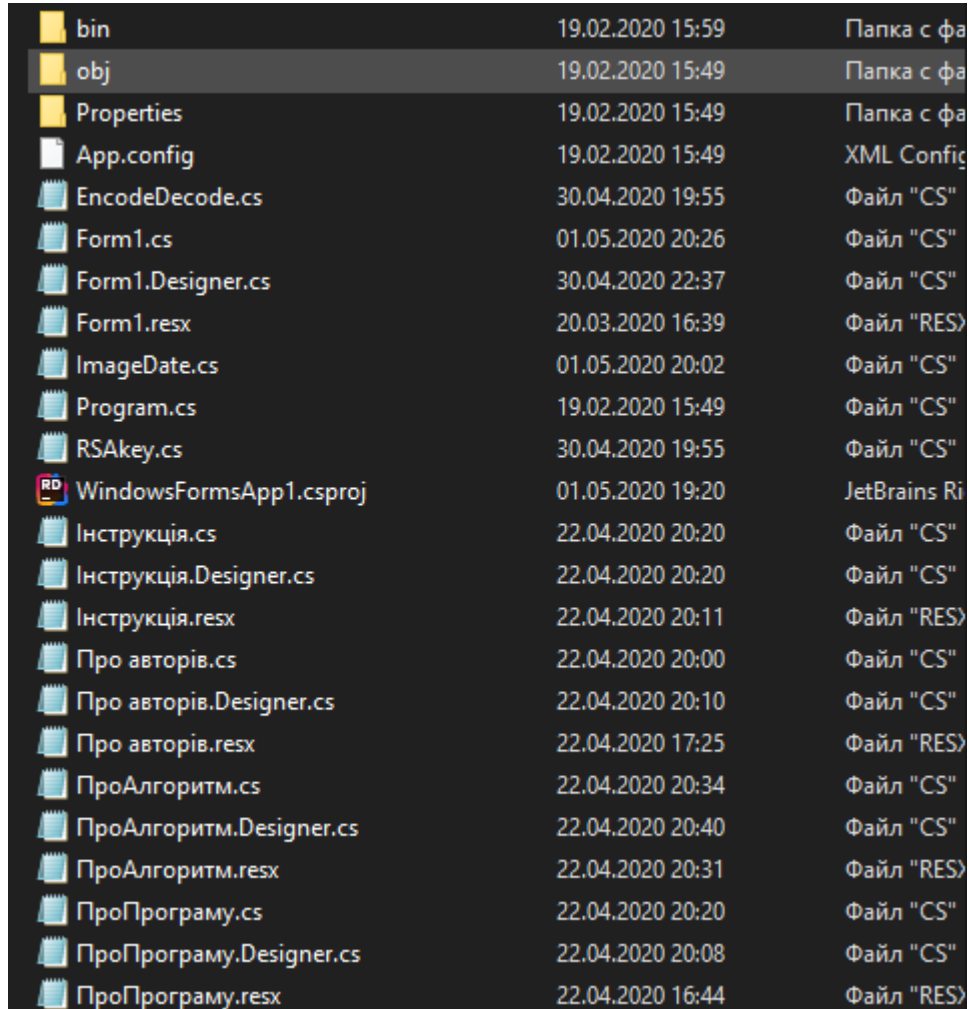


Рисунок 4.1 – Структура проекту

4.3 Звертання до програми

Для звернення необхідно розархівувати папку з програмою та розташувати її у каталог, до якого має доступ оператор. Далі для запуску програми у папці WindowsFormsApp1 (рис.4.2) треба запустити файл WindowsFormsApp1.csproj, дочекатися запуску проекту. Потім натиснути кнопку «Local Windows Debugger» і програма готова до роботи.



bin	19.02.2020 15:59	Папка с фа
obj	19.02.2020 15:49	Папка с фа
Properties	19.02.2020 15:49	Папка с фа
App.config	19.02.2020 15:49	XML Config
EncodeDecode.cs	30.04.2020 19:55	Файл "CS"
Form1.cs	01.05.2020 20:26	Файл "CS"
Form1.Designer.cs	30.04.2020 22:37	Файл "CS"
Form1.resx	20.03.2020 16:39	Файл "RESX"
ImageDate.cs	01.05.2020 20:02	Файл "CS"
Program.cs	19.02.2020 15:49	Файл "CS"
RSAkey.cs	30.04.2020 19:55	Файл "CS"
WindowsFormsApp1.csproj	01.05.2020 19:20	JetBrains Ri
Інструкція.cs	22.04.2020 20:20	Файл "CS"
Інструкція.Designer.cs	22.04.2020 20:20	Файл "CS"
Інструкція.resx	22.04.2020 20:11	Файл "RESX"
Про авторів.cs	22.04.2020 20:00	Файл "CS"
Про авторів.Designer.cs	22.04.2020 20:10	Файл "CS"
Про авторів.resx	22.04.2020 17:25	Файл "RESX"
ПроАлгоритм.cs	22.04.2020 20:34	Файл "CS"
ПроАлгоритм.Designer.cs	22.04.2020 20:40	Файл "CS"
ПроАлгоритм.resx	22.04.2020 20:31	Файл "RESX"
ПроПрограму.cs	22.04.2020 20:20	Файл "CS"
ПроПрограму.Designer.cs	22.04.2020 20:08	Файл "CS"
ПроПрограму.resx	22.04.2020 16:44	Файл "RESX"

Рисунок 4.2 – Папка з файлами програми

4.4 Вхідні та вихідні дані

Вхідними даними виступає зображення або файл з текстом, яке користувач бажає шифрувати, чи вже шифроване зображення або файл з текстом, за наступними форматами: bmp, jpeg, jpg, png, txt. Вихідними даними виступають вже шифроване чи дешифроване

зображення або файл з текстом, за наступними форматами: bmp, jpeg, jpg, png, txt. Також в програмі реалізовано обробка ситуацій коли вхідна чи вихідна інформація є пошкодженою.

4.5 Повідомлення

В процесі роботи програми можуть виникнути помилки в роботі програми. В такому разі оператор отримає інформаційні повідомлення щодо деталей помилки. При роботі може бути отримане таке повідомлення – помилка відкриття файлу, така помилка може виникнути у випадку, якщо обраний користувачем файл пошкоджено. Або може з'явитися повідомлення пов'язане з незаповненими полями, що є до цього обов'язковими (рис. 4.3 та 4.4).

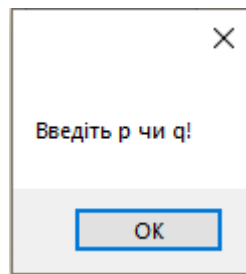


Рисунок 4.3 – Повідомлення не заповнених полів простих чисел

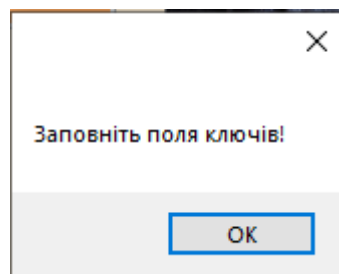


Рисунок 4.4 – Повідомлення не заповнених полів ключів

5 ІНСТРУКЦІЯ КОРИСТУВАЧА

5.1 Призначення програми

Програма призначена для візуалізації криптографічного алгоритму шифрування RSA (Rivest–Shamir–Adleman). Програма працює із зображеннями та текстовими файлами, перетворює зображення в масив байт, шифрує масив байт або текст, після виводить шифровану інформацію. Також дешифрує зображення або текст, виводить результат. Надаємо можливість збереження інформації.

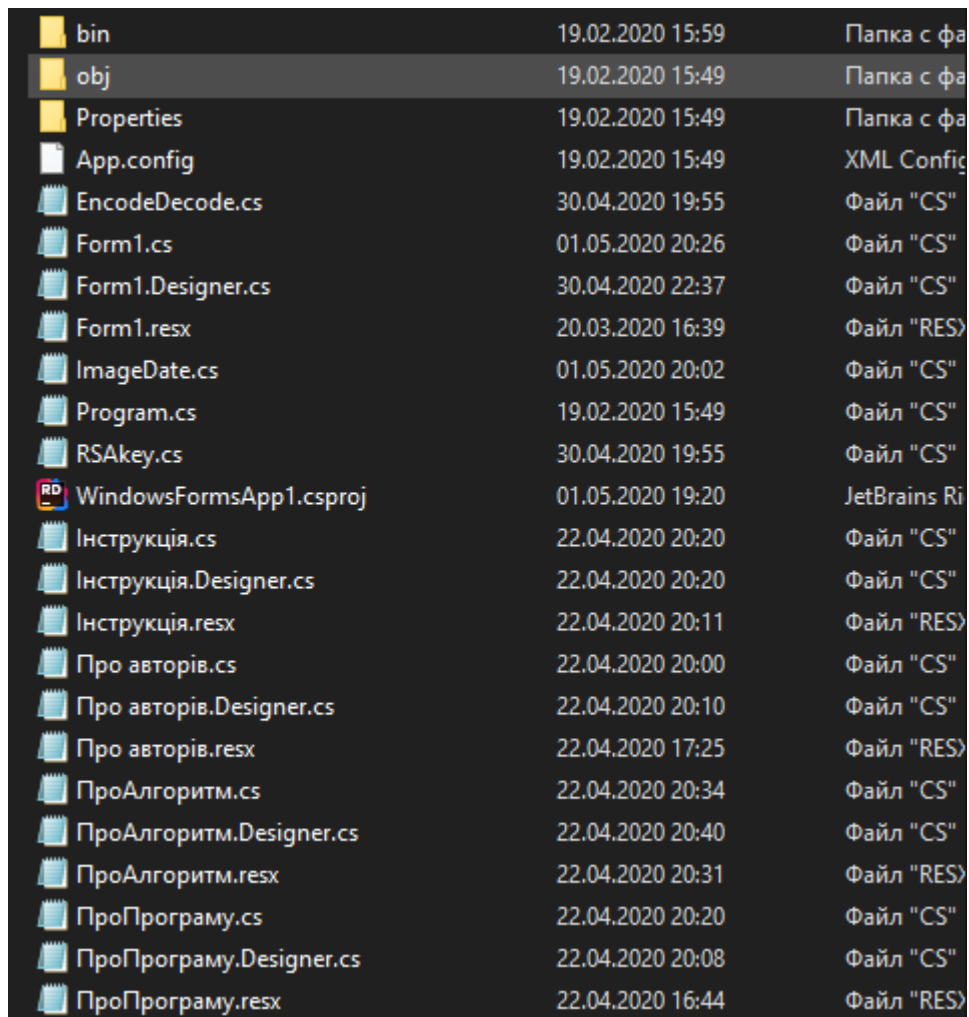
5.2 Умови виконання програми

Програма має коректно працювати за умов:

- запуску останньої версії програми;
- збереження наборів інформації у директорії, в яких є права на читання та запис файлів, що не є заблокованими;
- правильного вводу даних в обов'язкові поля;
- вхідне зображення або текстовий файл не пошкоджені.

5.3 Як запустити програму

Для роботи у програмі перш за все необхідно розархівувати папку з програмою та розташувати її у каталог, до якого ви маєте доступ. Для запуску застосунку у папці WindowsFormsApp1 (рис.5.1) треба запустити файл WindowsFormsApp1.csproj, дочекатися запуску проекту та натиснути кнопку «Local Windows Debugger». Програма готова до роботи.



bin	19.02.2020 15:59	Папка с фа
obj	19.02.2020 15:49	Папка с фа
Properties	19.02.2020 15:49	Папка с фа
App.config	19.02.2020 15:49	XML Config
EncodeDecode.cs	30.04.2020 19:55	Файл "CS"
Form1.cs	01.05.2020 20:26	Файл "CS"
Form1.Designer.cs	30.04.2020 22:37	Файл "CS"
Form1.resx	20.03.2020 16:39	Файл "RESX"
ImageDate.cs	01.05.2020 20:02	Файл "CS"
Program.cs	19.02.2020 15:49	Файл "CS"
RSakey.cs	30.04.2020 19:55	Файл "CS"
WindowsFormsApp1.csproj	01.05.2020 19:20	JetBrains Ri
Інструкція.cs	22.04.2020 20:20	Файл "CS"
Інструкція.Designer.cs	22.04.2020 20:20	Файл "CS"
Інструкція.resx	22.04.2020 20:11	Файл "RESX"
Про авторів.cs	22.04.2020 20:00	Файл "CS"
Про авторів.Designer.cs	22.04.2020 20:10	Файл "CS"
Про авторів.resx	22.04.2020 17:25	Файл "RESX"
ПроАлгоритм.cs	22.04.2020 20:34	Файл "CS"
ПроАлгоритм.Designer.cs	22.04.2020 20:40	Файл "CS"
ПроАлгоритм.resx	22.04.2020 20:31	Файл "RESX"
ПроПрограму.cs	22.04.2020 20:20	Файл "CS"
ПроПрограму.Designer.cs	22.04.2020 20:08	Файл "CS"
ПроПрограму.resx	22.04.2020 16:44	Файл "RESX"

Рисунок 5.1 – Каталог з файлами програми

5.4 Виконання програми

Після налагодження програми перед користувачем з'являється головне вікно. Форма головного вікна є стартовою. Містить у собі кнопки переходу на інші панелі. Головна форма зображена на рис. 5.2.

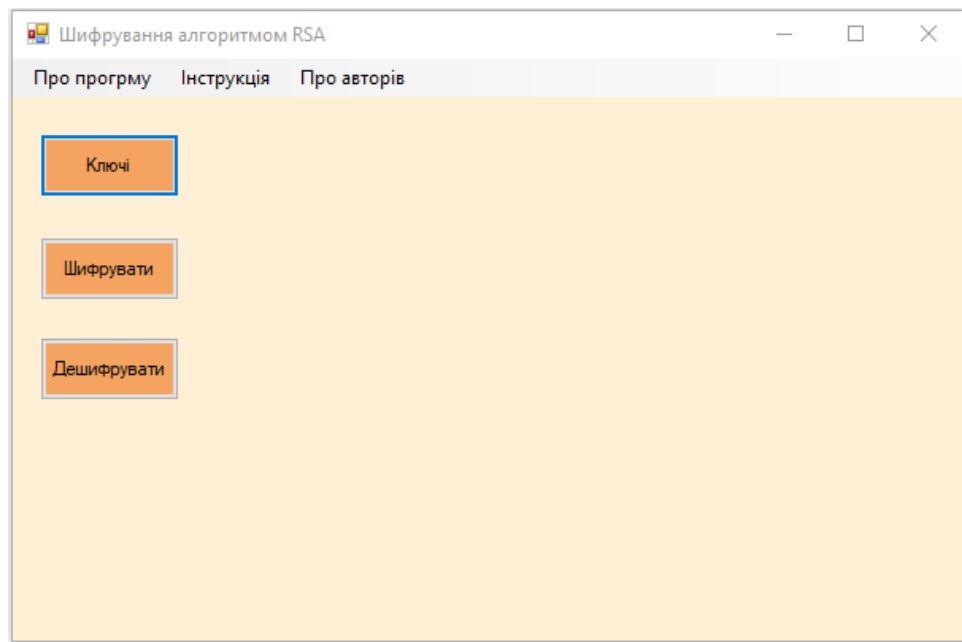


Рисунок 5.2 – Головна форма програми

Після того, як програму було відкрито слід натиснути кнопку «Ключі» і заповнити відповідні поля та натиснути кнопку «Створити пару ключів» після чого програма згенерує ключі відповідно до заповнених полів і збереже їх до файлу, що знаходиться в директорії проекту (рис. 5.3).

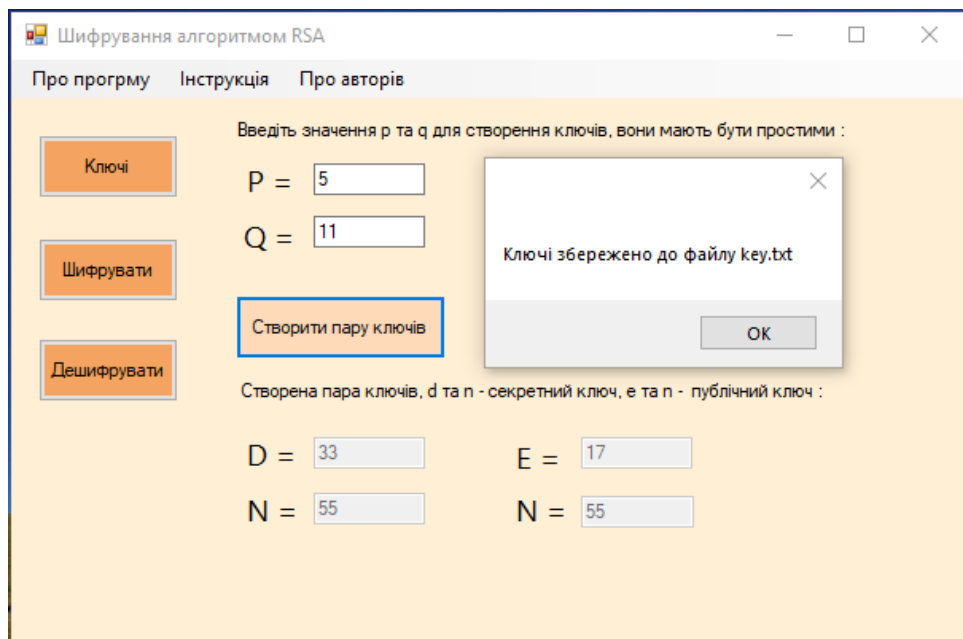


Рисунок 5.3 – Вікно панелі створення ключів

Після того, як буде згенеровано ключі слід натиснути кнопку «Шифрувати» і там обрати файл чи зображення для шифрування, для цього потрібно натиснути на кнопку «Завантажити зображення» чи «Завантажити документ». З'явиться відповідне діалогове вікно (рис. 5.4). Ці кнопки відкривають стандартне вікно відкриття файлів операційної системи Windows, у якому треба обрати зображення або файл відповідного формату та натиснути кнопку «Відкрити». Доступними форматами для завантаження є: bmp, jpeg, jpg, png, txt.

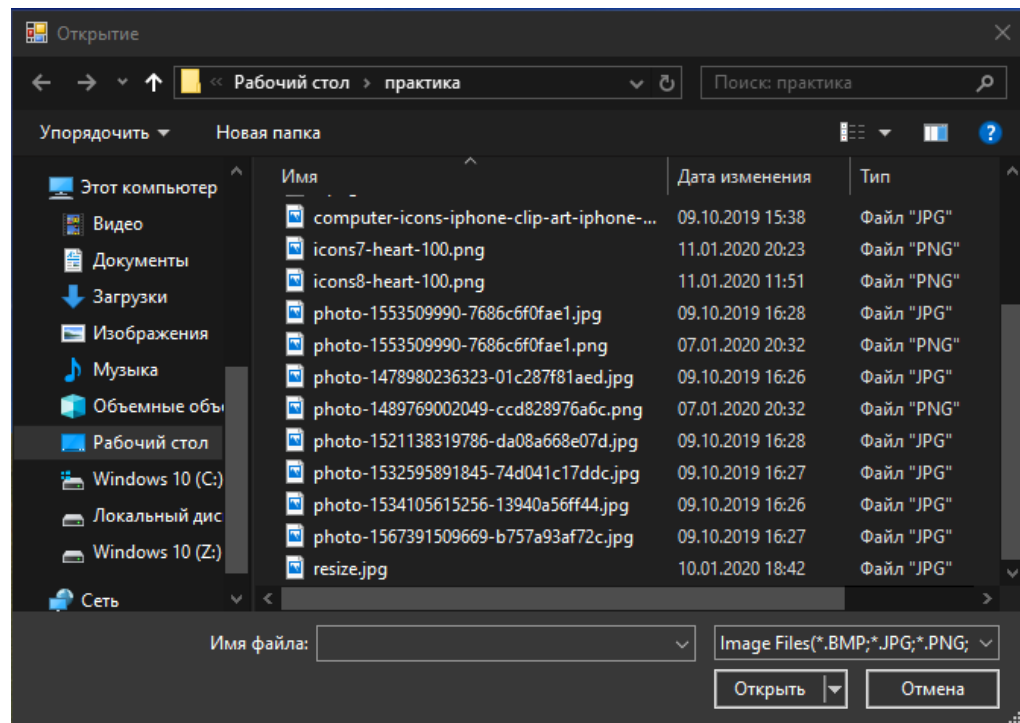


Рисунок 5.4 – Вікно для завантаження інформації

Далі слід заповнити поля ключів і натиснути кнопку «Шифрувати» в результаті чого введену інформацію буде змінено в відповідному її формату полі (рис. 5.5).

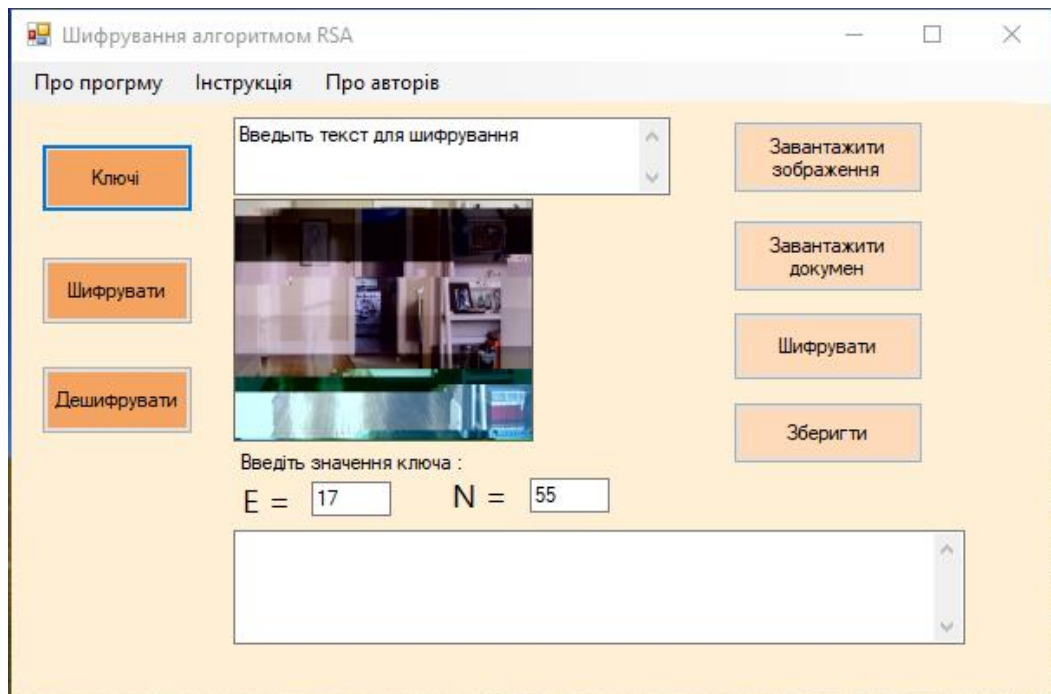


Рисунок 5.5 – Панель шифрування

Коли інформацію було зашифровано її можливо зберегти потрібно натиснути кнопку «Зберегти». З'явиться відповідне діалогове вікно (рис. 5.6). Ця кнопка відкриває стандартне вікно збереження файлу операційної системи Windows, у якому треба обрати місце розташування і ввести нову назву файлу, далі натиснути кнопку «Зберегти». Доступними форматами для збереження є: jpg, txt.

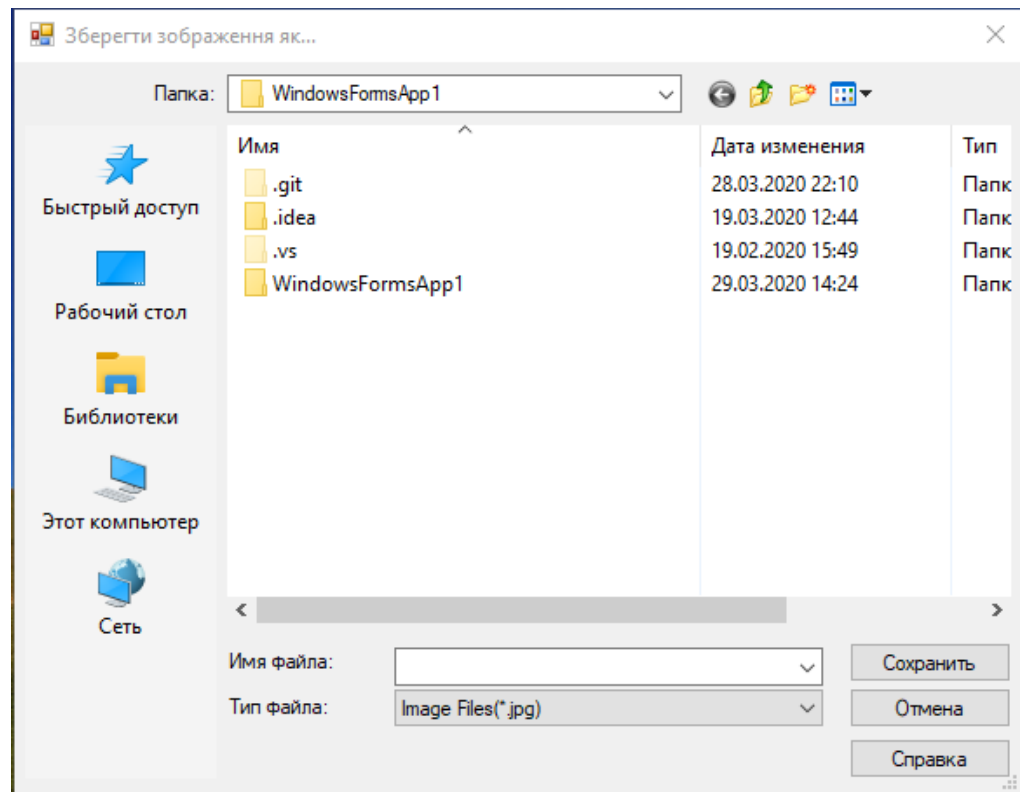


Рисунок 5.6 – Вікно для збереження інформації

Тоді можна дешифрувати нову збережену інформацію натиснувши кнопку «Дешифрувати» і там обрати файл чи зображення для дешифрування, для цього потрібно натиснути на кнопку «Завантажити зображення» чи «Завантажити документ». З'явиться відповідне діалогове вікно (рис. 5.4). Ці кнопки відкривають стандартне вікно відкриття файлів операційної системи Windows, у якому треба обрати зображення або файл відповідного формату та натиснути кнопку «Відкрити». Доступними форматами для завантаження є: bmp, jpeg, jpg, png, txt. Далі слід заповнити поля ключів і натиснути кнопку «Дешифрувати» в результаті чого введену інформацію буде змінено в відповідному її формату полі. Коли інформацію було зашифровано її можливо зберегти потрібно натиснути кнопку «Зберегти». З'явиться відповідне діалогове вікно. Ця кнопка відкриває стандартне вікно збереження файлу операційної системи Windows, у якому треба обрати місце розташування і ввести нову назву файлу, далі натиснути кнопку «Зберегти». Доступними форматами для збереження є: jpg, txt.

При натисканні на кнопку «Про програму» відкривається форма, що містить інформацію стосовно призначення програми (рис. 5.7).

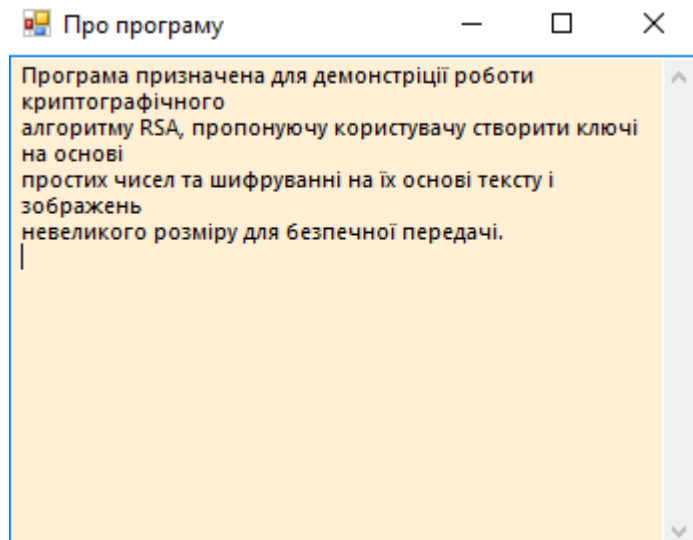


Рисунок 5.7 – Форма «Про програму»

При натисканні на кнопку «Про авторів» відкривається форма, що містить список розробників програми (рис. 5.8).

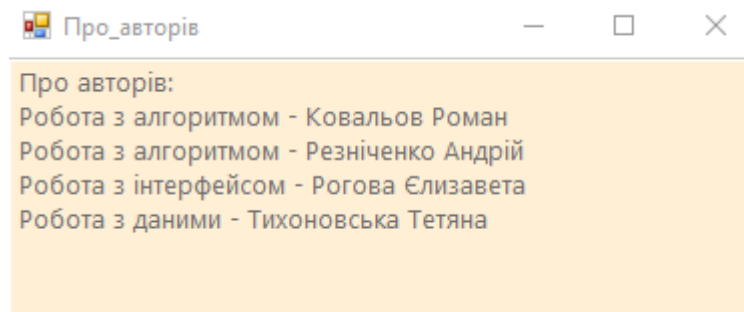


Рисунок 5.8 – Форма «Про авторів»

При натисканні на кнопку «Інструкція» відкривається форма з поясненням, як користуватися програмою (рис. 5.9).

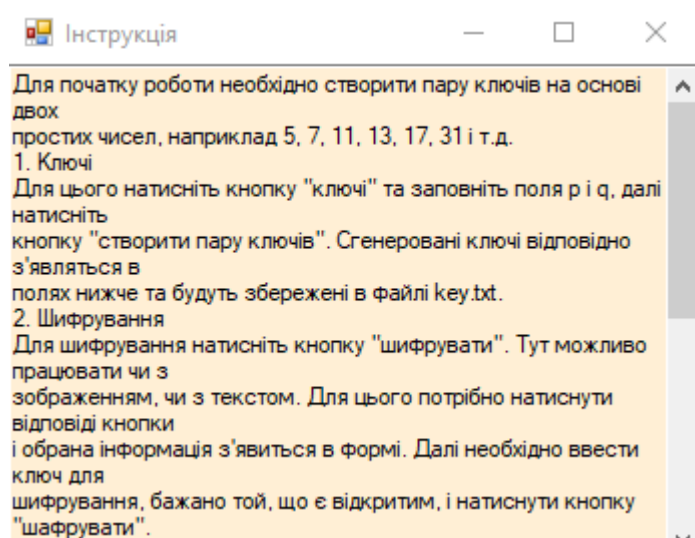


Рисунок 5.9 – Форма «Інструкція»

При натисканні на кнопку «Про алгоритм» відкривається форма з поясненням, який алгоритм використовується і як саме він працює(рис. 5.10).

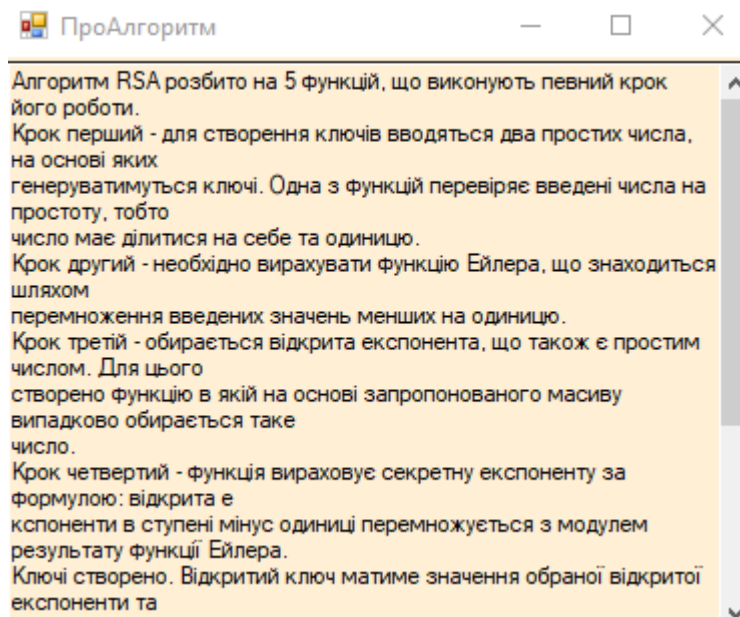


Рисунок 5.10 – Форма «Про алгоритм»

5.5 Повідомлення користувачу

В процесі роботи програми можуть виникнути помилки в роботі програми. В такому разі користувач отримає інформаційні повідомлення щодо деталей помилки. Одне з

можливих повідомлень, що може бути отримане при роботі – така помилка може виникнути у випадку, якщо обраний користувачем файл пошкоджено. Або може з'явитися повідомлення пов'язане з незаповненими полями, що є до цього обов'язковими. В такому випадку користувачу лише необхідно заповнити відповідні поля. Ілюстрація повідомлення приведена на рис. 5.10.

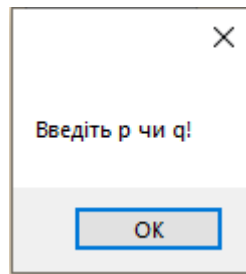


Рисунок 5.10 – Повідомлення не заповнених полів простих чисел

ВИСНОВКИ

Під час виконання курсового проекту було розроблено імітаційну програму на тему «Візуалізація криптографічних алгоритмів для Blockchain технології. RSA (Rivest–Shamir–Adleman)» для роботи з даними для забезпечення відображення алгоритму шифрування і дешифрування.

Створена імітаційна програма організовує систему для забезпечення вибору даних: зображення, текстовий файл. Для шифрування та дешифрування було використано криптографічний алгоритм RSA (Rivest–Shamir–Adleman). Таким чином, розробивши імітаційну програму ми можемо зробити висновок, що алгоритм є ефективним для шифрування даних, проте його час роботи залежить від розміру цих даних. Відповідно, якщо файл є великим тоді і час виконання є довшим.

Також було виконано огляд та аналіз існуючих методів вирішення завдання та існуючих програмних засобів, таких як «Кодировка и декодировка RSA» та «ImageCryptography».

Вирішені наступні завдання курсового проекту:

- виконано огляд сучасних програмних засобів реалізації візуалізації;
- розглянуті можливості та переваги мови програмування C#;
- розглянуті можливості та переваги середовища розробки JetBrains Rider;
- розроблено систему для відображення результатів роботи алгоритму.

ПЕРЕЛІК ПОСИЛАНЬ

1. Rider documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/idea/>
2. .Net documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.microsoft.com/ru-ru/dotnet/api/system.numerics?view=netframework-4.8>
3. «Алгоритм шифрования RSA» Учебная программа на C++ [Електронний ресурс] - Режим доступу до ресурсу: <https://orenstudent.ru/RSA.htm>
4. Image Cryptography using RSA Algorithm in C# [Електронний ресурс] - Режим доступу до ресурсу: <https://www.codeproject.com/Articles/723175/Image-Cryptography-using-RSA-Algorithm-in-Csharp>
5. Костриба О.В. Основи програмування. Частина 1. Visual C# Express Edition [Текст] / О.В. Костриба. – Білогір'я, 2008. – 74 с.
6. Плєскач В.Л., Затонацька Т.Г. Інформаційні системи і технології на підприємствах [Текст] / В.Л. Плєскач, Т.Г. Затонацька. – Київ, 2011. – 718 с.
7. Шнайер Б. Прикладна криптографія. Протоколи та алгоритми [Текст] / Б. Шнайер - М.: Триумф, 2003. - 806 с.
8. Яценко В.В. Вступ у криптографію [Текст] / В.В. Яценко - М.: МЦНМО, 1999. - 272 с.
9. Palzl J. Understanding Cryptography: A Textbook for Students and Practitioners [Текст] / J. Palzl – 2009. – 372 с.
10. Song Y. Cryptanalytic Attacks on RSA [Текст] / Y. Song - New York: Springer, 2008. - 270 с.

ДОДАТОК А

Текст програми

A.1 Текст файлу Form1.cpp

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

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        public static Image image;
        public static int[] mas;
        public byte[] bytes;

        public string s = "";
        public List<string> result = new List<string>();
        public List<string> input = new List<string>();
        public string resalt = "";
```

```

public string strImg = "";
public string stIm = "";
public List<string> sI = new List<string>();
public Image imgFromStream;

public Form1()
{
    InitializeComponent();
}

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

private void Ключі_Click(object sender, EventArgs e)
{
    panel1.Show();
    panel2.Visible = true;
    panel1.Visible = false;
    panel3.Visible = false;
    textBox7.Enabled = false;
    textBox8.Enabled = false;
    textBox9.Enabled = false;
    textBox10.Enabled = false;
}

private void Шифрувати_Click(object sender, EventArgs e)
{

```

```

panel2.Show();
panel1.Visible = true;
panel2.Visible = false;
panel3.Visible = false;
textBox14.Enabled = false;
textBox13.Enabled = false;
textBox2.Enabled = false;
button5.Enabled = false;
button6.Enabled = false;
pictureBox1.Image = null;
pictureBox2.Image = null;
textBox4.Text = "";
textBox3.Text = "";
}

private void Дешифрувати_Click(object sender, EventArgs e)
{
    panel3.Show();
    panel3.Visible = true;
    panel2.Visible = false;
    panel1.Visible = false;
    textBox11.Enabled = false;
    textBox12.Enabled = false;
    textBox3.Enabled = false;
    ШифруватиІнф.Enabled = false;
    Зберигти.Enabled = false;
    pictureBox1.Image = null;
    pictureBox2.Image = null;
    textBox4.Text = "";
    textBox3.Text = "";
}

```

```

private void textBox4_Enter(object sender, EventArgs e)
{
    textBox4.Text = null;
    textBox4.ForeColor = Color.Black;
}

private void Form1_Load(object sender, EventArgs e)
{
    textBox4.Text = "Введить текст для шифрування";//подсказка
    textBox4.ForeColor = Color.Black;
    textBox1.Text = "Введить текст для розшифрування";//подсказка
    textBox1.ForeColor = Color.Black;
}

private void textBox1_Enter(object sender, EventArgs e)
{
    textBox1.Text = null;
    textBox1.ForeColor = Color.Black;
}

private void ЗвантажитиЗображення_Click(object sender, EventArgs e)
{
    OpenFileDialog open_dialog = new OpenFileDialog();
    open_dialog.Filter = "Image
Files(*.BMP;*.JPG;*.PNG;)*.BMP;*.JPG;*.PNG|All files (*.*)|*.*";
    if (open_dialog.ShowDialog() == DialogResult.OK)
    {
        try
        {

```

```

        image = new Bitmap(open_dialog.FileName);
        pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
        pictureBox1.Image = image;
        pictureBox1.Invalidate();
    }
    catch
    {
        DialogResult result = MessageBox.Show("Неможливо відкрити
зображення",
        "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
ImageDate id = new ImageDate();
bytes = id.MasByte(image);
mas = bytes.Select(i =>(int) i).ToArray();

textBox11.Enabled = true;
textBox12.Enabled = true;
ШифруватиІнф.Enabled = true;
ЗавантажитиФайл.Enabled = false;
ЗвантажитиЗображення.Enabled = false;
}

private void ШифрФайл_Click(object sender, EventArgs e)
{
    OpenFileDialog open_dialog = new OpenFileDialog();
    open_dialog.Filter = "Image Files(*.txt)|*.txt|All files (*.*)|*.*";
    if (open_dialog.ShowDialog() == DialogResult.OK)
    {
        try
        {

```

```

        StreamReader sr = new StreamReader(open_dialog.FileName);
        while (!sr.EndOfStream)
        {
            input.Add(sr.ReadLine());
        }
        sr.Close();
    }
    catch
    {
        DialogResult result = MessageBox.Show("Неможливо відкрити
документ",
        "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

textBox1.Clear();
foreach (string item in input)
{
    textBox1.Text += item + " ";
}

textBox13.Enabled = true;
textBox14.Enabled = true;
button5.Enabled = true;
button6.Enabled = false;
}

private void Шифр3об_Click(object sender, EventArgs e)
{
    OpenFileDialog open_dialog = new OpenFileDialog();

```

```

open_dialog.Filter = "Image
Files(*.BMP;*.JPG;*.PNG)|*.BMP;*.JPG;*.PNG|All files (*.*)|*.*";
if (open_dialog.ShowDialog() == DialogResult.OK)
{
    try
    {
        image = new Bitmap(open_dialog.FileName);
        pictureBox2.SizeMode = PictureBoxSizeMode.StretchImage;
        pictureBox2.Image = image;
        pictureBox2.Invalidate();
    }
    catch
    {
        DialogResult result = MessageBox.Show("Неможливо відкрити
зображення",
        "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
ImageDate id = new ImageDate();
bytes = id.MasByte(image);
mas = bytes.Select(i =>(int) i).ToArray();

OpenFileDialog open_dialog1 = new OpenFileDialog();
open_dialog1.Filter = "Image Files(*.txt)|*.txt|All files (*.*)|*.*";
if (open_dialog1.ShowDialog() == DialogResult.OK)
{
    try
    {
        StreamReader sr = new StreamReader(open_dialog1.FileName);
        while (!sr.EndOfStream)
        {

```



```

        sI.Add(sr.ReadLine());
    }

    sr.Close();
}
catch
{
    DialogResult result = MessageBox.Show("Неможливо відкрити
документ",
        "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

textBox13.Enabled = true;
textBox14.Enabled = true;
button5.Enabled = true;
button6.Enabled = false;
}

private void ГенераціяКлючів_Click(object sender, EventArgs e)
{
    if ((textBox5.Text.Length > 0) && (textBox6.Text.Length > 0))
    {
        long p = Convert.ToInt64(textBox5.Text);
        long q = Convert.ToInt64(textBox6.Text);
        RSAkey k = new RSAkey();
        if (k.IsTheNumberSimple(p) && k.IsTheNumberSimple(q))
        {
            Ключі.Enabled = false;
            Дешифрувати.Enabled = false;
            Шифрувати.Enabled = false;
        }
    }
}

```

```
textBox5.Enabled = false;  
textBox6.Enabled = false;
```

```
RSAkey pas = new RSAkey();  
long n = p * q;  
long m = (p - 1) * (q - 1);  
long e_ = pas.Calculate_e();  
long d = pas.Calculate_d(e_, m);
```

```
textBox7.Text = d.ToString();  
textBox8.Text = n.ToString();
```

```
textBox9.Text = e_.ToString();  
textBox10.Text = n.ToString();
```

```
textBox5.Enabled = true;  
textBox6.Enabled = true;  
textBox7.Enabled = false;  
textBox8.Enabled = false;  
textBox9.Enabled = false;  
textBox10.Enabled = false;  
Ключі.Enabled = true;  
Дешифрувати.Enabled = true;  
Шифрувати.Enabled = true;
```

```
}
```

```
else
```

```
    MessageBox.Show("p чи q - не росте число!");
```

```
}
```

```
else
```

```
    MessageBox.Show("Введіть p чи q!");
```

```

List<string> result = new List<string>();

result.Add("Секретний ключ: ");
result.Add(textBox7.Text);
result.Add(textBox8.Text);
result.Add("Публічний ключ: ");
result.Add(textBox9.Text);
result.Add(textBox10.Text);

StreamWriter sw = new
StreamWriter(@"C:\Users\User\Desktop\WindowsFormsApp1\key.txt");
foreach (string item in result)
    sw.WriteLine(item);
sw.Close();
MessageBox.Show("Ключі збережено до файлу key.txt");
}

private void ШифруватиІнф_Click(object sender, EventArgs e)
{
    if (textBox4.Text.Length < 0 && pictureBox1.Image == null)
    {
        MessageBox.Show("Заповніть текстове поле, чи завантажте зображення!");
    }
    else
    {
        if (textBox4.Text.Length > 0)
        {
            if ((textBox11.Text.Length > 0) && (textBox12.Text.Length > 0))
            {
                textBox11.Enabled = false;
            }
        }
    }
}

```

```

textBox12.Enabled = false;
textBox4.Enabled = false;
textBox3.Enabled = false;
ШифруватиІнф.Enabled = false;
ЗвантажитиЗображення.Enabled = false;
ЗавантажитиФайл.Enabled = false;
Зберигти.Enabled = false;

long e_ = Convert.ToInt64(textBox11.Text);
long n = Convert.ToInt64(textBox12.Text);

EncodeDecode ed = new EncodeDecode();
result = ed.RSA_Endoce(s, e_, n);

for (int i = 0; i < result.Count; i++)
    textBox3.Text += result[i] + " ";

textBox11.Enabled = true;
textBox12.Enabled = true;
textBox4.Enabled = true;
textBox3.Enabled = true;
ШифруватиІнф.Enabled = true;
ЗвантажитиЗображення.Enabled = true;
ЗавантажитиФайл.Enabled = true;
Зберигти.Enabled = true;
}
else
    MessageBox.Show("Заповніть поля ключів!");
}
if (pictureBox1.Image != null)

```

```

{
    if ((textBox11.Text.Length > 0) && (textBox12.Text.Length > 0))
    {
        textBox11.Enabled = false;
        textBox12.Enabled = false;
        textBox4.Enabled = false;
        textBox3.Enabled = false;
        ШифруватиІнф.Enabled = false;
        ЗвантажитиЗображення.Enabled = false;
        ЗавантажитиФайл.Enabled = false;
        Зберигти.Enabled = false;

        long e_ = Convert.ToInt64(textBox11.Text);
        long n = Convert.ToInt64(textBox12.Text);

        var ran = new Random();
        int[] mos = new int[20];
        int[] g = new int[20];

        for (int i = 0; i < 20; i++)
        {
            g[i] = ran.Next(100, mas.Length);
            mos[i] = mas[g[i]];
        }

        strImg = string.Join(" ", mos);

        EncodeDecode ed = new EncodeDecode();
        result = ed.RSA_Endoce(strImg, e_, n);

        string som = string.Join(" ", result);
    }
}

```

```

int[] ia = som.Split(' ').Select(w => Convert.ToInt32(w)).ToArray();

for (int i = 0; i < 20; i++)
{
    mas[g[i]] = ia[i];
}

strImg = string.Join(" ", g);
stIm = string.Join(" ", ia);

ImageDate id = new ImageDate();
imgFromStream = id.RetIm(mas);
pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
pictureBox1.Image = imgFromStream;

textBox11.Enabled = true;
textBox12.Enabled = true;
textBox4.Enabled = true;
textBox3.Enabled = true;
ШифруватиІнф.Enabled = true;
ЗвантажитиЗображення.Enabled = true;
ЗавантажитиФайл.Enabled = true;
Зберигти.Enabled = true;
}
else
    MessageBox.Show("Заповніть поля ключів!");

}
}
}

```

```

private void ЗавантажитиФайл_Click(object sender, EventArgs e)
{
    OpenFileDialog open_dialog = new OpenFileDialog();
    open_dialog.Filter = "Image Files (*.txt)|*.txt|All files (*.*)|*.*";
    if (open_dialog.ShowDialog() == DialogResult.OK)
    {
        try
        {
            StreamReader sr = new StreamReader(open_dialog.FileName);
            while (!sr.EndOfStream)
            {
                s += sr.ReadLine();
            }
            sr.Close();
        }
        catch
        {
            DialogResult result = MessageBox.Show("Неможливо відкрити документ",
                "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    textBox4.Text = s;

    textBox11.Enabled = true;
    textBox12.Enabled = true;
    ШифруватиІнф.Enabled = true;
    ЗавантажитиФайл.Enabled = false;

```

```

ЗвантажитиЗображення.Enabled = false;
}

private void Зберегти_Click(object sender, EventArgs e)
{
    if (textBox3.Text.Length < 0 && pictureBox1.Image == null)
    {
        MessageBox.Show("Заповніть текстове поле, чи завантажте зображення!");
    }
    else
    {
        if (textBox3.Text.Length > 0)
        {
            SaveFileDialog savedialog = new SaveFileDialog();
            savedialog.Title = "Зберегти як...";
            savedialog.OverwritePrompt = true;
            savedialog.CheckPathExists = true;
            savedialog.Filter = "Image Files (*.txt)|*.txt|All files (*.*)|*.*";
            savedialog.ShowHelp = true;
            if (savedialog.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    StreamWriter sw = new StreamWriter(savedialog.FileName);
                    foreach (string item in result)
                        sw.WriteLine(item);
                    sw.Close();
                    MessageBox.Show("Зашифрований текст збережено!");
                }
                catch
                {

```



```

        MessageBox.Show("Неможливо зберегти", "Помилка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

if (pictureBox1.Image != null)
{
    SaveFileDialog savedialog = new SaveFileDialog();
    savedialog.Title = "Зберегти зображення як...";
    savedialog.OverwritePrompt = true;
    savedialog.CheckPathExists = true;
    savedialog.Filter = "Image Files (*.jpg)|*.jpg | All files (*.*)|*.*";
    savedialog.ShowHelp = true;
    if (savedialog.ShowDialog() == DialogResult.OK)
    {
        try
        {
            imgFromStream.Save(savedialog.FileName,
System.Drawing.Imaging.ImageFormat.Jpeg);
            MessageBox.Show("Зашифроване зображення збережено!");
        }
        catch
        {
            MessageBox.Show("Неможливо зберегти зображення", "Помилка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    SaveFileDialog savedialog1 = new SaveFileDialog();
    savedialog1.Title = "Зберегти як...";

```

```

savedialog1.OverwritePrompt = true;
savedialog1.CheckPathExists = true;
savedialog1.Filter = "Image Files(*.txt)|*.txt|All files (*.*)|*.*";
savedialog1.ShowHelp = true;
if (savedialog1.ShowDialog() == DialogResult.OK)
{
    try
    {
        StreamWriter sw = new StreamWriter(savedialog1.FileName);
        sw.WriteLine(strImg);
        sw.WriteLine(stIm);
        sw.Close();
        MessageBox.Show("Допоміжну інформацію збережено!");
    }
    catch
    {
        MessageBox.Show("Неможливо зберегти", "Помилка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
}
}

private void Дешифрування_Click(object sender, EventArgs e)
{
    if (textBox4.Text.Length < 0 && pictureBox2.Image == null)
    {
        MessageBox.Show("Заповніть текстове поле, чи завантажте зображення!");
    }
    else

```

```

{
    if (textBox4.Text.Length > 0)
    {
        if ((textBox13.Text.Length > 0) && (textBox14.Text.Length > 0))
        {
            textBox13.Enabled = false;
            textBox14.Enabled = false;
            textBox1.Enabled = false;
            textBox2.Enabled = false;
            button10.Enabled = false;
            button6.Enabled = false;
            button5.Enabled = false;
            button11.Enabled = false;

            long d = Convert.ToInt64(textBox13.Text);
            long n = Convert.ToInt64(textBox14.Text);

            EncodeDecode ed = new EncodeDecode();
            resalt = ed.RSA_Dedoce(input, d, n);

            textBox2.Text = resalt;

            textBox13.Enabled = true;
            textBox14.Enabled = true;
            textBox1.Enabled = true;
            textBox2.Enabled = true;
            button10.Enabled = true;
            button6.Enabled = true;
            button5.Enabled = true;
            button11.Enabled = true;
        }
    }
}

```

```

else
    MessageBox.Show("Заповніть поля ключів!");
}
if (pictureBox2.Image != null)
{
    if ((textBox13.Text.Length > 0) && (textBox14.Text.Length > 0))
    {
        textBox13.Enabled = false;
        textBox14.Enabled = false;
        textBox1.Enabled = false;
        textBox2.Enabled = false;
        button10.Enabled = false;
        button6.Enabled = false;
        button5.Enabled = false;
        button11.Enabled = false;

        long d = Convert.ToInt64(textBox13.Text);
        long n = Convert.ToInt64(textBox14.Text);

        long[] ia = new long[20];
        int[] ar = new int[20];

        ia = sI[0].Split(' ').Select(v => Convert.ToInt64(v)).ToArray();
        int[] kj = sI[1].Split(' ').Select(v => Convert.ToInt32(v)).ToArray();
        ar = ia.Select(i => (int)i).ToArray();
        int[] ms = new int[20];

        for (int i = 0; i < 20; i++)
        {
            ms[i] = mas[ar[i]];
        }
    }
}

```

```

string rlt = string.Join(" ", kj);
List<string> res = rlt.Split(' ').ToList();

EncodeDecode ed = new EncodeDecode();
resalt = ed.RSA_Dedoce(res, d, n);

int[] ai = resalt.Split(' ').Select(k => Convert.ToInt32(k)).ToArray();

for (int i = 0; i < 20; i++)
{
    mas[ar[i]] = ai[i];
}

ImageDate id = new ImageDate();
imgFromStream = id.RetIm(mas);
pictureBox2.SizeMode = PictureBoxSizeMode.StretchImage;
pictureBox2.Image = imgFromStream;

textBox13.Enabled = true;
textBox14.Enabled = true;
textBox1.Enabled = true;
textBox2.Enabled = true;
button10.Enabled = true;
button6.Enabled = true;
button5.Enabled = true;
button11.Enabled = true;
}
else
    MessageBox.Show("Заповніть поля ключів!");
}

```

```

    }
}

```

```

private void ЗберегтиДешифр_Click(object sender, EventArgs e)
{
    if (textBox2.Text.Length < 0 && pictureBox2.Image == null)
    {
        MessageBox.Show("Заповніть текстове поле, чи завантажте зображення!");
    }
    else
    {
        if (textBox2.Text.Length > 0)
        {
            SaveFileDialog savedialog = new SaveFileDialog();
            savedialog.Title = "Зберегти як...";
            savedialog.OverwritePrompt = true;
            savedialog.CheckPathExists = true;
            savedialog.Filter = "Image Files (*.txt)|*.txt|All files (*.*)|*.*";
            savedialog.ShowHelp = true;
            if (savedialog.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    StreamWriter sw = new StreamWriter(savedialog.FileName);
                    sw.WriteLine(resalt);
                    sw.Close();
                }
                catch
                {
                    MessageBox.Show("Неможливо зберегти", "Помилка",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
    }
}

```

```

    }
}
MessageBox.Show("Дешифрований текст збережено!");
}

if (pictureBox2.Image != null)
{
    SaveFileDialog savedialog = new SaveFileDialog();
    savedialog.Title = "Зберегти зображення як...";
    savedialog.OverwritePrompt = true;
    savedialog.CheckPathExists = true;
    savedialog.Filter = "Image Files (*.jpg)|*.jpg | All files (*.*)|*.*";
    savedialog.ShowHelp = true;
    if (savedialog.ShowDialog() == DialogResult.OK)
    {
        try
        {
            imgFromStream.Save(savedialog.FileName,
System.Drawing.Imaging.ImageFormat.Jpeg);
            MessageBox.Show("Зображення збережено!");
        }
        catch
        {
            MessageBox.Show("Неможливо зберегти зображення", "Помилка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
}
}

```

```
private void проАвторівToolStripMenuItem1_Click(object sender, EventArgs e)
{
    Про_авторів newForm = new Про_авторів();
    newForm.Show();
}
```

```
private void Інструкція_Click(object sender, EventArgs e)
{
    Інструкція newForm = new Інструкція();
    newForm.Show();
}
```

```
private void проАвторівToolStripMenuItem_Click(object sender, EventArgs e)
{
    ПроАлгоритм newForm = new ПроАлгоритм();
    newForm.Show();
}
}
```

A.2 Текст файлу Інструкція.cs

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

namespace WindowsFormsApp1
{
    public partial class Інструкція : Form
    {
        public Інструкція()
```



```

    {
        InitializeComponent();

        string t = "";

        StreamReader sr = new
StreamReader(@"C:\Users\User\Desktop\WindowsFormsApp1\Інструкція.txt");

        while (!sr.EndOfStream)
        {
            t = sr.ReadLine();
            textBox1.Text += t + Environment.NewLine;
        }

        sr.Close();
    }
}
}

```

A.3 Текст файлу Про авторів.cs

```

using System;
using System.Windows.Forms;
using System.IO;

namespace WindowsFormsApp1
{
    public partial class Про_авторів : Form
    {
        public Про_авторів()
        {

```

```

InitializeComponent();

string t = "";

StreamReader sr = new
StreamReader(@"C:\Users\User\Desktop\WindowsFormsApp1\про_авторів.txt");

while (!sr.EndOfStream)
{
    t = sr.ReadLine();
    textBox1.Text += t + Environment.NewLine;
}

sr.Close();

textBox1.Enabled = false;
}

}
}

```

A.4 Текст файлу ПроАлгоритм.cs

```

using System;
using System.Windows.Forms;
using System.IO;

namespace WindowsFormsApp1
{
    public partial class ПроАлгоритм : Form
    {
        public ПроАлгоритм()
    }
}

```

```

    {
        InitializeComponent();

        string t = "";

        StreamReader sr = new
StreamReader(@"C:\Users\User\Desktop\WindowsFormsApp1\про_алгоритм.txt");

        while (!sr.EndOfStream)
        {
            t = sr.ReadLine();
            textBox1.Text += t + Environment.NewLine;
        }

        sr.Close();
    }
}

```

A.5 Текст файла ПроПрограму.cs

```

using System;

using System.Windows.Forms;

using System.IO;

namespace WindowsFormsApp1
{
    public partial class ПроПрограму : Form
    {

```

```

public ПроПрограму()
{
    InitializeComponent();

    string t = "";

    StreamReader sr = new
StreamReader(@"C:\Users\User\Desktop\WindowsFormsApp1\про_програму.txt");

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

        textBox1.Text += t + Environment.NewLine;
    }

    sr.Close();
}
}
}

```

A.6 Текст файлу EncodeDecode.cs

```

using System;

using System.Numerics;

using System.Collections.Generic;

```

```

namespace ED

{

    public class EncodeDecode

    {

        char[] characters = new char[] { 'a', ' ', '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', 'ю', 'я', '-', ',',

            '.', 'A', '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', 'Ю', 'Я', '1', '2', '3', '4', '5', '6', '7', '8', '9',

            '0', '!', '?' };

        public 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;
}

public string RSA_Deduce(List<string> input, long d, long n)
{
    string result = "";

    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();
    }

    return result;
}
}
}

```

A.7 Текст файла RSAkey.cs

```

using System;

namespace Key
{
    public class RSAkey
    {
        public 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;  
}  
  
public long Calculate_e()  
{  
    long d = 0;  
    long[] mas = {3, 5, 7, 11, 17, 31};  
  
    var ran = new Random();  
    d = mas[ran.Next(0, 5)];
```



```
        return d;

    }

    public long Calculate_d(long d, long m)
    {
        long e = 10;

        while (true)
        {
            if ((e * d) % m == 1)
                break;

            else
                e++;
        }

        return e;
    }
}
```

A.8 Текст файла Program.cs

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;


namespace WindowsFormsApp1
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();

            Application.SetCompatibleTextRenderingDefault(false);

            Application.Run(new Form1());
        }
    }
}
```

A.9 Текст файла ImageDate.cs

```
using System.Drawing;

using System.IO;

using System.Linq;

namespace ImDa
{
    public class ImageDate
    {
        public byte[] MasByte(Image image)
        {
            byte[] bytes;

            var ms = new MemoryStream();

            if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Jpeg))
            {
                image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
            }

            if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Png))
            {
                image.Save(ms, System.Drawing.Imaging.ImageFormat.Png);
            }

            if (image.RawFormat.Equals(System.Drawing.Imaging.ImageFormat.Bmp))
```

```
{  
    image.Save(ms, System.Drawing.Imaging.ImageFormat.Bmp);  
}  
return bytes = ms.ToArray();  
}  
  
public Image RetIm(int[] mas)  
{  
    byte[] b = mas.Select(i => (byte)i).ToArray();  
    var imageMemoryStream = new MemoryStream(b);  
    Image imgFromStream = Image.FromStream(imageMemoryStream);  
    return imgFromStream;  
}  
}
```

ДОДАТОК Б

Інтерфейс програми

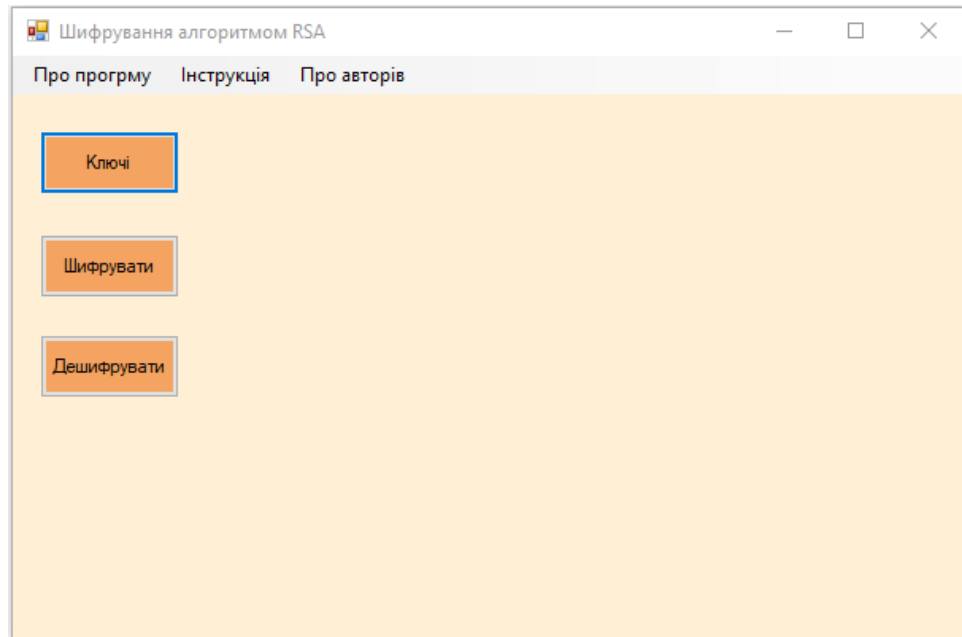


Рисунок Б.1 – Головне вікно програми

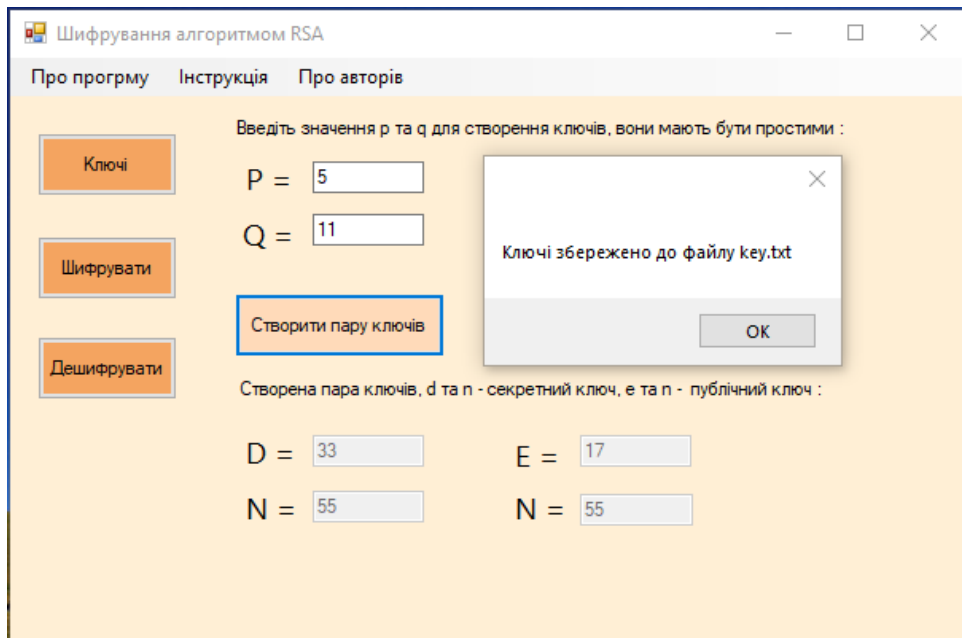


Рисунок Б.2 – Вікно панелі створення ключів

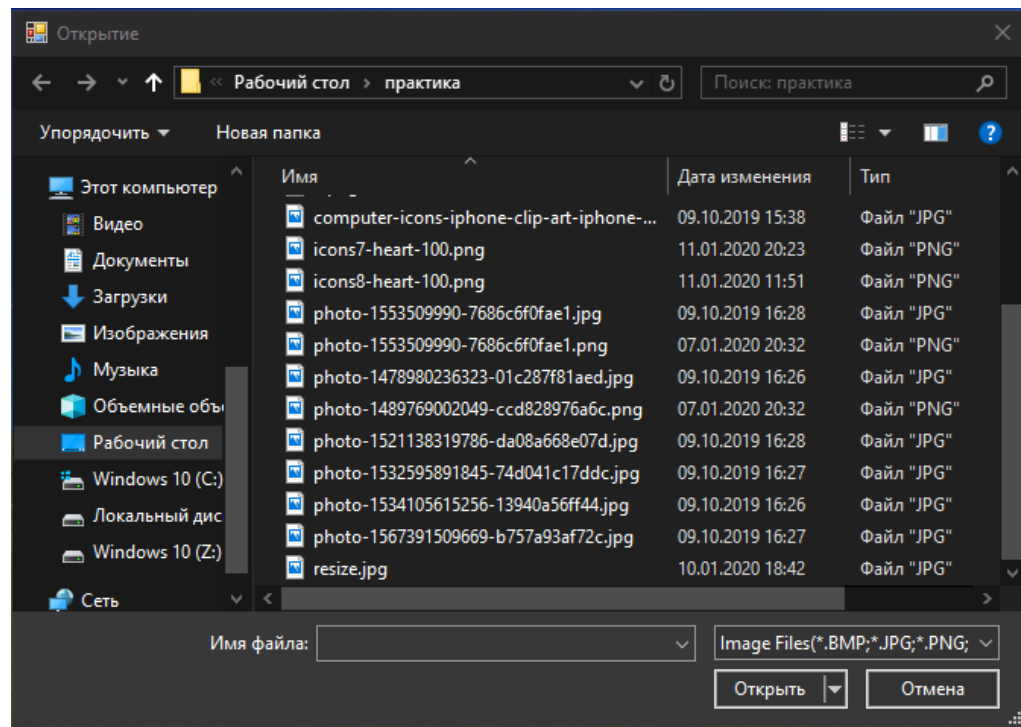


Рисунок Б.3 – Вікно для завантаження інформації

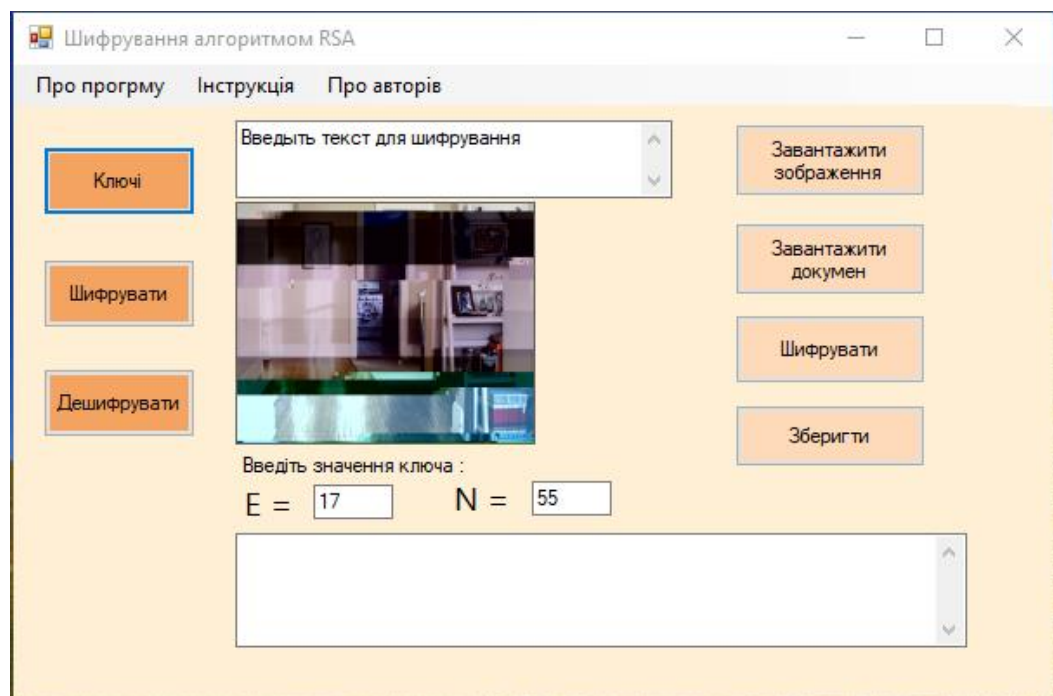


Рисунок Б.4 – Панель шифрування

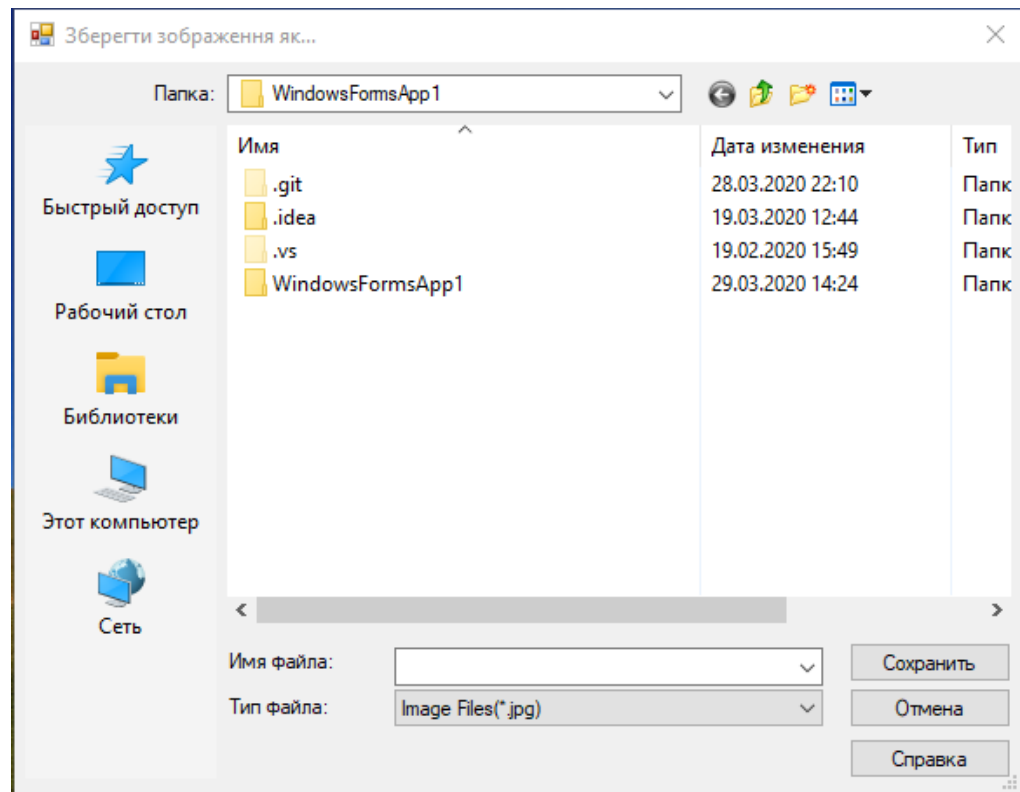


Рисунок Б.5 – Вікно для збереження інформації

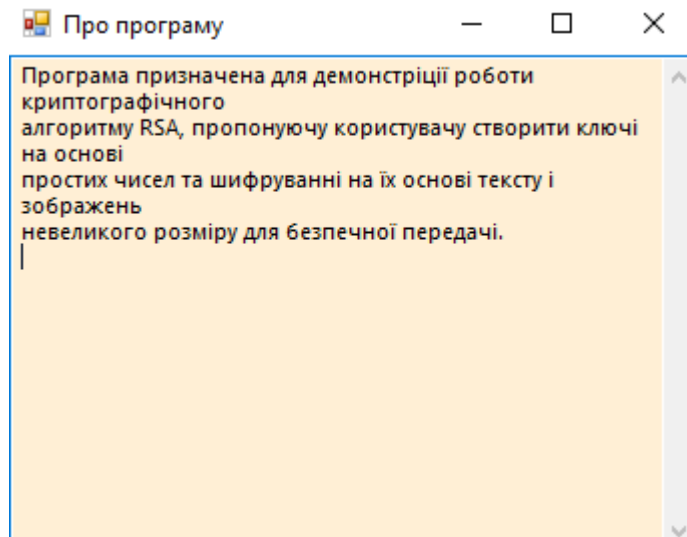


Рисунок Б.6 – Форма «Про програму»

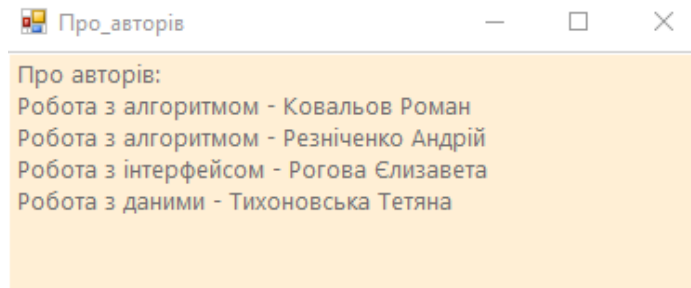


Рисунок Б.7 – Форма «Про авторів»

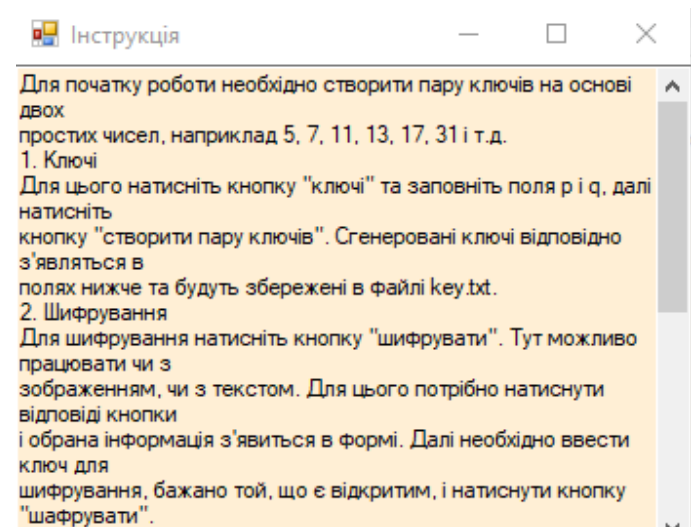


Рисунок Б.8 – Форма «Інструкція»

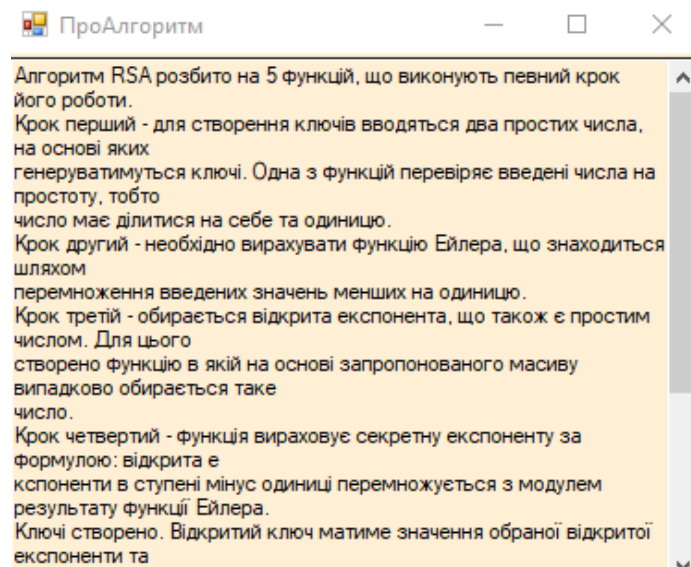


Рисунок Б.9 – Форма «Про алгоритм»

ДОДАТОК В

Слайди презентації

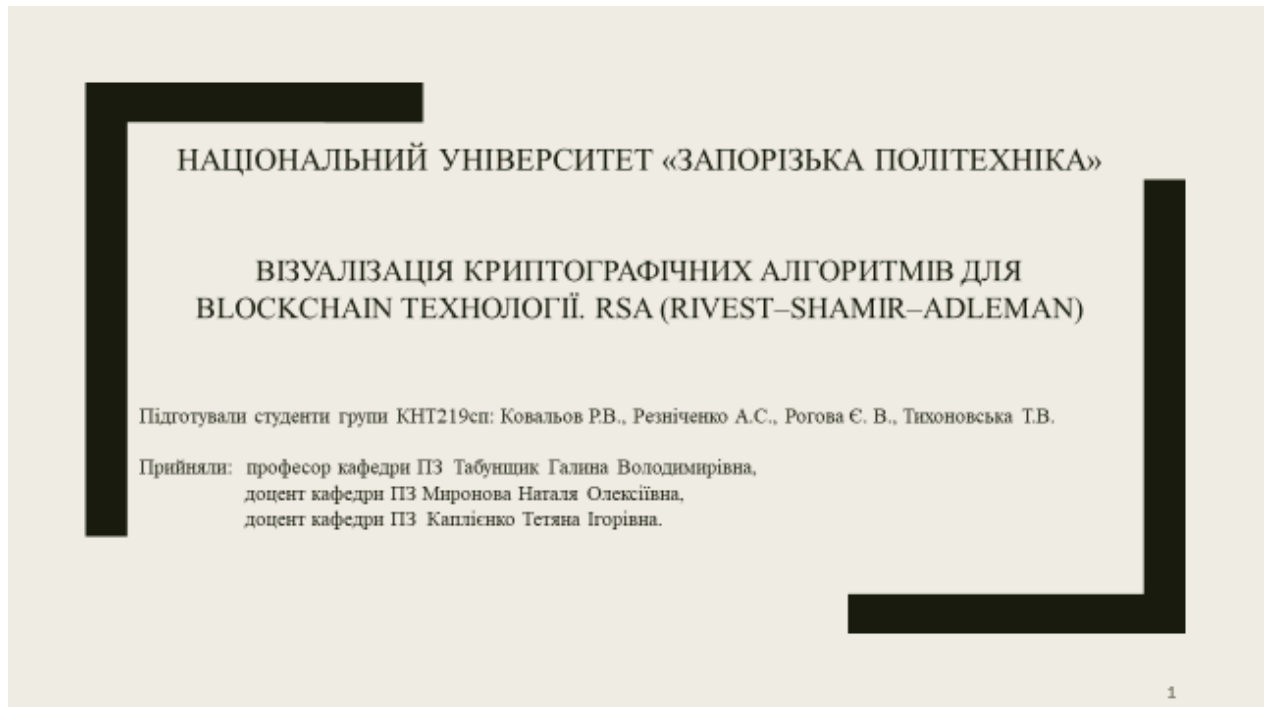


Рисунок В.1 – Титульний лист

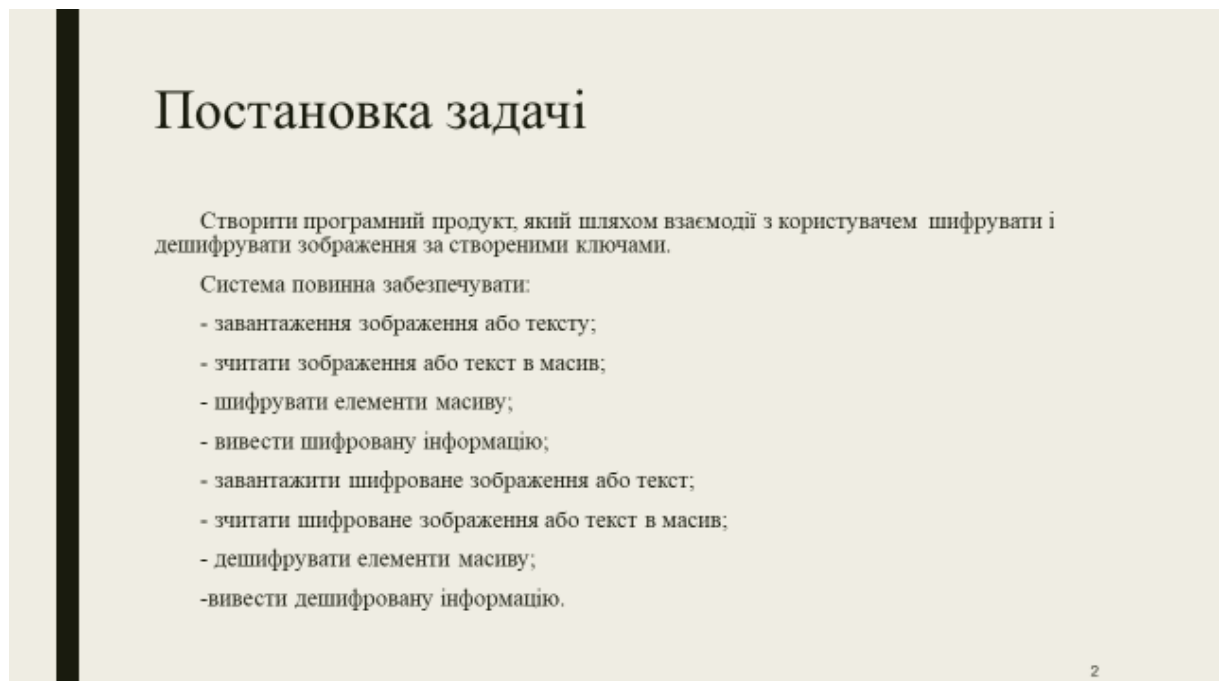


Рисунок В.2 – Постановка задачі

Мова програмування та середовище розробки

Основними перевагами мови C# є:

а) багато синтаксичного цукру. Синтаксичний цукор - це такі конструкції, які створені для полегшення написання і розуміння коду (особливо якщо це код іншого програміста) і не грають ролі при компіляції;

б) гарний розвиток завдяки підтримки Microsoft;

в) повністю ООП. Об'єктно-орієнтоване програмування - одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою;

г) величезна кількість вже готових класів на всі випадки життя;

г) є справжні (а не тільки вкладені) багатовимірні масиви і опціонально - перевірка переповнення.

JetBrains Rider - її перевагами є:

- JetBrains Rider є кросплатформним, він може працювати на платформах Windows, Mac або Linux з однаковою функціональністю і стабільністю;

- у складі Rider є значний набір для рефакторингу, перевірки коду і контекстних дій для всіх підтримуваних мов і технологій;

- підтримка систем контролю версій.



3

Рисунок В.3 – Мова програмування та середовище розробки

Основні класи програми

Клас	Опис
public class Form1	Клас Form1 є основним та містить в собі дані головної форми програми, забезпечує взаємодію з користувачем.
public class EncodeDecode	Клас EncodeDecode призначений для реалізації функцій шифрування і дешифрування даних методом RSA.
public class RSAkey	Клас RSAkey призначений для створення допоміжних змінних, що необхідні для позначення ключів методом RSA.
public class ПроАлгоритм	Відображення інформації про алгоритм, що використовується
public class ПроПрограму	Відображення інструкції для користувача

4

Рисунок В.4 – Основні класи програми 1

Клас	Опис
public class Інструкція	Відображення інформації стосовно призначення проекту
public class Про_авторів	Відображення інформації стосовно авторів проекту
public class ImageDate	Призначений для роботи з функціями, що необхідні для перетворення зображення в масив байт і навпаки.
static class Program	Клас Program, що має метод саме з нього програма починає свою роботу, там же ви побачите відкриття вікна: Application.Run (new Form1 ()) – що запускає першу форму.

5

Рисунок В.5 – Основні класи програми 2

Структура BigInteger

BigInteger структура надає тип BigInteger - це незмінний тип, який представляє довільно велике ціле число. Цей тип дозволяє зберігати цілі числа довільної (будь-який) довжини і вирішувати математичні операції з ними.

```

bi = new BigInteger(index);
bi = BigInteger.Pow(bi, (int)e);

BigInteger n_ = new BigInteger((int)n);

```

6

Рисунок В.6 – Структура BigInteger

Алгоритм RSA

Алгоритм RSA розбито на 5 функцій, що виконують певний крок його роботи.

Крок перший - для створення ключів вводяться два простих числа, на основі яких генеруватимуться ключі. Одна з функцій перевіряє введені числа на простоту, тобто число має ділитися на себе та одиницю.

Крок другий - необхідно вирахувати функцію Ейлера, що знаходиться шляхом перемноження введених значень менших на одиницю.

Крок третій - обирається відкрита експонента, що також є простим числом. Для цього створено функцію в якій на основі запропонованого масиву випадково обирається таке число.

Крок четвертий - функція вираховує секретну експоненту за формулою: відкрита експоненти в ступені мінус одиниці перемножується з модулем результату функції Ейлера.

Ключі створено. Відкритий ключ матиме значення обраної відкритої експоненти та добутку введених простих чисел. Закритий ключ матиме значення секретної експоненти та добутку введених простих чисел. Шифрування відбувається за наступною формулою: кожне значення введеної інформації зводиться в ступінь введеної експоненти та перемножується з модулем добутку простих чисел. Дешифрування відбувається за тією самою формулою, проте необхідно вказувати значення другого ключа.

7

Рисунок В.7 – Алгоритм RSA

Зберігання даних

Для збереження ключів використовуватиметься текстовий файл (.txt). Переваги :

- універсальність - текстовий файл може бути прочитаний (так чи інакше) на будь-якій системі або ОС ;
- стійкість - кожне слово та символ у такому файлі самодостатні і, якщо трапиться пошкодження байтів у такому файлі, то зазвичай можна відновити дані за контекстом або продовжити обробку решти вмісту.



8

Рисунок В.8 – Зберігання даних 1

Зображення формату bmp, png, jpeg, jpg – використовуються для шифрування (вони повинні мати не великий розмір, так як час роботи алгоритму збільшиться і можливі збої в роботі програми), бо вони є найбільш поширеними форматами зображень, про те шифроване зображення зберігається в форматі jpg. Його перевагами є:

- невеликий розмір файлу;
- впізнаваність усіма браузерами, графічними та текстовими редакторами, сумісність і правильне відображення на всіх комп'ютерах, планшетах і мобільних пристроях;
- правильна робота з кольоровими реалістичними зображеннями, де багато кольорних і контрастних переходів.

Також для шифрування використовується текст, що збережений в текстовому файлі.



9

Рисунок В.9 – Зберігання даних 2

Клас MemoryStream

Клас використано для розкладу зображення на масив байт і роботи з ним для безпосередньо шифрування зображення. Також він використовується для зворотного процесу – збору зашифрованого масиву в зображення.

```
byte[] b = mas.Select(i => (byte)i).ToArray();
var imageMemoryStream = new MemoryStream(b);
Image imgFromStream = Image.FromStream(imageMemoryStream);
```

10

Рисунок В.10 – Клас MemoryStream

Особливості реалізації системи роботи з зображенням

Особливістю є шифрування не всього масиву байт створеного з зображення, а обраних навмання байт, що є додатковою інформацією для дешифрування.

```
var ran = new Random();
int[] mas = new int[20];
int[] g = new int[20];

for (int i = 0; i < 20; i++)
{
    g[i] = ran.Next(100, mas.Length);
    mas[i] = mas[g[i]];
}

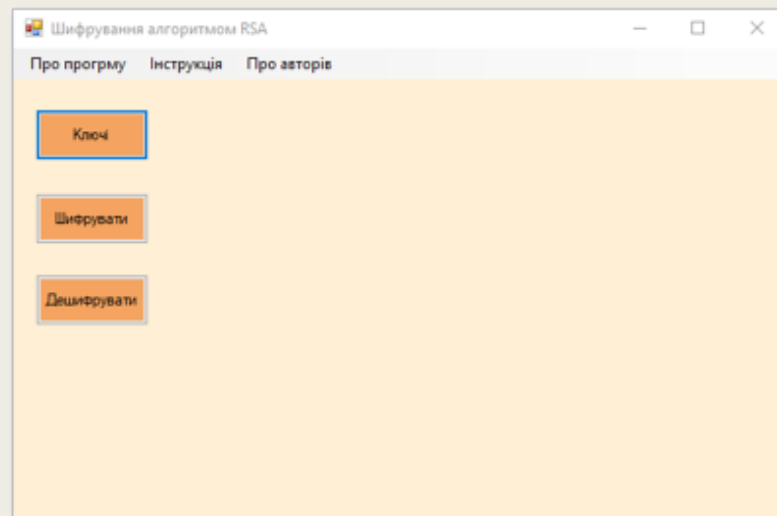
strIng = string.Join(" ", mas);

result = RSA_Encode(strIng, e_, n);
```

11

Рисунок В.11 – Особливості реалізації системи роботи з зображенням

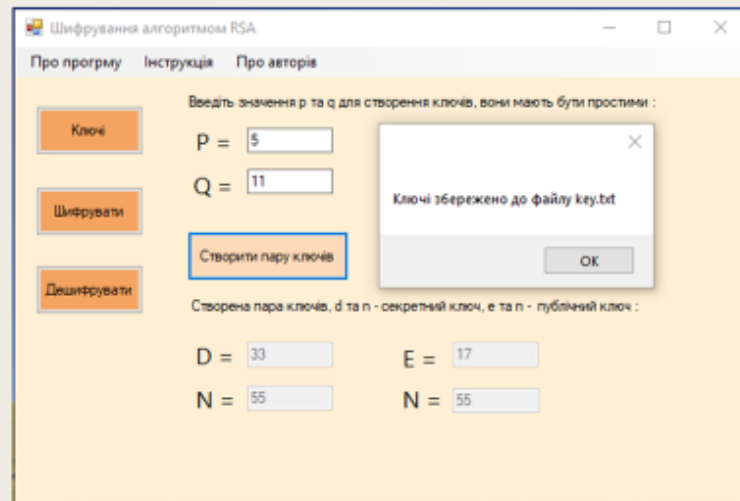
Інтерфейс програми: головна форма



12

Рисунок В.12 – Структура BigInteger

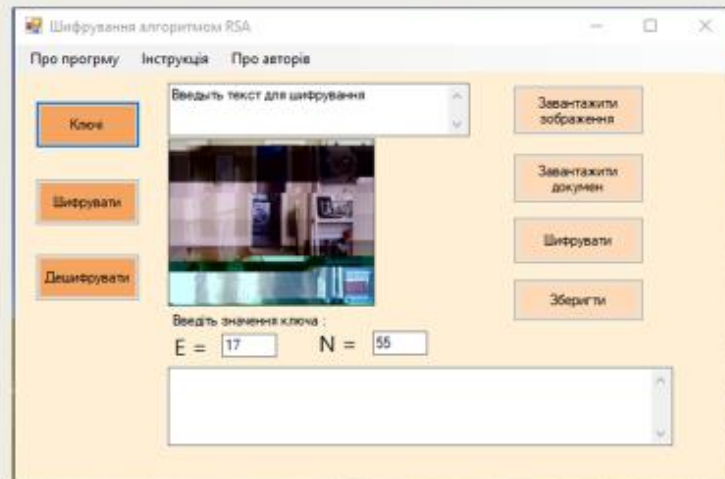
Інтерфейс програми: вікно панелі створення ключів



13

Рисунок В.13 – Інтерфейс програми: головна форма

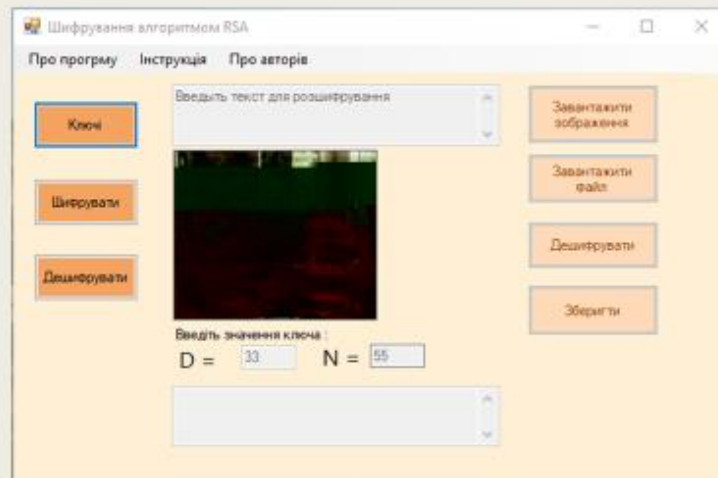
Інтерфейс програми: панель шифрування



14

Рисунок В.14 – Інтерфейс програми: вікно панелі створення ключів

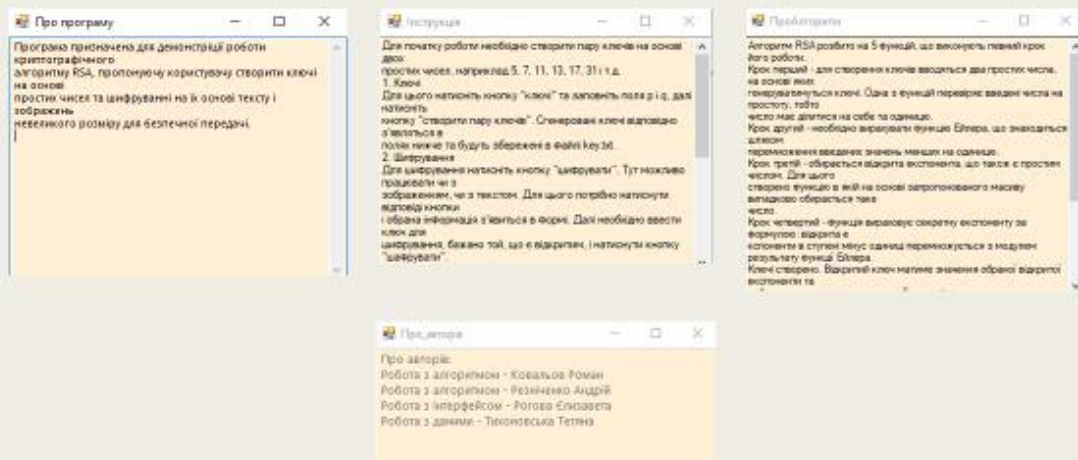
Інтерфейс програми: панель дешифрування



15

Рисунок В.15 – Інтерфейс програми: панель шифрування

Інтерфейс програми: інформаційні форми



16

Рисунок В.16 – Інтерфейс програми: панель дешифрування

Висновок

В ході виконання курсового проекту було вирішено наступні питання:

- розробити зрозумілий інтерфейс для користувача;
- продемонструвати результати роботи алгоритму;
- виконати роботу з зображеннями та текстовими файлами, а саме визначити у якій формі вони використовуються для роботи з алгоритмом.

17

Рисунок В.17 - Інтерфейс програми: інформаційні форми

Дякуємо
за увагу!!!

18

Рисунок В.18 - Заклучний слайд