Національний авіаційний університет

Факультет комп’ютерних наук та технологій

**ЗВІТ**

**З курсової роботи**

Дисципліна: «Криптологія»

Кафедра: прикладної математики

ОС: бакалавр

Спеціальність: 113 «Прикладна математика»

ОПП: «Прикладне програмне забезпечення»

Виконав: здобувач вищої освіти 3 курсу. 351 групи

Архіпов Олексій Тімурович

Перевірила: к.т.н.доц.Чолишкіна О.Г.

Дата: 25 травня 2023 рік

Оцінка \_\_\_\_\_\_\_\_\_\_\_\_

Київ 2023

**Зміст**

Вступ……………………………………………………………………………...3

1.Теоретичні відомості…………………………………………………………..5

1.1 Алгоритм BBS..………………………………………………………..5

1.2 Алгоритм IDEA………………………………………………………..6

1.3 Алгоритм MD5………………………………………………………..14

1.4 Алгоритм RSA………………………………………………………...17

1.5 Цифровий конверт……………………………………………………20

2.Практична частина…………………………………………………………….22

2.1 Опис роботи програми……………………………………………….22

2.2 UML – діаграми………………………………………………………33

Висновки………………………………………………………………………....37

Списки використаної літератури……………………………………………….39

Додаток А………………………………………………………………………...41

Додаток Б…………………………………………………………………………51

Додаток В………………………………………………………………………...51

**Вступ**

Основними напрямами використання криптографічних методів в прикладному програмному забезпеченні є переказ конфіденційної інформації

по каналах зв’язку (електронна пошта, телефонні розмови, Інтернет зв’язок

тощо), встановлення автентичності переданих повідомлень, зберігання

інформації на носіях в зашифрованому вигляді, а також застосування

електронного цифрового підпису фізичними та юридичними особами.

Саме тому тематика курсової роботи охоплює такі важливі розділи

криптографії, як симетричні криптосистеми, криптосистеми з відкритим

ключем та системи формування електронного цифрового підпису.

Цифровий конверт є важливою технологією в області безпеки інформації. В даний час все більше даних передається і зберігається в електронному вигляді, і захист цих даних від несанкціонованого доступу і внесення змін стає все більш важливим. Цифровий конверт дозволяє забезпечити конфіденційність і повноту даних, що передаються. Це робить його необхідним інструментом для отримання інформації в різних сферах, включаючи захист банківської, медичної та правової сфери.

Мета:застосування на практиці теоретичних знань з принципів побудови технології використання криптографічних систем для створення прикладного програмного забезпечення захисту інформації в комп’ютерних мережах та на зовнішніх носіях.

Завдання:

Розробити програмне забезпечення криптографічного захисту електронних даних «Цифровий конверт», що відповідає наступним вимогам.

1.На вимогу користувача повинна відбуватися генерація криптографічних ключів (пара – відкритий/закритий ключі для асиметричного шифрування та цифрового підпису й таємний ключ симетричного шифрування). Також

повинна бути реалізованою можливість використання ключів, що були

сгенеровані раніше з певного файлу або сертифікованого сховища (користувач вказує шлях).

2. Реалізувати можливість завантаження файлу електронних даних.

3. Розрахувати значення хеш-функції повідомлення згідно з вказаним

алгоритмом.

4. Реалізувати процедуру цифрового підпису, використовуючи закритий

ключ користувача та алгоритм цифрового підпису згідно з варіантом.

5. Провести направлене шифрування даних симетричним криптоалгоритмом з використанням таємного ключа користувача.

6. Зробити експорт отриманого цифрового конверту.

7. Реалізувати можливість верифікації цифрового підпису повідомлення.

В цій роботі ми розглянемо створення цифрового конверта на прикладі використання алгоритмів IDEA, MD5 і RSA, включаючи генерацію ключів для алгоритмів IDEA та RSA, шифрування повідомлення з використанням IDEA, створення хешу повідомлення з використанням MD5 та створення цифрового підпису на основі закритого ключа RSA. Також ми розглянемо процес перевірки цифрового підпису.

**1. Теоретичні відомості**

**1.1 Алгоритм BBS**

Алгоритм BBS (Blum-Blum-Shub) — це криптографічний генератор псевдовипадкових чисел, запропонований у 1986 році Ленором Блум, Менуелем Блумом і Шубом. Генератор заснований на труднощі зворотності квадратичних розрахунків по модулю великих простих чисел [1], [9].

Алгоритм BBS використовує наступні параметри:

* Два великих простих числа p і q, таких, що залишок від ділення на 4 дорівнює 3.
* Добуток n = p \* q.
* Початкове значення s, яке повинно бути взаємно простим з n.

Генерація псевдовипадкової послідовності здійснюється таким чином:

* Задається початкове значення x = s^2 mod n.
* Для кожного наступного значення x\_i генерується нове випадкове число, рівне x\_i^2 mod n. Молодший біт результату розраховується як черговий біт вихідної послідовності [12].

Генератор BBS вважається криптографічно стійким, оскільки вирахування наступного числа в послідовності вимагає вирахування квадрата попереднього числа за модулем n, що є складною задачею при достатньо великих значеннях n [1].

У цій роботі я використав цей генератор для створення ключа шифрування для алгоритму IDEA.

Тобто я рахую 128 значень х за формулою, потім для цього значення знаходжу відсоток від ділення на 2.

Таким чином я створюю масив бітів, котрі я потім конвертую у 16 байт, а потім у строку.

**1.2 Алгоритм IDEA**

Розробимо програмне забезпечення криптографічного захисту інформації

за використанням симетричних криптосистем на прикладі алгоритму IDEA.

Алгоритм IDEA оперує 64-бітовими блоками початкових даних. Безперечною перевагою алгоритму IDEA є те, що його ключ має довжину 128 бітів. Той самий алгоритм використовується також для зашифрування й розшифрування [4].

Метою розробки IDEA було створення відносно стійкого криптографічного алгоритму з досить простою реалізацією [4].

Характеристики IDEA, що характеризують криптографічну стійкість алгоритму:

Довжина блока: довжина блока повинна бути достатньою, щоб приховати всі статистичні характеристики початкового повідомлення. З іншого боку, складність реалізації криптографічної функції зростає експоненціально відповідно до розміру блока. Використання блока розміром у 64 біти в 90-ті рр. XX ст. означало достатнюсилу. Більше того, використання режиму шифрування СВС говорить про подальше посилення цього аспекту алгоритму [4];

довжина ключа: довжина ключа повинна бути досить великою для того, щоб запобігти можливості простого перебору ключа. За довжини ключа 128 бітів IDEA вважається досить безпечним [4];

конфузія: зашифровані дані повинні залежати від ключа складним і заплутаним способом [4];

дифузія: кожний біт початкових даних повинен впливати на кожний біт зашифрованих даних. Поширення одного незашифрованого біта на велику кількість зашифрованих бітів приховує статистичну структуру початкових даних. Визначити, як статистичні характеристики зашифрованих даних залежать від статистичних характеристик початкових даних, повинно бути непросто. IDEA з цього погляду є дуже ефективним алгоритмом [4].

В IDEA два останні пункти виконуються за допомогою трьох операцій. Це відрізняє його від DES, де все побудовано на використанні операції xor і маленьких нелінійних S-блоків заміни [4].

Кожна операція виконується над двома 16-бітовими входами та створює один 16-бітовий вихід. Цими операціями є:

1. Побітове xor, що позначається таким символом.
2. Сума цілих чисел за модулем 2^16 (65536), при цьому входи й виходи трактуються як беззнакові 16-бітові цілі. Цю операцію позначають як .
3. Множення цілих чисел за модулем 2^16 + 1 (65537), при цьомувходи й виходи трактуються як беззнакові 16-бітові цілі, за винятком того, що блок із одних нулів трактується як 2^17. Цю операцію позначають як [4].

Ці три операції є несумісними в тому сенсі, що:

1. Не існує пари з трьох операцій, що задовольняють дистрибу-

тивному закону [14]. Наприклад,



2. Не існує пари з трьох операцій, що задовольняють асоціатив-

ному закону [14]. Наприклад,



Використання комбінації з цих трьох операцій забезпечує комплексну трансформацію входу, роблячи криптографічний аналіз складнішим, ніж у такому алгоритмі як DES, заснованому виключно на функції xor [4].

Розглянемо загальну схему шифрування IDEA (рис 1.1). Як і в будь-якому алгоритмі шифрування, тут існує два входи: незашифрований блок і ключ. У даному випадку незашифрований блок має довжину 64 біти, ключ має довжину 128 бітів.

Алгоритм IDEA складається з восьми раундів, за якими йде кінцеве перетворення. Алгоритм поділяє блок даних на чотири 16-бітові підблоки. Кожний раунд отримує на вході чотири 16-бітові підблоки та створює чотири 16-бітові вихідні підблоки. Кінцеве перетворення також отримує на вході чотири 16-ти бітові підблоки та створює на виході чотири 16-бітові підблоки. Кожний раунд використовує шість 16-бітових раундових ключів, кінцеве

перетворення використовує тільки чотири 16-бітові раундові ключі, тобто всього в алгоритмі використовується 52 раундових ключі [4].

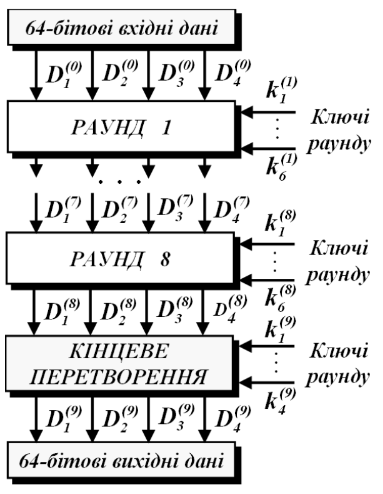


Рисунок 1.1. Структура алгоритму IDEA.

Ця структура повторюється в алгоритмі вісім разів, забезпечуючи високоефективну дифузію.

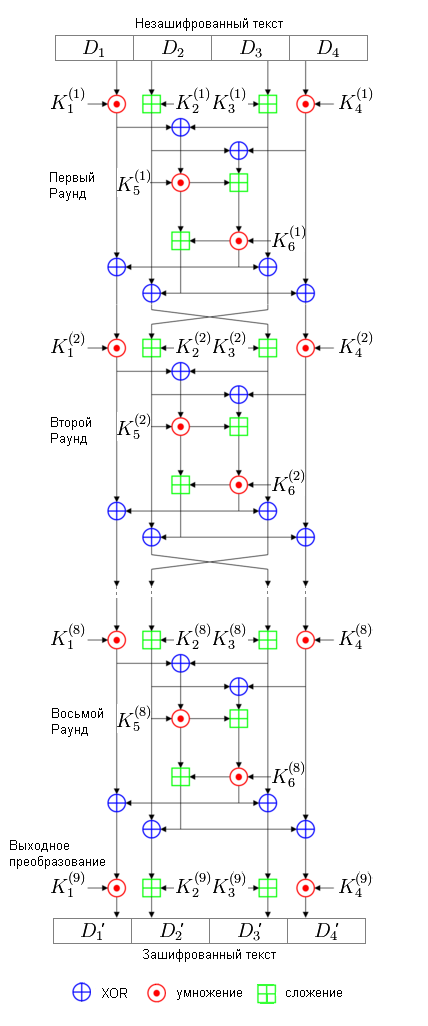


Рисунок 1.2. Детальніша структура алгоритму IDEA.

**Генерація раундових ключів для за шифрування даних**

З 128-бітного ключа для кожного з восьми раундів шифрування генерується по шість 16-бітних подключів, а для вихідного перетворення генерується чотири 16-бітних підключа. Всього буде потрібно 52 = 8 x 6 + 4 різних подключів по 16 біт кожен. Процес генерації п'ятдесяти двох 16-бітних ключів полягає в наступному [3]:

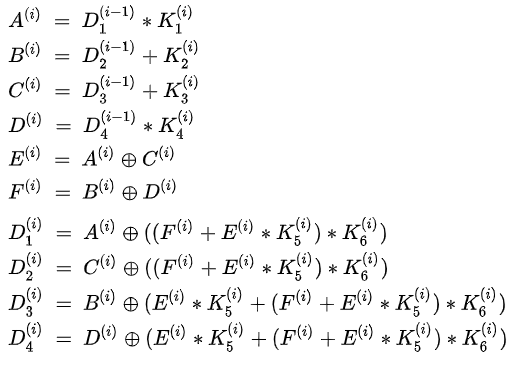
* Насамперед, 128-бітний ключ розбивається на вісім 16-бітних блоків. Це будуть перші вісім подключів по 16 біт кожен — (�1(1)�2(1)�3(1)�4(1)�5(1)�6(1)�1(2)�2(2)) 
* Потім цей 128-бітний ключ циклічно зсувається вліво на 25 позицій, після чого новий 128-бітний блок знову розбивається на вісім 16-бітних блоків. Це вже наступні вісім подключів по 16 біт кожен — (�3(2)�4(2)�5(2)�6(2)�1(3)�2(3)�3(3)�4(3)) 
* Процедура циклічного зсуву і розбивки на блоки триває до тих пір, поки не будуть згенеровані всі 52 16-бітних підключа [3].



Рисунок 1.3. Таблиця підключів.

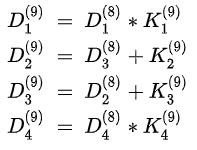
**Математичний опис**

* Блок відкритого тексту розміром 64 біт ділиться на чотири рівні подблока розміром по 16 біт (�1(0),�2(0),�3(0),�4(0)) 
* Для кожного раунду  (�=1...8) обчислюються:



Результатом виконання восьми раундів будуть наступні чотири підблока 

Виконується вихідне перетворення (і = 9):



Результатом виконання вихідного перетворення є зашифрований текст (�1(9),�2(9),�3(9),�4(9))  [3].

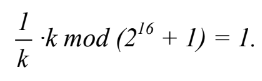
**Генерація раундових ключів для розшифрування даних**

Метод обчислення, що використовується для розшифрування даних по суті такий самий, як і при його зашифруванні. Єдина відмінність полягає в тому, що для розшифрування використовуються інші раундові ключі. У процесі розшифрування раундові ключі повинні використовуватися у зворотному порядку у відношенні до процесу зашифрования [20].

Для формування раундових ключів розшифрування спочатку формуються раундові ключі зашифрування [4].

Процес створення раундових ключів розшифрування визначається таким чином: перший і четвертий ключі i-го раунду розшифрування виходять із першого й четвертого ключа (10-i)-го раунду зашифрування мультиплікативної інверсії. Для 1-го і 9-го раундів другий і третій ключі розшифрування виходять із другого й третього ключів 9-го і 1-го раундів зашифрування адитивної інверсії. Для раундів з 2-го по 8-й другий і третій ключі розшифрування виходять із третього та другого ключів з 8-го по 2-й раунди зашифрування адитивної інверсії. Останні два ключі i-го раунду розшифрування дорівнюють останнім двом ключам (9-i)-го раунду зашифрування [4].





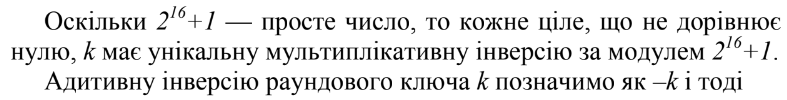






Рис 1.4. Ключі для розшифрування.

Криптостійкість IDEA заснована на використанні великої кількості раундів шифрування, кожен з яких включає різні перетворення, такі як заміни, перестановки і операції над бітами. Це робить IDEA криптостійким, щоб використовувати його для шифрування конфіденційної інформації [2].

Однак, з часом з'являються нові методи злому та атак на алгоритми шифрування, і IDEA вже не є найбільш рекомендованим алгоритмом для використання в нових системах [14].

**1.3 Алгоритм MD5**

Розробимо систему електронного цифрового підпису за використанням

алгоритму RSA та алгоритму хешування MD5.

Алгоритм MD5 — 128-бітний алгоритм хешування, розроблений професором Рональдом Л. Рівестом в 1991 році. Призначений для створення «відбитків» або «дайджестів» повідомлень довільної довжини. Прийшов на зміну MD4, що був недосконалим [6], [7].

**Початковий етап підготовки**

* Вхідні дані вирівнюються так, щоб їхній розмір можна було порівняти з 448 по модулю з 512. Спочатку дописують одиничний біт (навіть якщо довжина порівняна з 448), далі необхідна кількість нульових бітів [19].
* Дописування 64-бітного представлення довжини даних по вирівнюванню. Якщо довжина перевищує 264−12^64 - 1, то дописують молодші біти [6].

**Допоміжні таблиці та функції**

Ініціалізуть 4 змінних розміром по 32 біта [5]:

А: 01 23 45 67,

В: 89 АВ CD EF,

С: FE DC ВА 98,

D: 76 54 32 10.

Вирівнювані дані розбиваються на блоки по 32 біта, і кожен проходить 4 раунди з 16 операторів [16].

,

де

а, b, с, d — чотири слова буфера в певному порядку, що змінюються в процесі роботи,

g - одна з примітивних функцій F, G, Н, I,

<< 5 — циклічний зсув вліво (поворот) 32-бітового аргументу на 5 біт,

X[k] - k -е 32-битовое слово в q -м 512-бітовим блоком повідомлення.

T[i] - i -е 32-бітовое слово в матриці T,

+ - складання по модулю 2^32.



Рисунок 1.5. Елементарна операція MD5 (один крок раунду).

За допомогою MD5 перевіряли цілісність та справжність завантажених файлів – так, деякі програми поставляються разом зі значенням контрольної суми. Наприклад, пакети для інсталяції вільного ПЗ [5] .

MD5 використовувався для хешування паролів. У системі UNIX кожен користувач має свій пароль, і його знає лише користувач. Для захисту паролів використовується хешування. Передбачалося, що отримати цей пароль можна лише повним перебором [17].

Існує думка, що зламати хеш MD5 неможливо, проте це неправда, існує безліч програм, що підбирають вихідне слово на основі хеша. Абсолютна більшість з них здійснює перебір за словником, проте існують такі методи як Rainbow Crack, він заснований на генеруванні безлічі хешей з набору символів, щоб по базі, що вийшла, здійснювати пошук хеша [11].  
Також у MD5, як у будь-якої хеш-функції, існує таке поняття як колізії - це отримання однакових хеш для різних вихідних рядків. У 1996 році Ганс Доббертін знайшов псевдоколізії в MD5, використовуючи певний буфер, що ініціалізує (ABCD). Також у 2004 році китайські дослідники Ван Сяоюнь, Фен Денгуо, Лай Сюецзя та Юй Хунбо оголосили про виявлену ними вразливість у алгоритмі, що дозволяє за невеликий час (1 годину на кластері IBM p690) знаходити колізії. Однак у 2006 році чеський дослідник Властиміл Кліма опублікував алгоритм, що дозволяє знаходити колізії на звичайному комп'ютері з будь-яким початковим вектором (A,B,C,D) за допомогою методу, який він назвав «тунелюванням» [6].

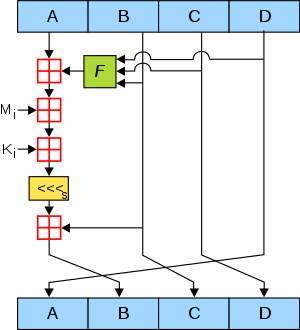


Рисунок 1.6. Блок-схема MD5.

**1.4 Алгоритм RSA**

Асиметричні криптосистеми — ефективні системи криптографічного захисту даних, які також називають криптосистемами з відкритим ключем. В таких системах для зашифровування даних використовують один ключ, а для розшифровування — інший (звідси і назва — асиметричні) [15]. Перший ключ є *відкритим* і може бути опублікованим для використання усіма користувачами системи, які шифрують дані. Розшифровування даних за допомогою відкритого ключа неможливе. Для розшифровування даних отримувач зашифрованої інформації використовує другий ключ, який є секретним (*закритим*). Зрозуміло, що ключ розшифровування не може бути визначеним з ключа зашифровування [10].

Головне досягнення асиметричного шифрування в тому, що воно дозволяє людям, що не мають наперед наявної домовленості про безпеку, обмінюватися секретними повідомленнями. Необхідність відправникові й одержувачеві погоджувати таємний ключ по спеціальному захищеному каналу цілком відпала [10].

Алгоритм RSA (Rivest–Shamir–Adleman) - це асимметричний алгоритм шифрування, який використовує два ключа: відкритий і закритий. Він був розроблений в 1977 році Рональдом Рівестом, Аді Шаміром і Леонардом Адлеманом [8].

Систему побудовано на факті, що добуток великих простих чисел

здійснюється легко, проте розкладання на множники добутку двох таких чисел

практично неможливе. Доведено (теорема Рабіна), що розкриття шифру RSA

еквівалентно такому розкладу. Тому для довільної довжини ключа дають нижню оцінку кількості операцій для розкриття шифру, а з урахуванням

продуктивності сучасних комп’ютерів оцінити і необхідний для цього час [5].

Можливість гарантовано оцінити захищеність алгоритму RSA стала

однією з причин популярності цієї криптосистеми на фоні десятків інших схем. Тому алгоритм RSA використовується в банківських комп’ютерних мережах, особливо для роботи з віддаленими клієнтами [5].

Алгоритм RSA містить наступні кроки:

Крок 1: Генерація ключів

1.1. Вибираються два простих числа p і q.

1.2. Обчислюється їх добуток n = p \* q.

1.3. Обчислюється значення функції Ейлера від числа n: φ(n) = (p - 1) \* (q - 1).

1.4. Вибирається ціле число e, таке, що 1 < e < φ(n), і e взаємно просто з φ(n). e стане відкритим ключем.

1.5. Обчислюється число d, зворотне по модулю φ(n) до числа e, тобто

d \* e = 1 (mod φ(n)). d стане закритим ключем.

1.6. Пара ключей (e, n) стає відкритим ключем, а пара ключей (d, n) стає закритим ключем.

Крок 2: Шифрування повідомлень

2.1. Перетворення тексту M в число m.

2.2. Шифрування числа m з використанням відкритого ключа (e, n): c = m^e mod n.

2.3. Зашифроване повідомлення c є шифротекстом.

Крок 3: Дешифрування повідомлень

3.1. Отриманий шифротекст c розшифровується за допомогою закритого ключа (d, n): m = c^d mod n.

3.2. Отримане число m відображається в повідомленні М.

Алгоритм RSA може використовуватися для шифрування та дешифрування повідомлень, а також для створення електронних підписів.

Ми можемо побачити на рисунку 1.7 принцип шифрування та дешифрування за допомогою алгоритму RSA.



Рисунок 1.7. Алгоритм RSA.

Алгоритм створення цифрового підпису за допомогою алгоритму RSA:

1. Обрати будь-яку хеш-функцію, наприклад SHA-256, для створення хеш-значення повідомлення, яке потрібно підписати.
2. Знайти хеш-значення повідомлення за допомогою обраної хеш-функції.
3. Зашифрувати хеш-значення повідомлення за допомогою зачиненого ключа d, використовуючи алгоритм RSA.
4. Отримане зашифроване значення і буде підписом для нашого повідомлення.
5. Для перевірки підпису, потрібно цей підпис розшифрувати за допомогою публічного ключа e, перевірити отриманий хеш з хешем повідомлення, якщо вони співпадають, то підпис е вірною

Криптостійкість RSA заснована на складності факторизації великих цілих чисел на прості множники. Щоб зламати RSA, потрібно знайти два простих числа, з яких було сформовано відкрите число, що є дуже складним завданням на даний момент [13]. В даний час немає відомих алгоритмів, які можуть ефективно розкласти велику кількість на прості множники. Однак, при неправильному використанні RSA, наприклад, при використанні надто коротких ключів або при використанні вразливих реалізацій, можливі атаки на алгоритм RSA [18].

* 1. **Цифровий конверт**

На сьогоднішній день існує проблема передачі захищеної інформації між людьми.

Цифровий конверт - це спосіб забезпечення конфіденційності, цілісності та довжини інформації за допомогою криптографічних алгоритмів. Цифровий конверт дозволяє зашифрувати інформацію за допомогою симетричних алгоритмів шифрування, а також підписати дані за допомогою алгоритмів хешування та цифрового підпису.

У цій роботі ми розглядаємо створення цифрового конверту на прикладі використання алгоритмів IDEA, MD5 і RSA.

Розглядаємо процес створення цифрового конверту, включаючи генерацію ключів для алгоритмів IDEA і RSA, шифрування повідомлень з використанням IDEA, створення хеш-повідомлень з використанням MD5 і створення цифрового підпису на основі закритого ключа RSA. Також ми розглядаємо процес перевірки цифрового підпису та розшифровки повідомлень.

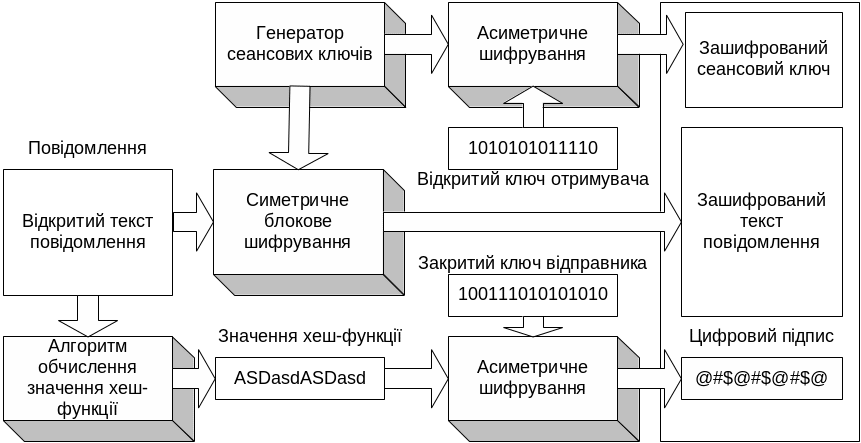


Рисунок 1.8. Схема формування цифрового конверту.

На рисунку 1.8 ми бачимо, що для створення цифрового конверту, вхідний текст шифрується симетричним блоковим шифруванням, за допомогою попередньо згенерованого ключа, потім цей ключ шифрується асиметричним шифруванням за допомогою відкритого ключа і зашифрований ключ та зашифрований текст додаються до файлу. Потім для вхідного повідомлення обраховується хеш і за допомогою асиметричного шифрування створюється цифровий підпис, який також додається до файлу. Цей файл і буде цифровим конвертом.

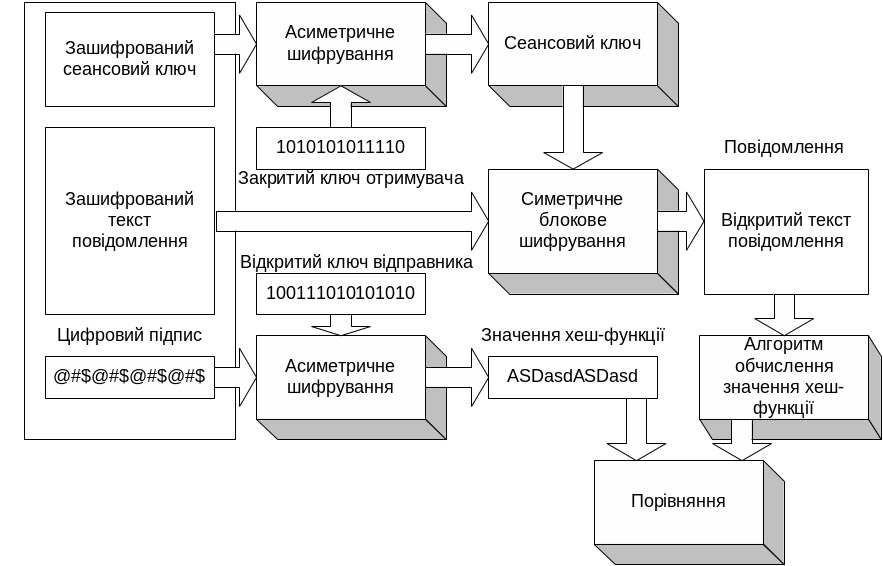


Рисунок 1.9. Схема перевірки цифрового конверту.

На рисунку 1.9 відбувається перевірка цифрового конверту. Для цього, спочатку розшифровуємо ключ для симетричного шифрування, потім беремо зашифрований текст і розшифровуємо за допомогою цього ключа. У нас повинно вийти початкове повідомлення потім розраховуємо його хеш за допомогою нашого алгоритму хешування.

Далі за допомогою відкритого ключа ми розшифровуємо цифровий підпис і знаходимо хеш. Порівнявши ці два хеші, мі зможемо сказати про те чи було змінено повідомлення, чи ні.

Алгоритми IDEA, MD5 та RSA в цілому вважаються криптографічно стійкими і широко використовуються в різних додатках для забезпечення конфіденційності, цілісності та автентичності даних. Однак, залежно від конкретних вимог безпеки, можуть бути більш відповідні алгоритми. Наприклад, для забезпечення конфіденційності даних часто використовують алгоритми симетричного шифрування, такі як AES, який вважається швидшим та безпечнішим, ніж IDEA. Крім того, для забезпечення цілісності та автентичності даних можуть використовуватися хеш-функції, відмінні від MD5, такі як SHA-256 або SHA-3, які вважаються більш стійкими до атак. RSA загалом гарно підходить для створення цифрового підпису тож його можна не змінювати.

**2. Практична частина**

**2.1 Опис роботи програми**

Базуючись на теоретичній частині, що було наведено раніше, я створив програму для створення та перевірки цифрового конверта за допомогою алгоритмів IDEA, MD, RSA.

Програму було реалізовано на мові c# у winforms.

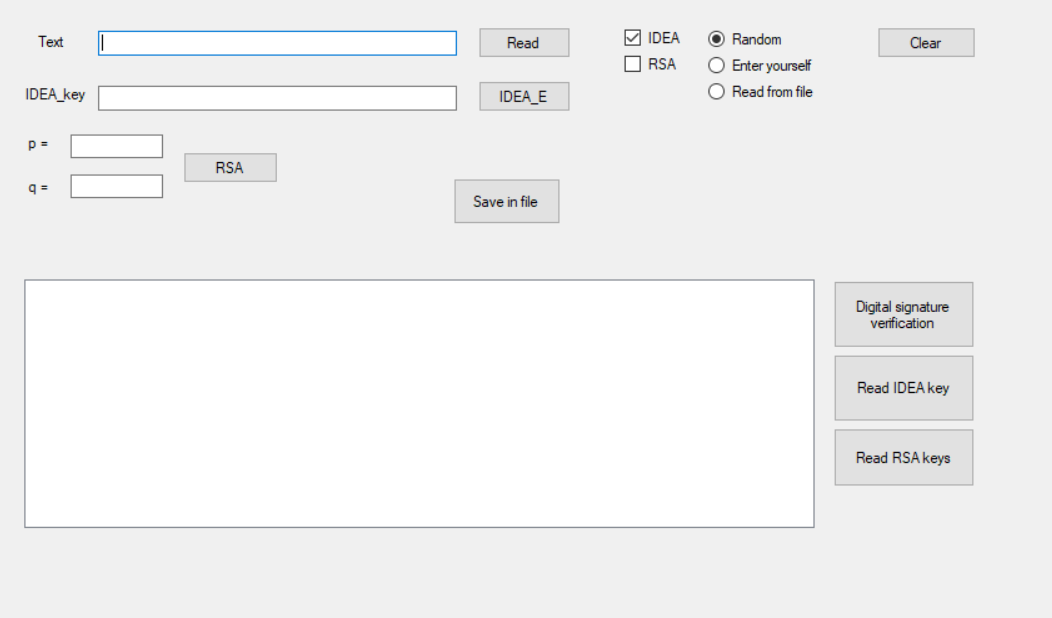


Рисунок 2.1. Головне вікно програми.

Після запуску програми з’явиться вікно, як на рисунку 2.1.

Вхідне повідомлення можна ввести у textbox, що знаходиться напроти слова Text, або можна натиснути на button Read, після чого відкриється вікно для вибору файла, з якого можна буде взяти вхідне повідомлення.

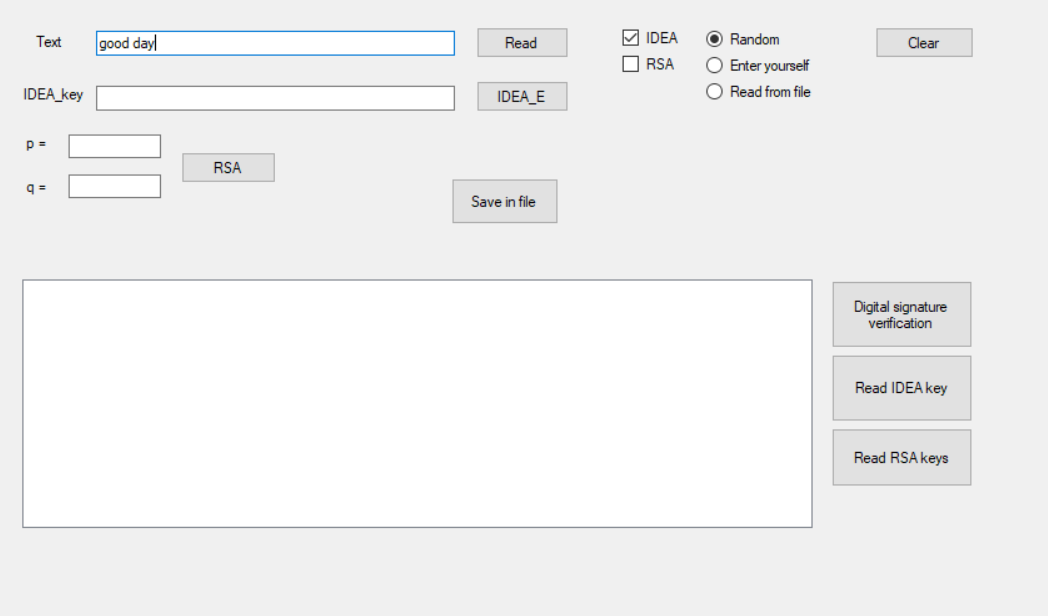


Рисунок 2.2. Введення повідомлення через textbox.

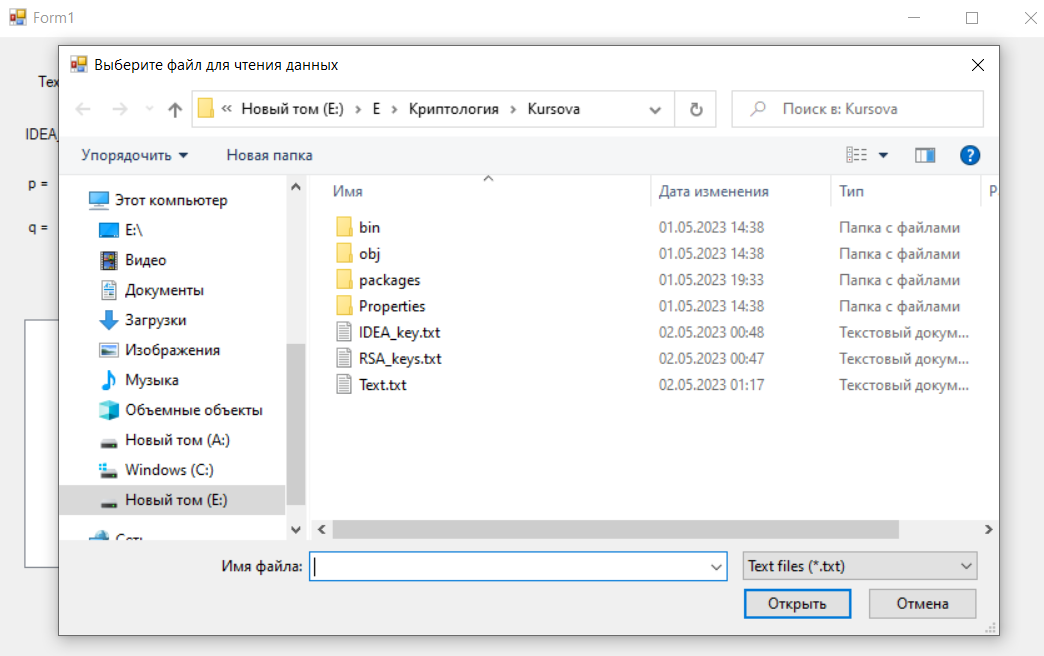


Рисунок 2.3. Введення повідомлення через button Read.

Обравши файл Text.txt, тповідомлення, що там знаходиться, буде зчитано програмою та буде виведено у textbox.

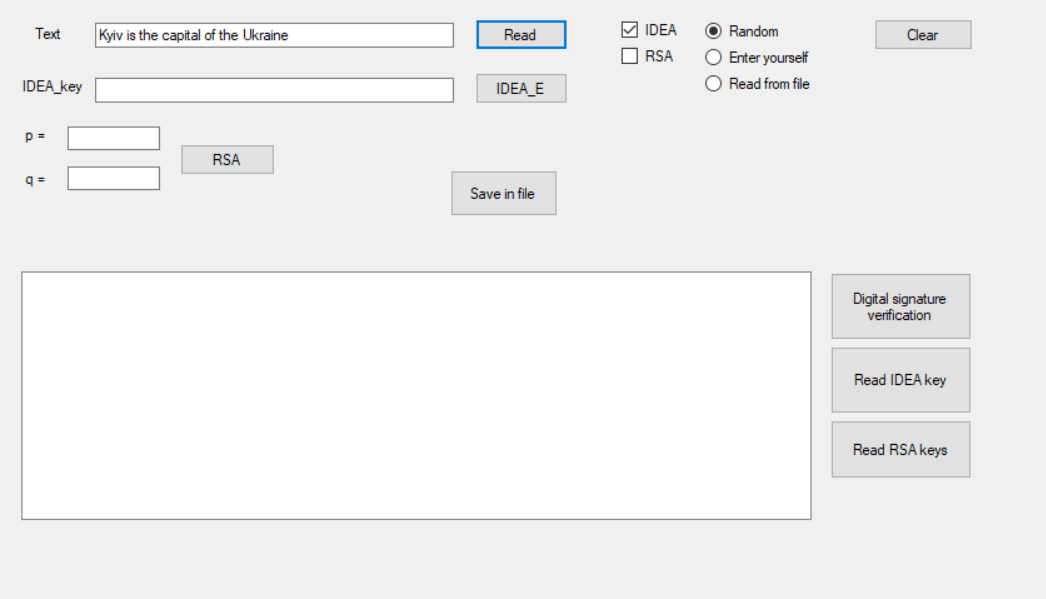


Рисунок 2.4. Результат зчитування повідомлення з файлу.

На даному етапі у нас є вхідне повідомлення. Далі потрібно обрати спосіб введення ключа для алгоритму IDEA. Згори праворуч, ми бачимо два перемикача між алгоритмами, то способами введення ключів. На разу обрано алгоритм IDEA та random, тобто рандомно згенерувати ключ для симетричного шифрування довжиною 128 біт. Ми також можемо обрати спосіб Enter yourself, тобто ввести самому, або також зчитати файл. Наприклад на рисунку 2.3 видно файл з назвою IDEA\_key, там лежить ключ для шифрування.

Оберем наприклад ввести самому цей ключ.

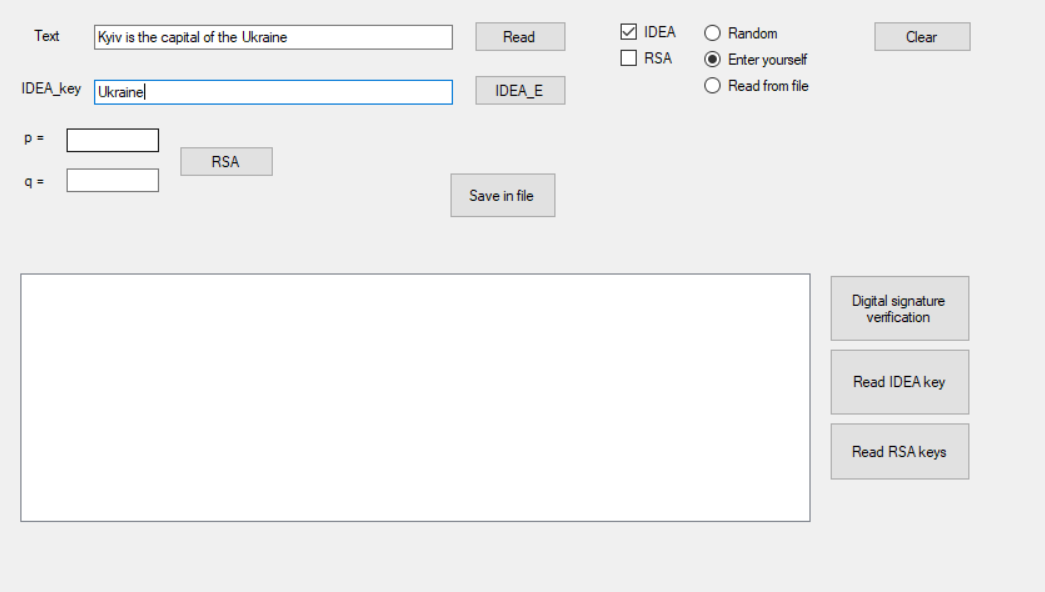


Рисунок 2.5. Ключ для IDEA.

Після вводу ключа можемо натиснути button IDEA\_E, після чого з’явиться зашифроване повідомлення.

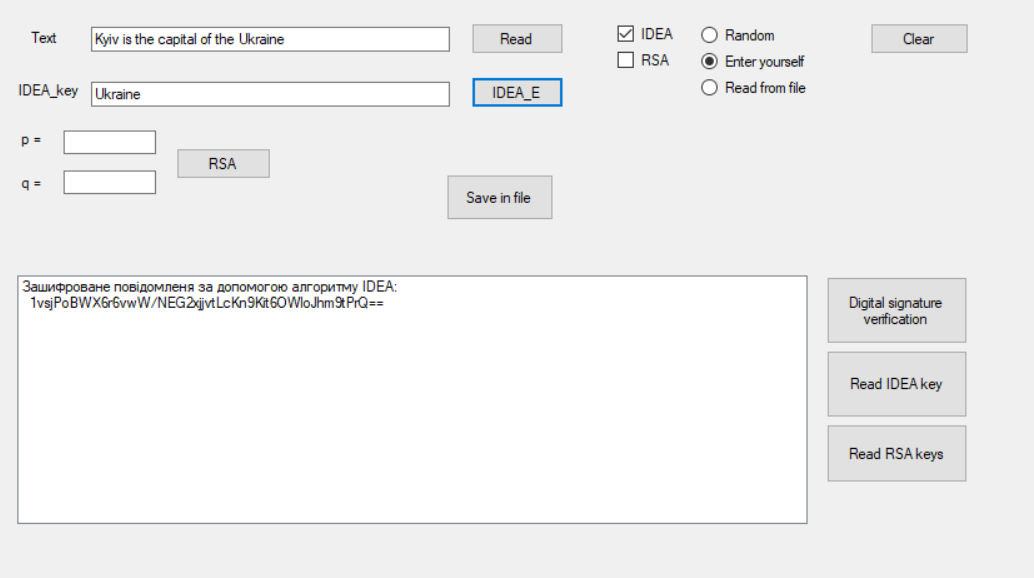


Рисунок 2.6. Результат шифрування алгоритмом IDEA.

Далі нам необхідно створити цифровий підпис. Для цього ми перемикаємо на RSA і також обираємо спосіб введення параметрів p та q. Якщо обрати зчитати, то на рисунку 2.3 видно файл RSA\_keys де знаходяться необхідні ключі. Давайте оберемо третій вид знаходження даних – Random.

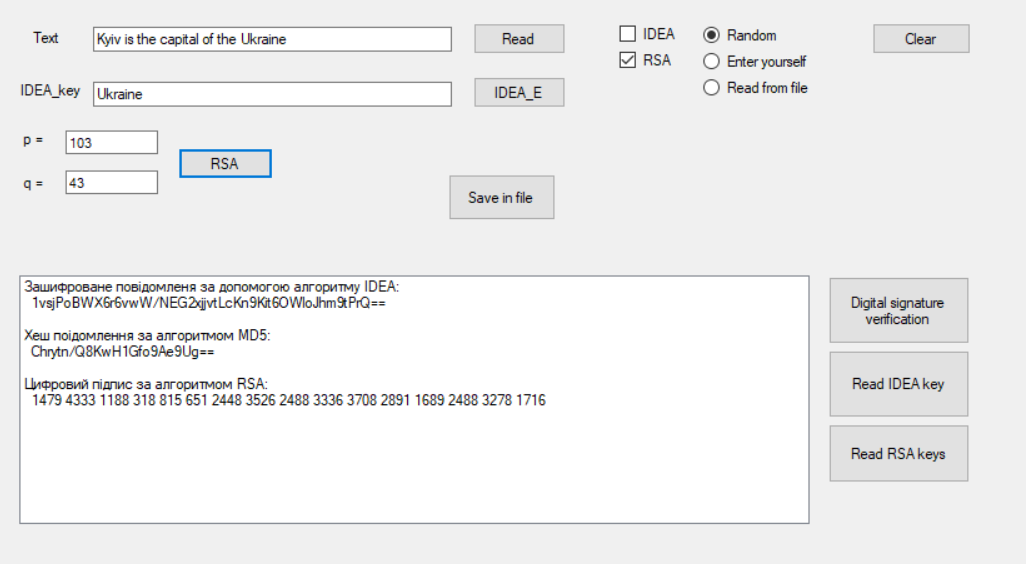


Рисунок 2.7. Результат цифрового підпису алгоритмом RSA.

Як ми можемо побачити програма вивела хеш повідомлення знайденого за допомогою алгоритму MD5. Потім за допомогою цього хешу було створено цифровий підпис за допомогою алгоритму RSA.

Всі необхідні дані для створення цифрового конверта у нас вже є. Нам необхідно зберегти його у якомусь файлі. Для цього натискаємо button Save in file. Після натискання нам необхідно буде створити файл у який буде записано цифровий конверт.

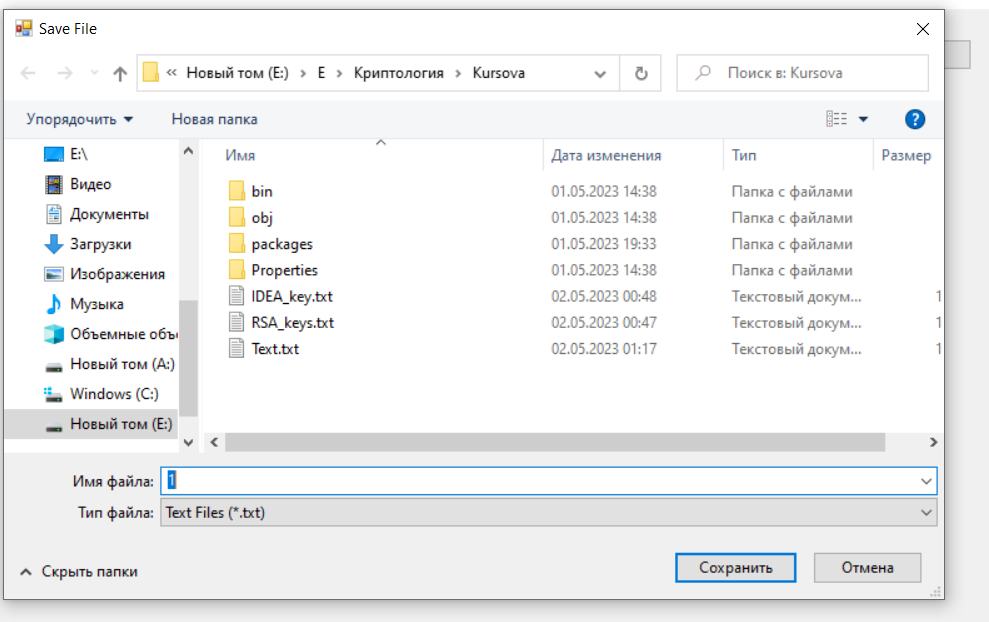


Рисунок 2.8. Створення файлу для цифрового конверта.

Нехай створимо файл 1.txt, тоді програма запише туди цифровий конверт:

Відкривши цей файл ми побачимо:

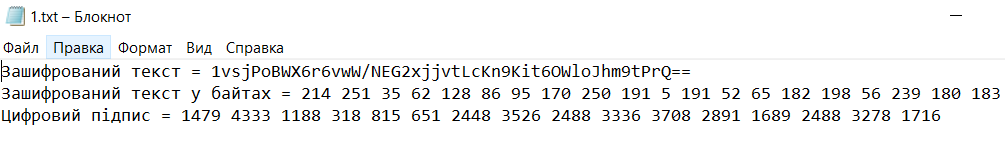


Рисунок 2.9. Цифровий конверт.

Програма записала зашифрований текст, зашифрований текст у байтах та цифровий підпис.

Для перевірки цифрового конверта нам потрібно спочатку зчитати ключі шифрування для алгоритму IDEA та RSA. Після генерації ключів для цих алгоритмів, ці ключі записувалися в окремі файли. Ми можемо перевірити цифровий конверт навіть якщо ми закриємо програму.

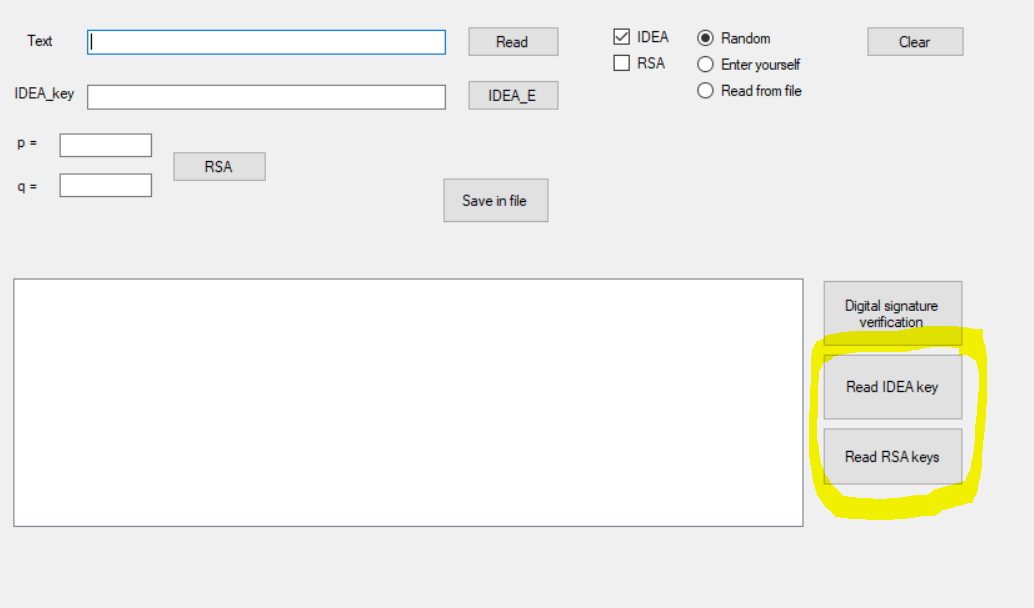


Рисунок 2.10. Завантаження ключів.

На рисунку 2.10 намальовано куди потрібно натискати щоб завантажити ключі. Завантажимо наприклад спочатку ключ для алгоритму IDEA. Для цього натиснемо button Read IDEA key.

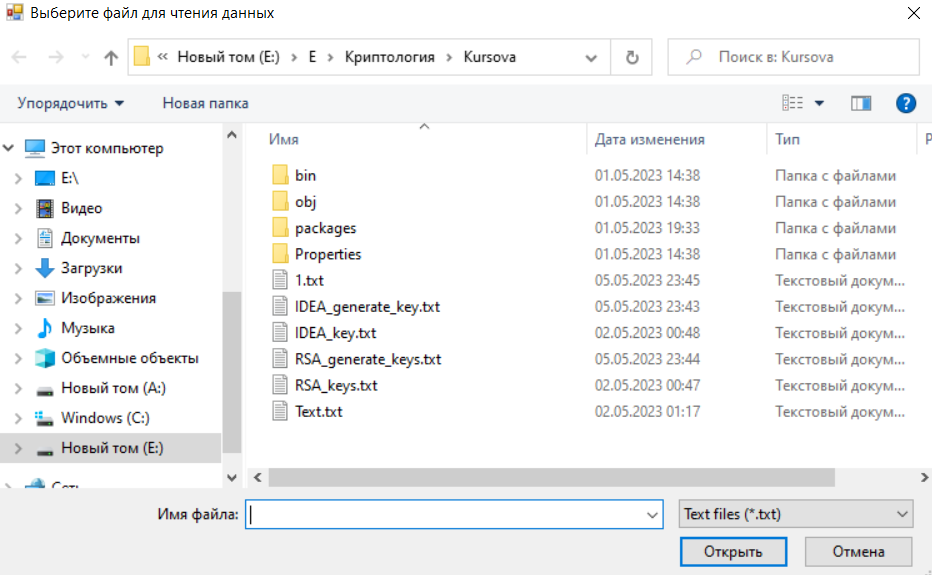


Рисунок 2.11. Зчитування ключа IDEA.

Як ми можемо побачити, після роботи програми у нас додалося три файла.

* 1.txt – цифровий конверт;
* IDEA\_generate\_key – згенерований ключ для алгоритму IDEA під час роботи програми;
* RSA\_generate\_keys – згенеровані ключі для алгоритму RSA під час роботи програми.

Нам потрібно обрати другий файл:

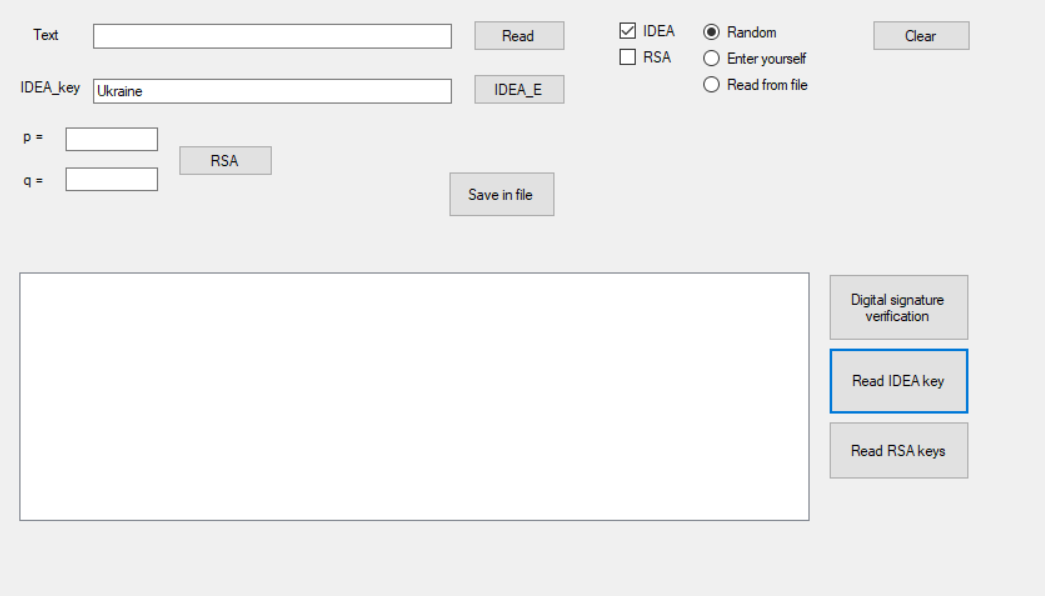


Рисунок 2.12. Результат зчитування ключа IDEA.

Для завантаження ключів RSA необхідно натиснути button Read RSA keys і обрати третій файл.

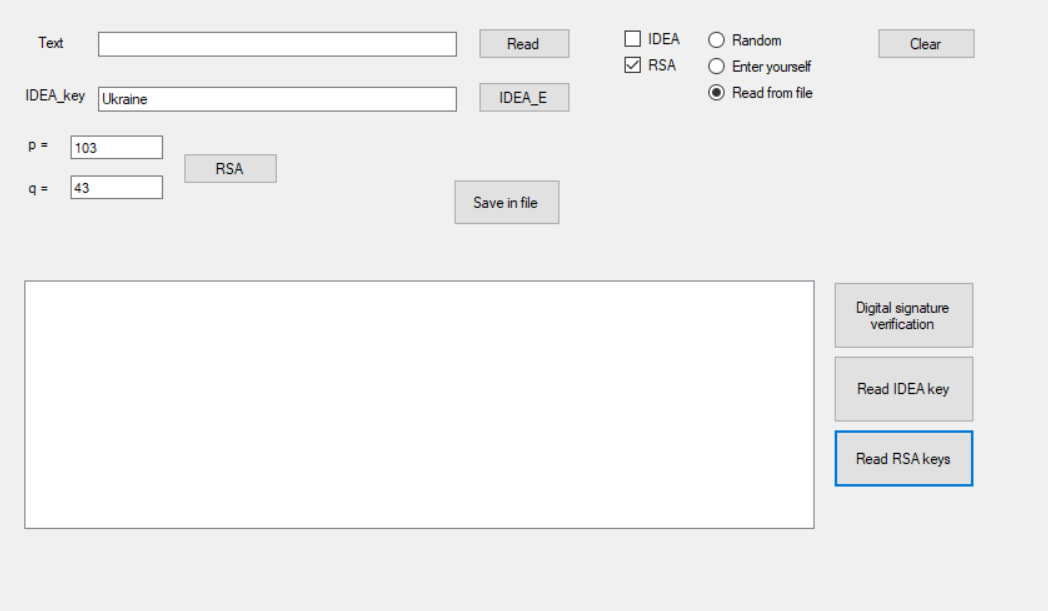


Рисунок 2.13. Результат зчитування ключів RSA.

Після того як всі ключі завантажені, нам необхідно завантажити цифровий конверт. Для цього натискаємо на button Digital signature verification, після чого обираємо перший файл:

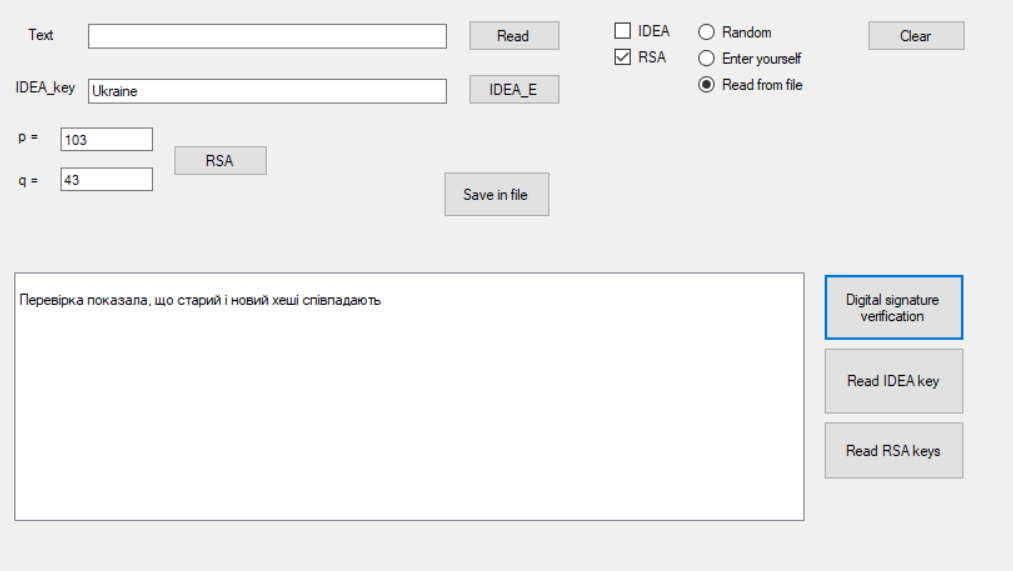


Рисунок 2.14. Результат перевірки цифрового конверта.

Після зчитування цифрового конверта, з’явиться повідомлення, що старий і новий хеші співпадають, це значить, що все було зроблено правильно і цифровий конверт не змінювався.

Хочу продемонструвати, що буде якщо або завантажити не той файл, або щось змінити у новому файлі. Наприклад зчитаємо інший ключ для алгоритму IDEA.

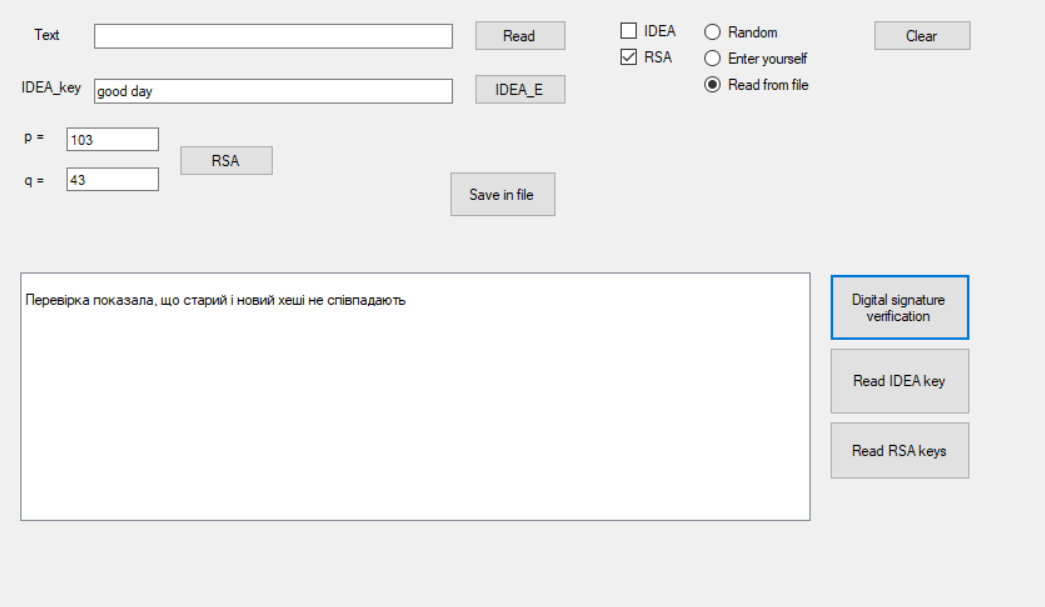


Рисунок 2.15. Не пройдена перевірка.

Як ми можемо побачити на рисунку 2.15 у нас інший ключ для алгоритму IDEA ніж на рисунку 2.12. Зчитавши цифровий конверт, з цим ключем він перевірку не пройшов, тобто відбулися зміни.

Також ми можемо побачити, що у вікні є button Clear, якщо на неї натиснути то все, що написано у listbox очиститься.

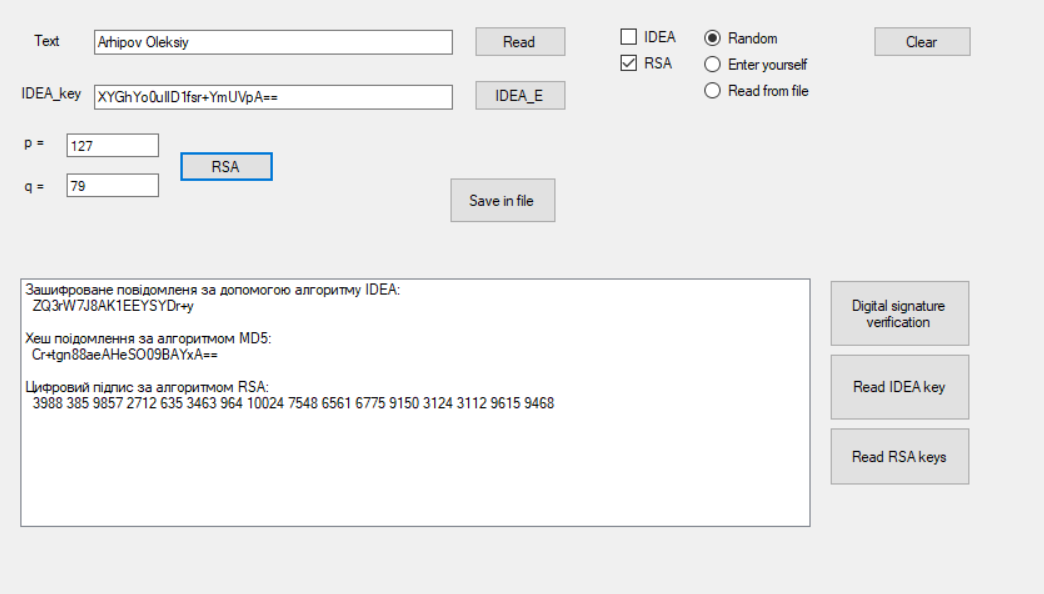


Рисунок 2.16. Інший приклад цифрового конверта.

Я створив цифровий конверт за допомогою алгоритмів IDEA, MD5, RSA. Наведені результати роботи програми показали, що програма створює цифровий конверт та зберігає усі необхідні дані в окремому файлі і навіть після зачинення програми, за її допомогою можна буде потім провести верифікацію цифрового конверту.

**2.2 UML – діаграми**

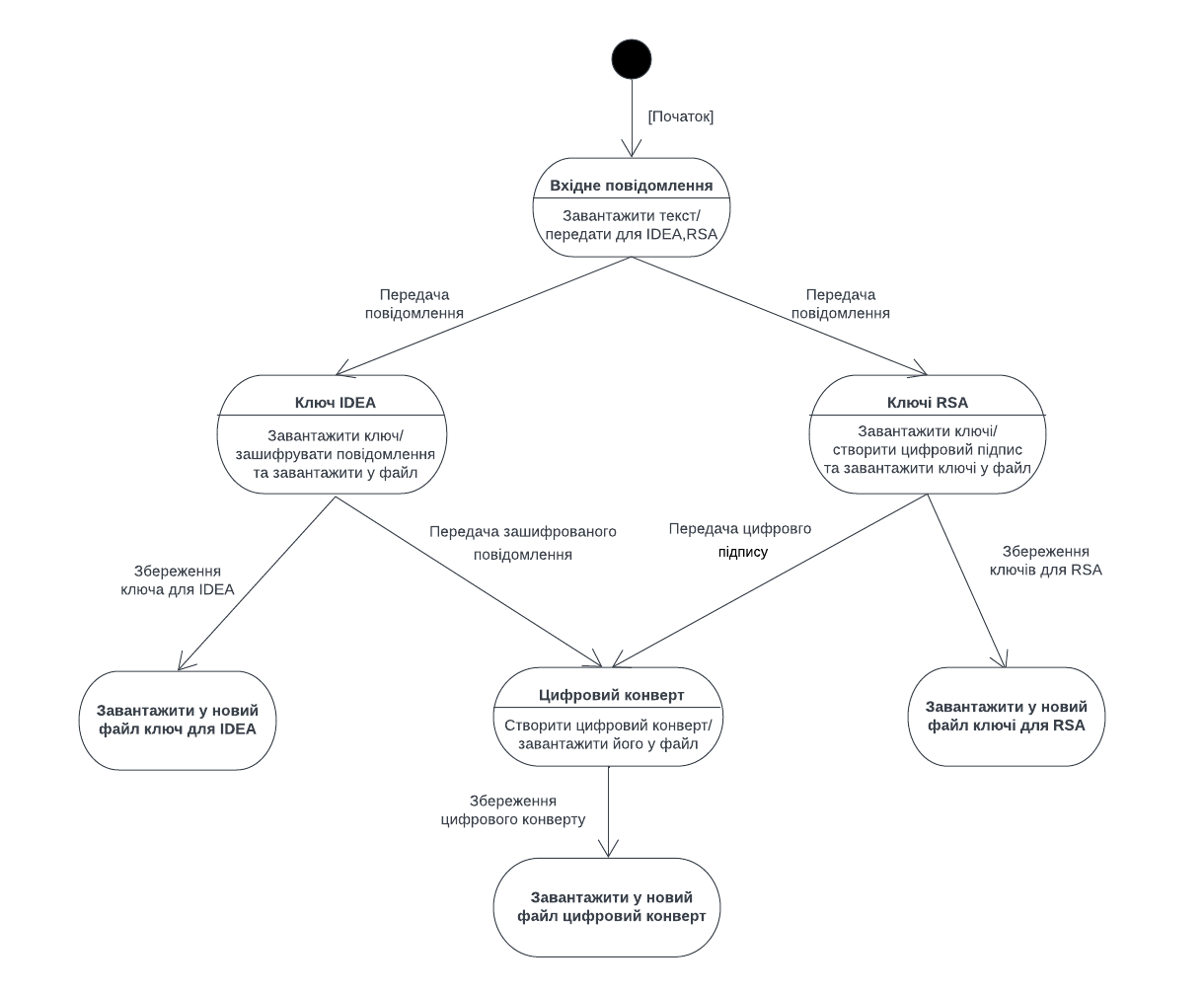


Рисунок 2.18. UML діаграма станів для створення цифрового конверту.

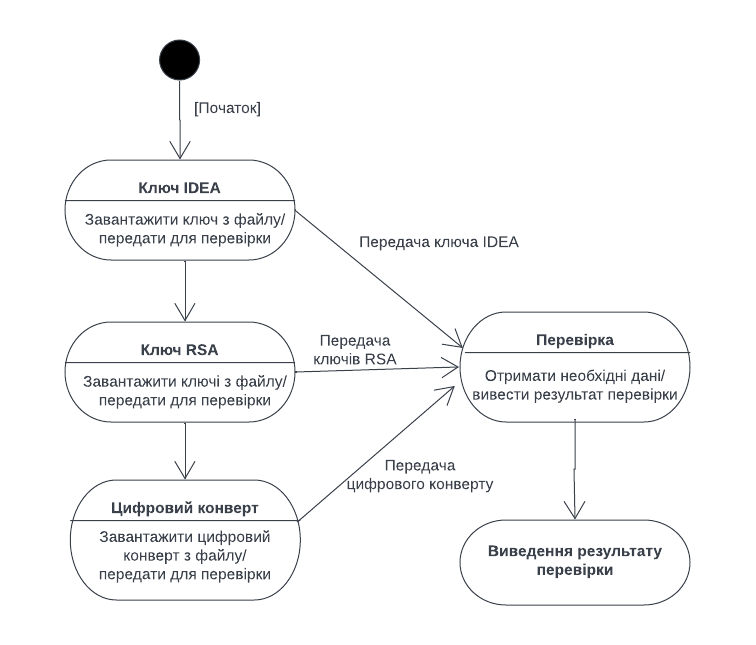


Рисунок 2.19. UML діаграма станів для перевірки цифрового конверту.

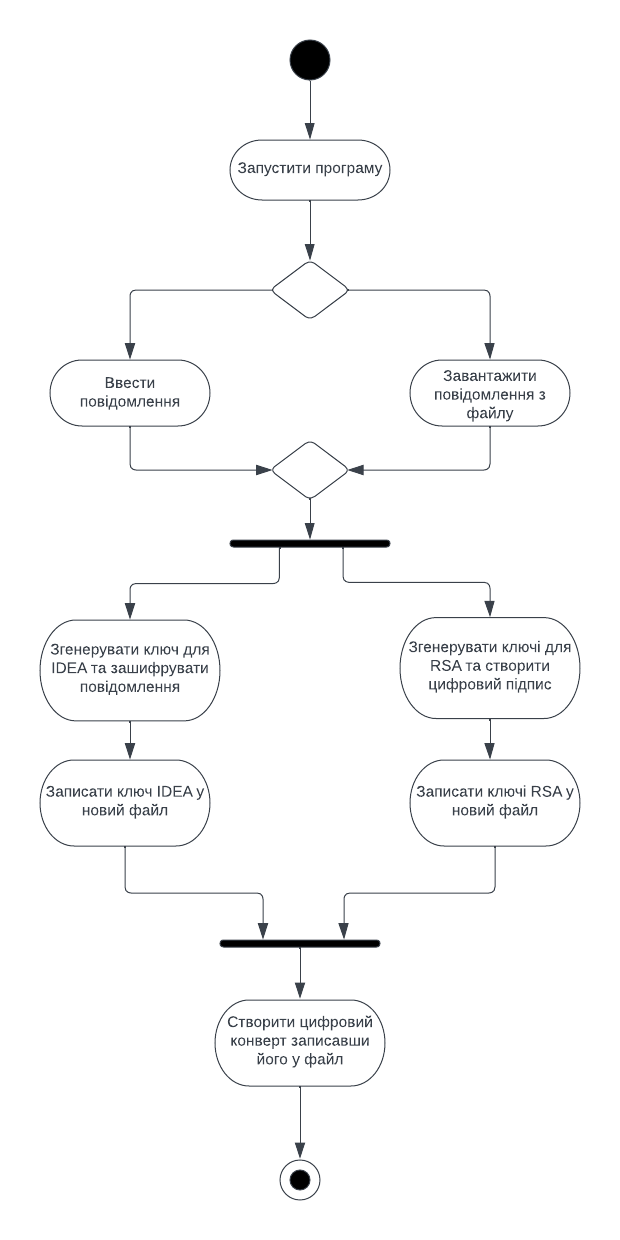


Рисунок 2.20. UML діаграма діяльності для створення цифрового конверту.

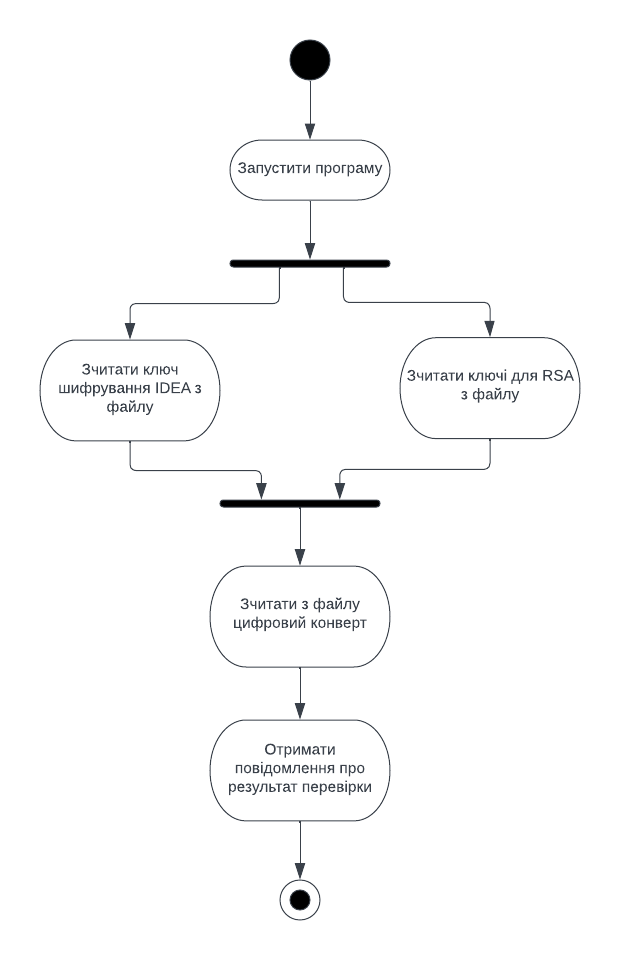


Рисунок 2.21. UML діаграма діяльності для перевірки цифрового конверту.

**Висновки**

1. Реалізовано програмно цифровий конверт за допомогою алгоритму симетричного блокового шифрування IDEA, функції хешування MD5 та алгоритму асиметричного шифрування RSA.
2. На вимогу користувача відбувається генерація криптографічних

ключів (пара – відкритий/закритий ключі для асиметричного шифрування та цифрового підпису й таємний ключ симетричного шифрування). Також повинна бути реалізовано можливість використання ключів, що були сгенеровані раніше з певного файлу або сертифікованого сховища (користувач вказує шлях).

1. Реалізовано можливість завантаження файлу електронних даних.
2. Розраховано значення хеш-функції повідомлення згідно з вказаним алгоритмом.
3. Реалізовано процедуру цифрового підпису, використовуючи закритий

ключ користувача та алгоритм цифрового підпису згідно з варіантом.

1. Проведено направлене шифрування даних симетричним криптоалгоритмом з використанням таємного ключа користувача.
2. Зроблено експорт отриманого цифрового конверту(збереження у файл).
3. Реалізовано можливість верифікації цифрового підпису повідомлення.
4. Обраний алгоритм симетричного блокового шифрування IDEA є ефективним та безпечним алгоритмом шифрування, який може бути використаний у різних сферах та додатках.
5. Результати роботи показали, що для забезпечення максимального захисту від атак, потрібно обирати досить довгий ключ.
6. Обраний алгоритм хешування MD5 забезпечує швидке та ефективну хешування даних, але він не є абсолютно стійким до атак і може бути зламаний за допомогою колізій.
7. Обраний алгоритм асиметричного шифрування RSA один із безпечніших алгоритм асиметричного шифрування.
8. Результати тестування програми показали, що при використанні поганих ключів для RSA (маленьких чисел), алгоритм RSA стає менш захищеним до атак.
9. Обрані алгоритми для створення цифрового конверту добре підходять для цієї задачі, проте при використанні поганих ключів для цих алгоритмів, вони стають менш захищеними до атак і наслідком цього буде витік секретних даних.

**Списки використаної літератури**

1. Vanstone S. A., Menezes A. J., Oorschot P. C. v. Handbook of Applied Cryptography. Taylor & Francis Group, 2018. 810 p.
2. Stallings W. Cryptography and Network Security: Principles and Practice. Pearson Education, Limited, 2010. 744 p.
3. Учасники проектів Вікімедіа. IDEA (шифр) – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/IDEA\_(шифр).
4. Прикладна криптологія : системи шифрування : підручник / О. Г. Корченко, В. П. Сіденко, Ю. О. Дрейс. – К. : ДУТ, 2014. – 448 с.:іл.
5. Криптологія: навч.-метод. посіб. / Людмила Ярославівна Глинчук – Луцьк: Вежа-Друк, 2014. – 164 с.
6. Алгоритм md5. *StudFiles*.URL: https://studfile.net/preview/9468983/page:17/.
7. Учасники проектів Вікімедіа. MD5 – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/MD5.
8. Учасники проектів Вікімедіа. RSA – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/RSA.
9. Schneier B. Applied Cryptography: Protocols, Algorithms and Source Code in C. Wiley & Sons, Incorporated, John, 2015. 784 p.
10. Mollin R. A. RSA and Public-Key Cryptography. Chapman & Hall/CRC, 2002. 304 p.
11. Ferguson N., Schneier B., Kohno T. Cryptography Engineering: Design Principles and Practical Applications. Wiley & Sons, Limited, John, 2015. 384 p.
12. Stinson D. R., Paterson M. Cryptography: Theory and Practice. Taylor & Francis Group, 2018. 580 p.
13. Paar C. Understanding cryptography: A textbook for students and practitioners. Heidelberg : Springer, 2010. 372 p.
14. Serious Cryptography: A Practical Introduction to Modern Encryption. No Starch Press, 2017. 312р.
15. Applied cryptography for cyber security and defense: Information encryption and cyphering / ed. by N. H. R. 1958-, Yang, Li, 1974 Oct. 29-. Hershey, PA : Information Science Reference, 2010.
16. Singh S. The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. Anchor, 2000. 432 p.
17. Bose R. Information theory, coding and cryptography. McGraw-Hill, 2003. 277 p.
18. Batten L. M. Public Key Cryptography: Applications and Attacks. Wiley & Sons, Incorporated, John, 2013. 224 p.
19. Baldoni M. W., Ciliberto C., Cattaneo G. M. P. Elementary Number Theory, Cryptography and Codes. Berlin, Heidelberg : Springer Berlin Heidelberg, 2009. 522р.
20. Cusick T. W. Stream ciphers and number theory. Amsterdam : Elsevier, 1998. 431 p.

**Додаток А**

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Security.Cryptography;

using Org.BouncyCastle.Crypto;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using Org.BouncyCastle.Crypto.Parameters;

using Org.BouncyCastle.Crypto.Modes;

using Org.BouncyCastle.Crypto.Engines;

using Org.BouncyCastle.Utilities;

using static System.Net.Mime.MediaTypeNames;

using System.Runtime.Remoting.Lifetime;

namespace Kursova

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

radioButton1.Checked = true;

checkBox1.Checked = true;

}

byte[] symmetricKey, IDEA\_byte;

List<int> m2, symmetricKey\_e;

int n, e\_rsa, d;

string Idea\_e = "", Rsa\_e = "";

byte[] BBS()

{

List<int> B = new List<int> ();

listBox1.Items.Clear();

int n = 1000;

List<int> x = new List<int>();

for (var i = 0u; i < n; i++)

{

if (IsPrimeNumber(i))

{

if(Convert.ToInt32(i)%4 == 3) x.Add(Convert.ToInt32(i));

}

}

Random r = new Random();

int k = r.Next(27, x.Count);

int k1 = r.Next(27, x.Count);

int p = x[k];

int q = x[k1];

int s = 0;

do

{

s = r.Next(q, 300000);

bool t = AreRelativelyPrime(s, p);

bool t1 = AreRelativelyPrime(s, q);

if (t == true && t1 == true) break;

} while (true);

int n1 = 128;

List<double> X = new List<double>();

double N = p \* q;

X.Add((Math.Pow(s, 2)) % N);

for (int i = 0; i < n1; i++)

{

int c1 = (int)DateTime.Now.Millisecond;

X.Add((Math.Pow(X[i], 2) + Math.Pow(c1, 2)) % N);

B.Add(Convert.ToInt32(X[i] % 2));

}

byte[] bytes = new byte[16];

for (int i = 0; i < 16; i++)

{

byte b = 0;

for (int j = 0; j < 8; j++)

{

b <<= 1;

b |= (byte)B[i \* 8 + j];

}

bytes[i] = b;

}

return bytes;

}

byte[] IDEAEncrypt(byte[] key, byte[] input)

{

BufferedBlockCipher cipher = new BufferedBlockCipher(new CfbBlockCipher(new IdeaEngine(), 64));

KeyParameter keyParam = new KeyParameter(key);

cipher.Init(true, new ParametersWithIV(keyParam, new byte[8]));

byte[] output = new byte[input.Length];

int outputLength = cipher.ProcessBytes(input, 0, input.Length, output, 0);

cipher.DoFinal(output, outputLength);

return output;

}

byte[] IDEADecrypt(byte[] key, byte[] input)

{

BufferedBlockCipher cipher = new BufferedBlockCipher(new CfbBlockCipher(new IdeaEngine(), 64));

KeyParameter keyParam = new KeyParameter(key);

cipher.Init(false, new ParametersWithIV(keyParam, new byte[8]));

byte[] output = new byte[input.Length];

int outputLength = cipher.ProcessBytes(input, 0, input.Length, output, 0);

cipher.DoFinal(output, outputLength);

return output;

}

byte[] MD5(byte[] t1)

{

byte[] text = t1;

List<byte> list;

list = text.ToList();

if (text.Length % 64 != 0)

{

list.Add(0x80);

if (list.Count % 64 != 0)

{

do

{

list.Add(0);

} while (list.Count % 64 != 0);

}

text = list.ToArray();

}

uint A = 0x67452301;

uint B = 0xefcdab89;

uint C = 0x98badcfe;

uint D = 0x10325476;

for (int i = 0; i < text.Length; i += 64)

{

uint[] X = new uint[16];

for (int j = 0; j < 16; j++)

{

X[j] = BitConverter.ToUInt32(text, i \* 64 + j \* 4);

}

uint A1 = A, B1 = B, C1 = C, D1 = D;

for (int j = 0; j < 64; j++)

{

A = B + ((A + G(B, C, D, j) + X[g(j)] + T(j)) << S(i));

uint t = D;

D = C; ;

C = A;

B = t;

}

A = A + A1;

B = B + B1;

C = C + C1;

D = D + D1;

}

byte[] a = BitConverter.GetBytes(A);

byte[] b = BitConverter.GetBytes(B);

byte[] c = BitConverter.GetBytes(C);

byte[] d = BitConverter.GetBytes(D);

List<byte[]> W = new List<byte[]>();

W.Add(a);

W.Add(b);

W.Add(c);

W.Add(d);

//string res = "";

byte[] r = new byte[a.Length + b.Length + c.Length + d.Length];

int c1 = 0;

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < a.Length; j++)

{

//res += W[i][j].ToString("x2");

r[c1] = W[i][j];

c1++;

}

}

return r;

}

uint T(int i)

{

uint t = (uint)(Math.Pow(2, 32) \* Math.Abs(Math.Sin(i + 1)));

return t;

}

int g(int i)

{

int g1 = 0;

if (i < 16)

{

g1 = i;

}

else if (i > 15 && i < 32)

{

g1 = (5 \* i + 1) % 16;

}

else if (i > 31 && i < 48)

{

g1 = (3 \* i + 5) % 16;

}

else if (i > 47 && i < 64)

{

g1 = (7 \* i) % 16;

}

return g1;

}

uint G(uint b, uint c, uint d, int i)

{

uint g = 0;

if (i < 16)

{

g = (b & c) | (~b & d);

}

else if (i > 15 && i < 32)

{

g = (b & d) | (c & ~d);

}

else if (i > 31 && i < 48)

{

g = b ^ c ^ d;

}

else if (i > 47 && i < 64)

{

g = c ^ (b | ~d);

}

return g;

}

int S(int i)

{

int[] s = new int[] { 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22 };

return s[i % 16];

}

void Read()

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.InitialDirectory = "E:\\Е\\Криптология\\Kursova";

openFileDialog1.Title = "Выберите файл для чтения данных";

openFileDialog1.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

DialogResult result = openFileDialog1.ShowDialog();

string filename = openFileDialog1.FileName;

string text = File.ReadAllText(filename);

tbText.Text = text;

}

void Read\_parameters()

{

if(checkBox2.Checked && radioButton3.Checked)

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.InitialDirectory = "E:\\Е\\Криптология\\Kursova";

openFileDialog1.Title = "Выберите файл для чтения данных";

openFileDialog1.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

DialogResult result = openFileDialog1.ShowDialog();

string filename = openFileDialog1.FileName;

string[] lines = File.ReadAllLines(filename);

string input = lines[0];

string[] parts = input.Split('=');

string numberString = parts[1].Trim();

int p = int.Parse(numberString);

input = lines[1];

parts = input.Split('=');

numberString = parts[1].Trim();

int q = int.Parse(numberString);

input = lines[2];

parts = input.Split('=');

numberString = parts[1].Trim();

d = int.Parse(numberString);

input = lines[3];

parts = input.Split('=');

numberString = parts[1].Trim();

e\_rsa = int.Parse(numberString);

tbp.Text = p.ToString();

tbq.Text = q.ToString();

n = p \* q;

}

else if (checkBox1.Checked && radioButton3.Checked)

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.InitialDirectory = "E:\\Е\\Криптология\\Kursova";

openFileDialog1.Title = "Выберите файл для чтения данных";

openFileDialog1.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

DialogResult result = openFileDialog1.ShowDialog();

string filename = openFileDialog1.FileName;

string text = File.ReadAllText(filename);

tbSymkey.Text = text;

symmetricKey = Encoding.UTF8.GetBytes(text);

}

}

private void btnRead\_Click(object sender, EventArgs e)

{

Read();

}

private void btnIDEA\_E\_Click(object sender, EventArgs e)

{

if (checkBox2.Checked)

{

checkBox2.Checked = false;

checkBox1.Checked = true;

}

string key1 = "";

if (checkBox1.Checked && radioButton2.Checked)

{

key1 = tbSymkey.Text;

}

else if (checkBox1.Checked && radioButton1.Checked)

{

symmetricKey = BBS();

tbSymkey.Text = Convert.ToBase64String(symmetricKey);

key1 = tbSymkey.Text;

}

else if (checkBox1.Checked && radioButton3.Checked)

{

Read\_parameters();

}

string text1 = tbText.Text;

byte[] text = Encoding.UTF8.GetBytes(text1);

byte[] key = Encoding.UTF8.GetBytes(key1);

symmetricKey = key;

byte[] res = IDEAEncrypt(key, text);

IDEA\_byte = res;

Idea\_e = Convert.ToBase64String(res);

listBox1.Items.Add("Зашифроване повідомленя за допомогою алгоритму IDEA:");

listBox1.Items.Add(" " + Idea\_e);

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Text Files (\*.txt)|\*.txt|All Files (\*.\*)|\*.\*";

saveFileDialog.Title = "Save File";

saveFileDialog.FileName = "E:\\Е\\Криптология\\Kursova\\IDEA\_generate\_key.txt";

string filename = saveFileDialog.FileName;

using (StreamWriter writer = new StreamWriter(filename))

{

writer.WriteLine(tbSymkey.Text);

}

}

bool IsPrimeNumber(uint n)

{

var result = true;

if (n > 1)

{

for (var i = 2u; i < n; i++)

{

if (n % i == 0)

{

result = false;

break;

}

}

}

else

{

result = false;

}

return result;

}

int ModPow(int a, int b, int m)

{

int res = 1;

a %= m;

while (b > 0)

{

if ((b & 1) != 0)

res = (res \* a) % m;

a = (a \* a) % m;

b >>= 1;

}

if (res < 0)

res += m;

return res;

}

void Generate()

{

int n = 500;

List<int> x = new List<int>();

for (var i = 0u; i < n; i++)

{

if (IsPrimeNumber(i))

{

x.Add(Convert.ToInt32(i));

}

}

Random r = new Random();

int j, k = 1;

do

{

j = r.Next(x.Count);

} while (j < 3);

if (x[j] > 100)

{

do

{

k = r.Next(x.Count);

} while (x[k] > 100);

}

else

{

do

{

k = r.Next(x.Count);

} while (x[k] < 100);

}

tbp.Text = x[j].ToString();

tbq.Text = x[k].ToString();

}

void Generate2()

{

Random r = new Random();

int p = Convert.ToInt32(tbp.Text);

int q = Convert.ToInt32(tbq.Text);

n = p \* q;

int f = (p - 1) \* (q - 1);

List<int> e1 = new List<int>();

List<int> x = new List<int>();

bool t;

for (var i = 0u; i < f; i++)

{

if (IsPrimeNumber(i))

{

x.Add(Convert.ToInt32(i));

t = AreRelativelyPrime(x[x.Count - 1], f);

if (t == true) e1.Add(x[x.Count - 1]);

if (e1.Count == 10) break;

}

}

int e2 = r.Next(0, e1.Count - 1);

e\_rsa = e1[e2];

d = FindD(p, q, e\_rsa);

}

bool AreRelativelyPrime(int a, int b)

{

int gcd = 1;

for (int i = 1; i <= a && i <= b; ++i)

{

if (a % i == 0 && b % i == 0)

gcd = i;

}

return gcd == 1;

}

int FindD(int p, int q, int e)

{

int phi = (p - 1) \* (q - 1);

int d = 0;

int k = 1;

while (true)

{

d = (k \* phi + 1) / e;

if (d \* e == k \* phi + 1)

return d;

k++;

}

}

void Calc()

{

if(checkBox1.Checked)

{

checkBox1.Checked = false;

checkBox2.Checked = true;

}

if (checkBox2.Checked && radioButton1.Checked)

{

Generate();

Generate2();

}

else if(checkBox2.Checked && radioButton2.Checked)

{

Generate2();

}

else if (checkBox2.Checked && radioButton3.Checked)

{

Read\_parameters();

}

string text = tbText.Text;

byte[] hashBytes = MD5(Encoding.UTF8.GetBytes(text));

listBox1.Items.Add("");

listBox1.Items.Add("Хеш поідомлення за алгоритмом MD5:");

listBox1.Items.Add(" " + Convert.ToBase64String(hashBytes));

m2 = Encrypt\_RSA(hashBytes, d, n);

string res = "";

for(int i = 0; i < m2.Count; i++)

{

if (i != m2.Count - 1) res += m2[i].ToString() + " ";

else res += m2[i].ToString();

}

Rsa\_e = res;

listBox1.Items.Add("");

listBox1.Items.Add("Цифровий підпис за алгоритмом RSA:");

listBox1.Items.Add(" " + res);

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Text Files (\*.txt)|\*.txt|All Files (\*.\*)|\*.\*";

saveFileDialog.Title = "Save File";

saveFileDialog.FileName = "E:\\Е\\Криптология\\Kursova\\RSA\_generate\_keys.txt";

string filename = saveFileDialog.FileName;

using (StreamWriter writer = new StreamWriter(filename))

{

writer.WriteLine("p = " + tbp.Text);

writer.WriteLine("q = " + tbq.Text);

writer.WriteLine("d = " + d.ToString());

writer.WriteLine("e = " + e\_rsa.ToString());

}

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox1.Checked) checkBox2.Checked = false;

if (checkBox1.Checked == false) checkBox2.Checked = true;

}

private void checkBox2\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox2.Checked) checkBox1.Checked = false;

if (checkBox2.Checked == false) checkBox1.Checked = true;

}

List<int> Encrypt\_RSA(byte[] t1, int d1, int n1)

{

byte[] text = t1;

List<int> c = new List<int>();

for (int i = 0; i < text.Length; i++)

{

c.Add(ModPow(text[i], d1, n1));

}

return c;

}

private void btnClear\_Click(object sender, EventArgs e)

{

listBox1.Items.Clear();

}

private void btnReadIdeaKey\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.InitialDirectory = "E:\\Е\\Криптология\\Kursova";

openFileDialog1.Title = "Выберите файл для чтения данных";

openFileDialog1.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

DialogResult result = openFileDialog1.ShowDialog();

string filename = openFileDialog1.FileName;

string[] lines = File.ReadAllLines(filename);

symmetricKey = Encoding.UTF8.GetBytes(lines[0]);

tbSymkey.Text = lines[0];

}

private void btnReadRSAkeys\_Click(object sender, EventArgs e)

{

checkBox2.Checked = true;

radioButton3.Checked = true;

Read\_parameters();

}

private void btnPerevirka\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.InitialDirectory = "E:\\Е\\Криптология\\Kursova";

openFileDialog1.Title = "Выберите файл для чтения данных";

openFileDialog1.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

DialogResult result = openFileDialog1.ShowDialog();

string filename = openFileDialog1.FileName;

string[] lines = File.ReadAllLines(filename);

string input = lines[1];

string[] parts = input.Split('=');

string numberString = parts[1].Trim();

string[] t = numberString.Split(' ');

List<byte> text\_byte = new List<byte>();

for(int i = 0; i < t.Length; i++)

{

text\_byte.Add(Convert.ToByte(t[i]));

}

input = lines[2];

parts = input.Split('=');

numberString = parts[1].Trim();

string[] pid = numberString.Split(' ');

List<int> pidpus = new List<int>();

for (int i = 0; i < pid.Length; i++)

{

pidpus.Add(Convert.ToInt32(pid[i]));

}

byte[] t1 = IDEADecrypt(symmetricKey, text\_byte.ToArray());

string text = Encoding.UTF8.GetString(t1);

byte[] hashBytes = MD5(Encoding.UTF8.GetBytes(text));

string hash = Convert.ToBase64String(hashBytes);

string hash2 = Convert.ToBase64String(Decrypt\_RSA(pidpus, e\_rsa, n));

listBox1.Items.Add("");

if (hash == hash2)

{

listBox1.Items.Add("Верифікація пройшла успішно");

listBox1.Items.Add("Перевірка показала, що старий і новий хеші співпадають");

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Text Files (\*.txt)|\*.txt|All Files (\*.\*)|\*.\*";

saveFileDialog.Title = "Save File";

saveFileDialog.FileName = "E:\\Е\\Криптология\\Kursova\\Text\_result.txt";

string filename1 = saveFileDialog.FileName;

using (StreamWriter writer = new StreamWriter(filename1))

{

writer.WriteLine(text);

}

}

else

{

listBox1.Items.Add("Верифікація провалилася");

listBox1.Items.Add("Перевірка показала, що старий і новий хеші не співпадають");

}

}

private void btnSave\_Click(object sender, EventArgs e)

{

Save\_in\_file();

}

private void btnRSA\_Click(object sender, EventArgs e)

{

Calc();

}

byte[] Decrypt\_RSA(List<int> t1, int e1, int n1)

{

byte[] m2 = new byte[t1.Count];

int temp;

for (int i = 0; i < t1.Count; i++)

{

temp = ModPow(t1[i], e1, n1);

if(temp > 255)

{

byte[] bytes = new byte[temp];

return bytes;

}

m2[i] = Convert.ToByte(temp);

}

return m2;

}

void Save\_in\_file()

{

string res1 = "";

for (int i = 0; i < IDEA\_byte.Length; i++)

{

if (i != IDEA\_byte.Length - 1) res1 += IDEA\_byte[i].ToString() + " ";

else res1 += IDEA\_byte[i].ToString();

}

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Text Files (\*.txt)|\*.txt|All Files (\*.\*)|\*.\*";

saveFileDialog.Title = "Save File";

saveFileDialog.ShowDialog();

if (saveFileDialog.FileName != "")

{

using (StreamWriter writer = new StreamWriter(saveFileDialog.OpenFile()))

{

writer.Dispose();

writer.Close();

}

}

string filename = saveFileDialog.FileName;

using (StreamWriter writer = new StreamWriter(filename))

{

writer.WriteLine("Зашифрований текст = " + Idea\_e);

writer.WriteLine("Зашифрований текст у байтах = " + res1);

writer.WriteLine("Цифровий підпис = " + Rsa\_e);

}

}

}

}

Додаток Б

Файл Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kursova

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Додаток В

Файл Form1.Designer.cs

namespace Kursova

{

partial class Form1

{

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

private void InitializeComponent()

{

this.tbSymkey = new System.Windows.Forms.TextBox();

this.btnRead = new System.Windows.Forms.Button();

this.saveFileDialog1 = new System.Windows.Forms.SaveFileDialog();

this.tbText = new System.Windows.Forms.TextBox();

this.label1 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.listBox1 = new System.Windows.Forms.ListBox();

this.btnIDEA\_E = new System.Windows.Forms.Button();

this.btnRSA = new System.Windows.Forms.Button();

this.label3 = new System.Windows.Forms.Label();

this.label4 = new System.Windows.Forms.Label();

this.tbq = new System.Windows.Forms.TextBox();

this.tbp = new System.Windows.Forms.TextBox();

this.radioButton1 = new System.Windows.Forms.RadioButton();

this.radioButton2 = new System.Windows.Forms.RadioButton();

this.radioButton3 = new System.Windows.Forms.RadioButton();

this.checkBox1 = new System.Windows.Forms.CheckBox();

this.checkBox2 = new System.Windows.Forms.CheckBox();

this.btnSave = new System.Windows.Forms.Button();

this.btnPerevirka = new System.Windows.Forms.Button();

this.btnClear = new System.Windows.Forms.Button();

this.btnReadIdeaKey = new System.Windows.Forms.Button();

this.btnReadRSAkeys = new System.Windows.Forms.Button();

this.SuspendLayout();

//

// tbSymkey

//

this.tbSymkey.Location = new System.Drawing.Point(106, 87);

this.tbSymkey.Name = "tbSymkey";

this.tbSymkey.Size = new System.Drawing.Size(381, 22);

this.tbSymkey.TabIndex = 0;

//

// btnRead

//

this.btnRead.Location = new System.Drawing.Point(512, 29);

this.btnRead.Name = "btnRead";

this.btnRead.Size = new System.Drawing.Size(99, 31);

this.btnRead.TabIndex = 2;

this.btnRead.Text = "Read";

this.btnRead.UseVisualStyleBackColor = true;

this.btnRead.Click += new System.EventHandler(this.btnRead\_Click);

//

// tbText

//

this.tbText.Location = new System.Drawing.Point(106, 33);

this.tbText.Name = "tbText";

this.tbText.Size = new System.Drawing.Size(381, 22);

this.tbText.TabIndex = 3;

this.tbText.Text = "Arhipov Oleksiy";

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(25, 87);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(67, 16);

this.label1.TabIndex = 4;

this.label1.Text = "IDEA\_key";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(39, 36);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(33, 16);

this.label2.TabIndex = 5;

this.label2.Text = "Text";

//

// listBox1

//

this.listBox1.FormattingEnabled = true;

this.listBox1.ItemHeight = 16;

this.listBox1.Location = new System.Drawing.Point(28, 278);

this.listBox1.Name = "listBox1";

this.listBox1.Size = new System.Drawing.Size(841, 244);

this.listBox1.TabIndex = 6;

//

// btnIDEA\_E

//

this.btnIDEA\_E.Location = new System.Drawing.Point(512, 83);

this.btnIDEA\_E.Name = "btnIDEA\_E";

this.btnIDEA\_E.Size = new System.Drawing.Size(99, 31);

this.btnIDEA\_E.TabIndex = 7;

this.btnIDEA\_E.Text = "IDEA\_E";

this.btnIDEA\_E.UseVisualStyleBackColor = true;

this.btnIDEA\_E.Click += new System.EventHandler(this.btnIDEA\_E\_Click);

//

// btnRSA

//

this.btnRSA.Location = new System.Drawing.Point(197, 153);

this.btnRSA.Name = "btnRSA";

this.btnRSA.Size = new System.Drawing.Size(102, 31);

this.btnRSA.TabIndex = 8;

this.btnRSA.Text = "RSA";

this.btnRSA.UseVisualStyleBackColor = true;

this.btnRSA.Click += new System.EventHandler(this.btnRSA\_Click);

//

// label3

//

this.label3.AutoSize = true;

this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 7.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label3.Location = new System.Drawing.Point(29, 178);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(25, 16);

this.label3.TabIndex = 12;

this.label3.Text = "q =";

//

// label4

//

this.label4.AutoSize = true;

this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 7.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label4.Location = new System.Drawing.Point(29, 136);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(25, 16);

this.label4.TabIndex = 11;

this.label4.Text = "p =";

//

// tbq

//

this.tbq.Font = new System.Drawing.Font("Microsoft Sans Serif", 7.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.tbq.Location = new System.Drawing.Point(77, 175);

this.tbq.Name = "tbq";

this.tbq.Size = new System.Drawing.Size(97, 22);

this.tbq.TabIndex = 10;

//

// tbp

//

this.tbp.Font = new System.Drawing.Font("Microsoft Sans Serif", 7.8F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.tbp.Location = new System.Drawing.Point(77, 136);

this.tbp.Name = "tbp";

this.tbp.Size = new System.Drawing.Size(97, 22);

this.tbp.TabIndex = 9;

//

// radioButton1

//

this.radioButton1.AutoSize = true;

this.radioButton1.Location = new System.Drawing.Point(758, 31);

this.radioButton1.Name = "radioButton1";

this.radioButton1.Size = new System.Drawing.Size(80, 20);

this.radioButton1.TabIndex = 13;

this.radioButton1.TabStop = true;

this.radioButton1.Text = "Random";

this.radioButton1.UseVisualStyleBackColor = true;

//

// radioButton2

//

this.radioButton2.AutoSize = true;

this.radioButton2.Location = new System.Drawing.Point(758, 57);

this.radioButton2.Name = "radioButton2";

this.radioButton2.Size = new System.Drawing.Size(109, 20);

this.radioButton2.TabIndex = 14;

this.radioButton2.TabStop = true;

this.radioButton2.Text = "Enter yourself";

this.radioButton2.UseVisualStyleBackColor = true;

//

// radioButton3

//

this.radioButton3.AutoSize = true;

this.radioButton3.Location = new System.Drawing.Point(758, 83);

this.radioButton3.Name = "radioButton3";

this.radioButton3.Size = new System.Drawing.Size(111, 20);

this.radioButton3.TabIndex = 15;

this.radioButton3.TabStop = true;

this.radioButton3.Text = "Read from file";

this.radioButton3.UseVisualStyleBackColor = true;

//

// checkBox1

//

this.checkBox1.AutoSize = true;

this.checkBox1.Location = new System.Drawing.Point(668, 31);

this.checkBox1.Name = "checkBox1";

this.checkBox1.Size = new System.Drawing.Size(60, 20);

this.checkBox1.TabIndex = 16;

this.checkBox1.Text = "IDEA";

this.checkBox1.UseVisualStyleBackColor = true;

this.checkBox1.CheckedChanged += new System.EventHandler(this.checkBox1\_CheckedChanged);

//

// checkBox2

//

this.checkBox2.AutoSize = true;

this.checkBox2.Location = new System.Drawing.Point(668, 57);

this.checkBox2.Name = "checkBox2";

this.checkBox2.Size = new System.Drawing.Size(57, 20);

this.checkBox2.TabIndex = 17;

this.checkBox2.Text = "RSA";

this.checkBox2.UseVisualStyleBackColor = true;

this.checkBox2.CheckedChanged += new System.EventHandler(this.checkBox2\_CheckedChanged);

//

// btnSave

//

this.btnSave.Location = new System.Drawing.Point(486, 178);

this.btnSave.Name = "btnSave";

this.btnSave.Size = new System.Drawing.Size(114, 46);

this.btnSave.TabIndex = 18;

this.btnSave.Text = "Save in file";

this.btnSave.UseVisualStyleBackColor = true;

this.btnSave.Click += new System.EventHandler(this.btnSave\_Click);

//

// btnPerevirka

//

this.btnPerevirka.Location = new System.Drawing.Point(891, 279);

this.btnPerevirka.Name = "btnPerevirka";

this.btnPerevirka.Size = new System.Drawing.Size(151, 67);

this.btnPerevirka.TabIndex = 19;

this.btnPerevirka.Text = "Digital signature\r\nverification";

this.btnPerevirka.UseVisualStyleBackColor = true;

this.btnPerevirka.Click += new System.EventHandler(this.btnPerevirka\_Click);

//

// btnClear

//

this.btnClear.Location = new System.Drawing.Point(937, 29);

this.btnClear.Name = "btnClear";

this.btnClear.Size = new System.Drawing.Size(105, 31);

this.btnClear.TabIndex = 20;

this.btnClear.Text = "Clear";

this.btnClear.UseVisualStyleBackColor = true;

this.btnClear.Click += new System.EventHandler(this.btnClear\_Click);

//

// btnReadIdeaKey

//

this.btnReadIdeaKey.Location = new System.Drawing.Point(891, 352);

this.btnReadIdeaKey.Name = "btnReadIdeaKey";

this.btnReadIdeaKey.Size = new System.Drawing.Size(151, 67);

this.btnReadIdeaKey.TabIndex = 21;

this.btnReadIdeaKey.Text = "Read IDEA key";

this.btnReadIdeaKey.UseVisualStyleBackColor = true;

this.btnReadIdeaKey.Click += new System.EventHandler(this.btnReadIdeaKey\_Click);

//

// btnReadRSAkeys

//

this.btnReadRSAkeys.Location = new System.Drawing.Point(891, 425);

this.btnReadRSAkeys.Name = "btnReadRSAkeys";

this.btnReadRSAkeys.Size = new System.Drawing.Size(151, 58);

this.btnReadRSAkeys.TabIndex = 22;

this.btnReadRSAkeys.Text = "Read RSA keys";

this.btnReadRSAkeys.UseVisualStyleBackColor = true;

this.btnReadRSAkeys.Click += new System.EventHandler(this.btnReadRSAkeys\_Click);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(1133, 622);

this.Controls.Add(this.btnReadRSAkeys);

this.Controls.Add(this.btnReadIdeaKey);

this.Controls.Add(this.btnClear);

this.Controls.Add(this.btnPerevirka);

this.Controls.Add(this.btnSave);

this.Controls.Add(this.checkBox2);

this.Controls.Add(this.checkBox1);

this.Controls.Add(this.radioButton3);

this.Controls.Add(this.radioButton2);

this.Controls.Add(this.radioButton1);

this.Controls.Add(this.label3);

this.Controls.Add(this.label4);

this.Controls.Add(this.tbq);

this.Controls.Add(this.tbp);

this.Controls.Add(this.btnRSA);

this.Controls.Add(this.btnIDEA\_E);

this.Controls.Add(this.listBox1);

this.Controls.Add(this.label2);

this.Controls.Add(this.label1);

this.Controls.Add(this.tbText);

this.Controls.Add(this.btnRead);

this.Controls.Add(this.tbSymkey);

this.Name = "Form1";

this.Text = "Form1";

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.TextBox tbSymkey;

private System.Windows.Forms.Button btnRead;

private System.Windows.Forms.SaveFileDialog saveFileDialog1;

private System.Windows.Forms.TextBox tbText;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.ListBox listBox1;

private System.Windows.Forms.Button btnIDEA\_E;

private System.Windows.Forms.Button btnRSA;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.TextBox tbq;

private System.Windows.Forms.TextBox tbp;

private System.Windows.Forms.RadioButton radioButton1;

private System.Windows.Forms.RadioButton radioButton2;

private System.Windows.Forms.RadioButton radioButton3;

private System.Windows.Forms.CheckBox checkBox1;

private System.Windows.Forms.CheckBox checkBox2;

private System.Windows.Forms.Button btnSave;

private System.Windows.Forms.Button btnPerevirka;

private System.Windows.Forms.Button btnClear;

private System.Windows.Forms.Button btnReadIdeaKey;

private System.Windows.Forms.Button btnReadRSAkeys;

}

}