**Звіт**

**До практичної роботи**

**з дисципліни «Основи інформаційної безпеки»**

Студента Київського Національного Університету ім. Тараса Шевченко

Переверзева Олексія Сергійовича

Факультет інформаційних технологій

Група МІТ-21

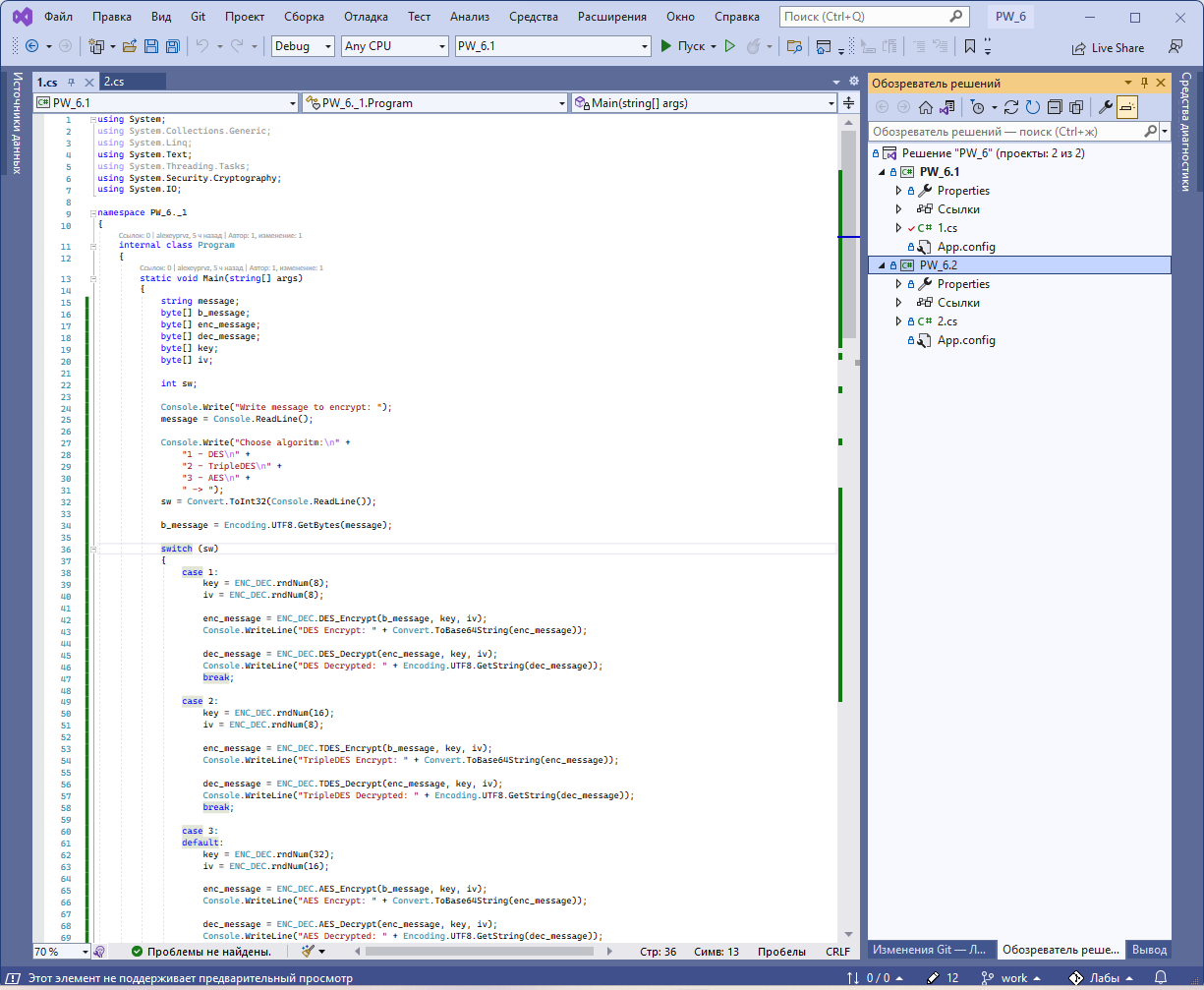


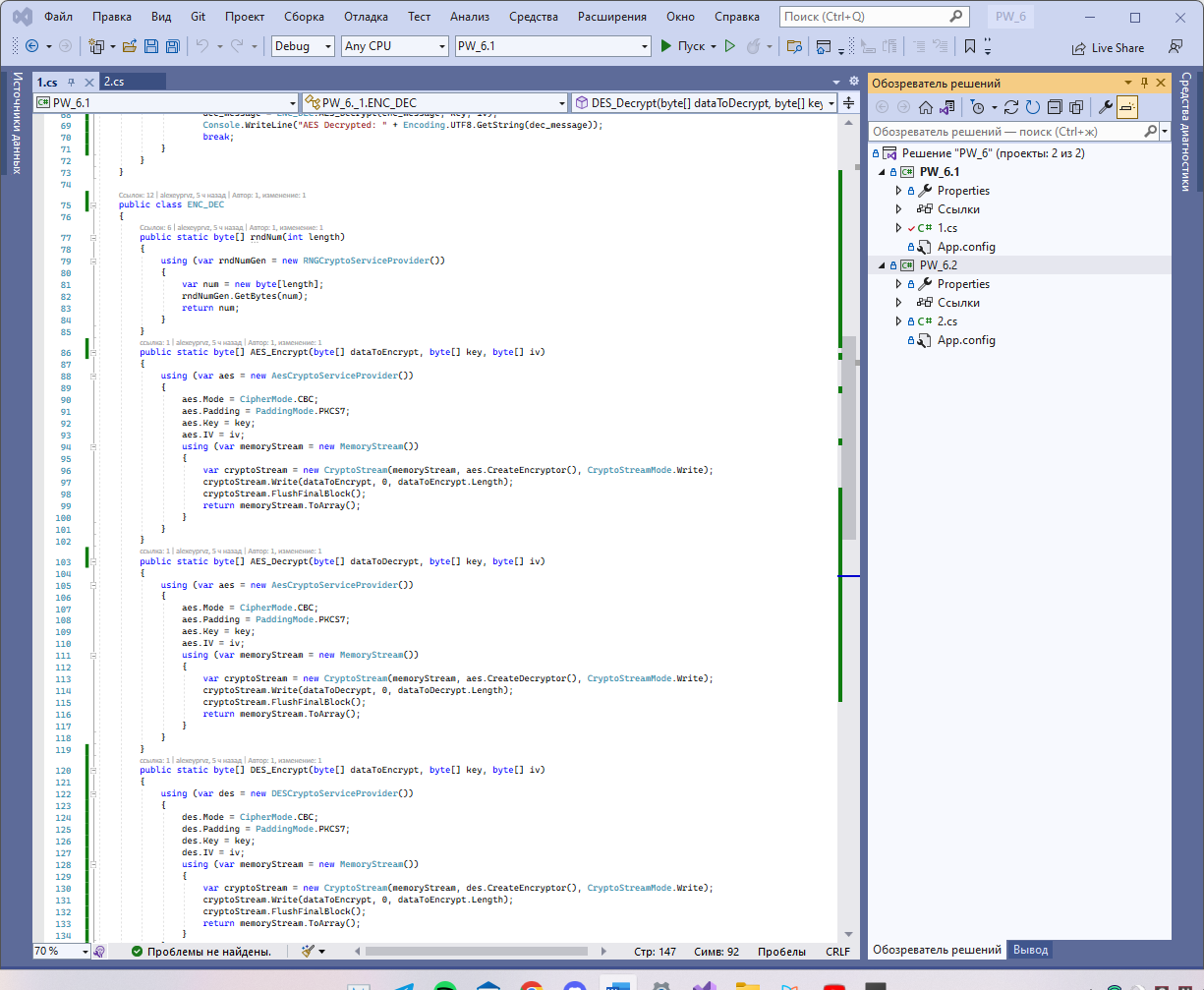
**Практичне заняття №6**

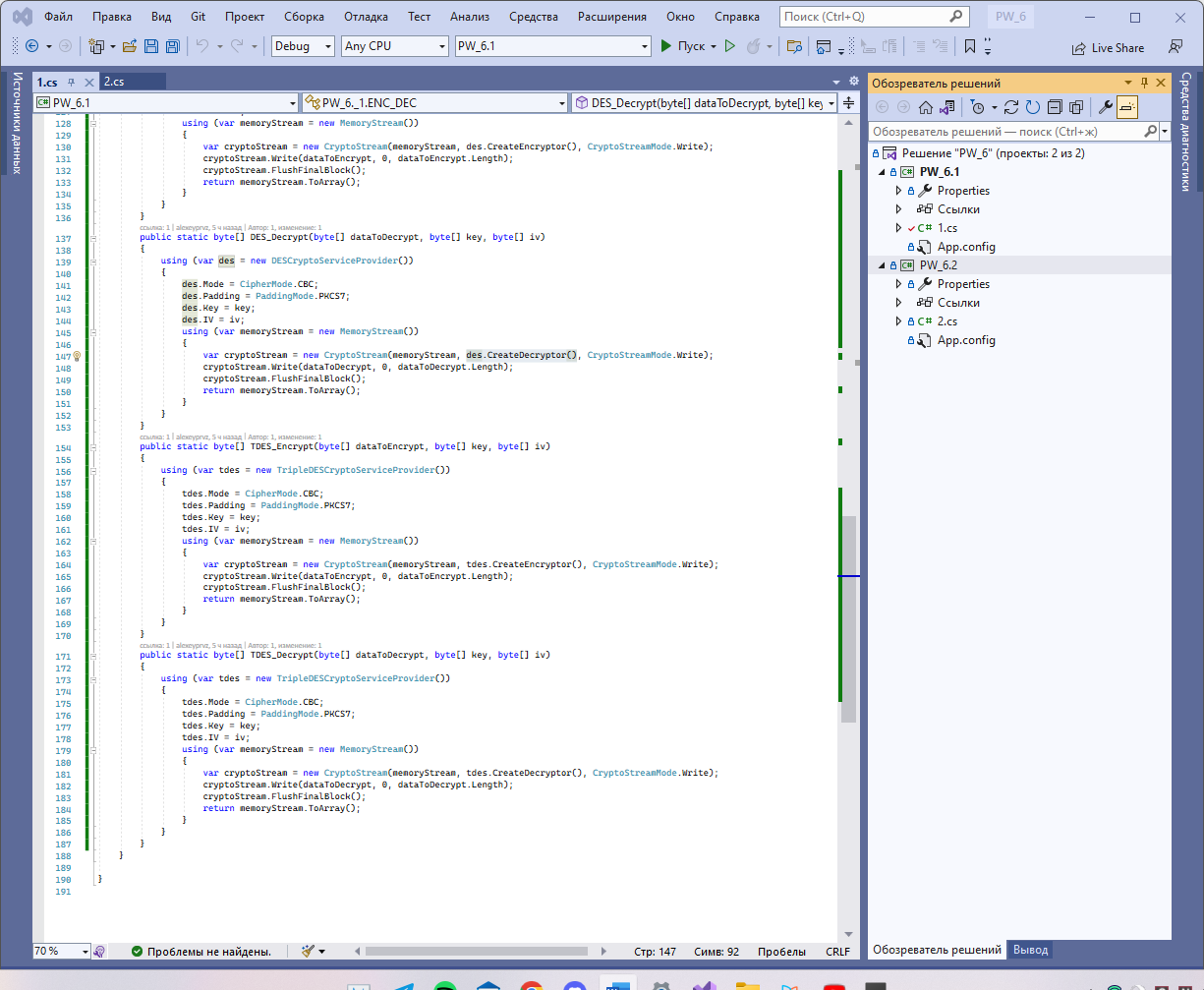
**Тема:** Забезпечення конфіденційності інформації з використанням симетричного шифрування

**Хід роботи**

1. Написати програму, яка виконує зашифровування та розшифровування даних з використанням алгоритмів симетричного шифрування DES, Triple-DES, AES. Секретний ключ та вектор ініціалізації генерується випадковим чином.







Спочатку розглянемо class ENC\_DEC:

У методі rndNum генеруємо генеруємо симетричны ключі шифрування та вектори ініціалізації:

public static byte[] rndNum(int length)

{

using (var rndNumGen = new RNGCryptoServiceProvider())

{

var num = new byte[length];

rndNumGen.GetBytes(num);

return num;

}

}

Далі розглянемо методи шифрування на прикладі об’єкту DESCryptoServiceProvider.

public static byte[] DES\_Encrypt(byte[] dataToEncrypt, byte[] key, byte[] iv)

{

using (var des = new DESCryptoServiceProvider())

{

des.Mode = CipherMode.CBC;

des.Padding = PaddingMode.PKCS7;

des.Key = key;

des.IV = iv;

using (var memoryStream = new MemoryStream())

{

var cryptoStream = new CryptoStream(memoryStream, des.CreateEncryptor(), CryptoStreamMode.Write);

cryptoStream.Write(dataToEncrypt, 0, dataToEncrypt.Length);

cryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

Метод шифрування приймає три параметри, кожен з яких є масивом байтів. Перший параметр - це дані, які ми хочемо зашифрувати. Наступним параметром є ключ шифрування, а останнім параметром є вектор ініціалізації. Перше, що робить метод Encrypt, - це створити екземпляр Об'єкта DESCryptoServiceProvider. Потім явно задаються режими заповнення і шифрування. Значення, встановлені в цьому прикладі, є значеннями за замовчуванням, але я встановив їх явними. Далі призначаються ключ і вектори ініціалізації, які були передані в якості параметрів. Потім створюються екземпляри MemoryStream і CryptoStream. Симетричні бібліотеки шифрування в. NET базуються на потоці, а криптографічні операції виконуються в CryptoStream. Оскільки ми передаємо дані, що підлягають шифруванню, до цього методу, ми можемо використовувати потік пам'яті.

Коли ми створюємо екземпляр Об'єкта CryptoStream, ми передаємо des.CreateEncryptor(), який налаштовує криптопотік за допомогою нашого налаштованого Об'єкта DESCryptoServiceProvider.

var cryptoStream = new CryptoStream(memoryStream, aes.CreateEncryptor(), CryptoStreamMode.Write);

Як тільки потокові об'єкти налаштовані, нам потрібно виконати фактичну операцію шифрування, використовуючи наступний код:

cryptoStream.Write(dataToEncrypt, 0, dataToEncrypt.Length);

cryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

Виклик Write приймає дані, які ми хочемо зашифрувати, та їх довжину в байтах та виконує операцію шифрування. Потім ми викликаємо FlushFinalBlock, який оновлює джерело даних поточним станом буфера. Потім буфер очищається. Наші зашифровані дані повертаються до методу виклику.

Для методі дешифрування аналогічний код, тільки коли створюємо екземпляр Об'єкта CryptoStream, ми передаємо des.CreateDecryptor():

public static byte[] DES\_Decrypt(byte[] dataToDecrypt, byte[] key, byte[] iv)

{

using (var des = new DESCryptoServiceProvider())

{

des.Mode = CipherMode.CBC;

des.Padding = PaddingMode.PKCS7;

des.Key = key;

des.IV = iv;

using (var memoryStream = new MemoryStream())

{

var cryptoStream = new CryptoStream(memoryStream, des.CreateDecryptor(), CryptoStreamMode.Write);

cryptoStream.Write(dataToDecrypt, 0, dataToDecrypt.Length);

cryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

Відповідно код для об’єктів TripleDESCryptoServiceProvider та AesCryptoServiceProvider буде аналогічній код для шифрування та дешифрування:

public static byte[] AES\_Encrypt(byte[] dataToEncrypt, byte[] key, byte[] iv)

{

using (var aes = new AesCryptoServiceProvider())

{

aes.Mode = CipherMode.CBC;

aes.Padding = PaddingMode.PKCS7;

aes.Key = key;

aes.IV = iv;

using (var memoryStream = new MemoryStream())

{

var cryptoStream = new CryptoStream(memoryStream, aes.CreateEncryptor(), CryptoStreamMode.Write);

cryptoStream.Write(dataToEncrypt, 0, dataToEncrypt.Length);

cryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

public static byte[] AES\_Decrypt(byte[] dataToDecrypt, byte[] key, byte[] iv)

{

using (var aes = new AesCryptoServiceProvider())

{

aes.Mode = CipherMode.CBC;

aes.Padding = PaddingMode.PKCS7;

aes.Key = key;

aes.IV = iv;

using (var memoryStream = new MemoryStream())

{

var cryptoStream = new CryptoStream(memoryStream, aes.CreateDecryptor(), CryptoStreamMode.Write);

cryptoStream.Write(dataToDecrypt, 0, dataToDecrypt.Length);

cryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

public static byte[] TDES\_Encrypt(byte[] dataToEncrypt, byte[] key, byte[] iv)

{

using (var tdes = new TripleDESCryptoServiceProvider())

{

tdes.Mode = CipherMode.CBC;

tdes.Padding = PaddingMode.PKCS7;

tdes.Key = key;

tdes.IV = iv;

using (var memoryStream = new MemoryStream())

{

var cryptoStream = new CryptoStream(memoryStream, tdes.CreateEncryptor(), CryptoStreamMode.Write);

cryptoStream.Write(dataToEncrypt, 0, dataToEncrypt.Length);

cryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

public static byte[] TDES\_Decrypt(byte[] dataToDecrypt, byte[] key, byte[] iv)

{

using (var tdes = new TripleDESCryptoServiceProvider())

{

tdes.Mode = CipherMode.CBC;

tdes.Padding = PaddingMode.PKCS7;

tdes.Key = key;

tdes.IV = iv;

using (var memoryStream = new MemoryStream())

{

var cryptoStream = new CryptoStream(memoryStream, tdes.CreateDecryptor(), CryptoStreamMode.Write);

cryptoStream.Write(dataToDecrypt, 0, dataToDecrypt.Length);

cryptoStream.FlushFinalBlock();

return memoryStream.ToArray();

}

}

}

Розглянемо основний код, class Program:

Ініціалізуємо потрібні змінні:

string message;

byte[] b\_message;

byte[] enc\_message;

yte[] dec\_message;

byte[] key;

byte[] iv;

int sw;

Далі виконуємо запит для введення повідомлення з клавіатури, яке ми будемо шифрувати та зберігаємо його у змінну message:

Console.Write("Write message to encrypt: ");

message = Console.ReadLine();

Далі виконуємо запит на вибір алгоритму шифрування:

Console.Write("Choose algoritm:\n" +

"1 - DES\n" +

"2 - TripleDES\n" +

"3 - AES\n" +

" -> ");

sw = Convert.ToInt32(Console.ReadLine());

Також перетворюємо текст для шифрування у масив байтів:

b\_message = Encoding.UTF8.GetBytes(message);

Відповідно, в залежності від обраного алгоритму, буде виконуватись потрібний код в операторі switch:

switch (sw)

{

case '1':

key = ENC\_DEC.rndNum(8);

iv = ENC\_DEC.rndNum(8);

enc\_message = ENC\_DEC.DES\_Encrypt(b\_message, key, iv);

Console.WriteLine("DES Encrypt: " + Convert.ToBase64String(enc\_message));

dec\_message = ENC\_DEC.DES\_Decrypt(enc\_message, key, iv);

Console.WriteLine("DES Decrypted: " + Encoding.UTF8.GetString(dec\_message));

break;

case '2':

key = ENC\_DEC.rndNum(16);

iv = ENC\_DEC.rndNum(8);

enc\_message = ENC\_DEC.TDES\_Encrypt(b\_message, key, iv);

Console.WriteLine("TripleDES Encrypt: " + Convert.ToBase64String(enc\_message));

dec\_message = ENC\_DEC.TDES\_Decrypt(enc\_message, key, iv);

Console.WriteLine("TripleDES Decrypted: " + Encoding.UTF8.GetString(dec\_message));

break;

case '3':

default:

key = ENC\_DEC.rndNum(32);

iv = ENC\_DEC.rndNum(16);

enc\_message = ENC\_DEC.AES\_Encrypt(b\_message, key, iv);

Console.WriteLine("AES Encrypt: " + Convert.ToBase64String(enc\_message));

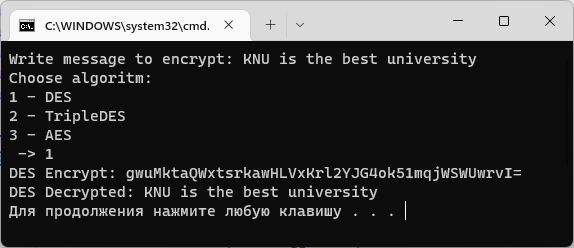
dec\_message = ENC\_DEC.AES\_Decrypt(enc\_message, key, iv);

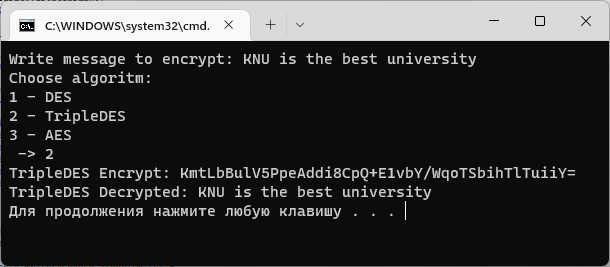
Console.WriteLine("AES Decrypted: " + Encoding.UTF8.GetString(dec\_message));

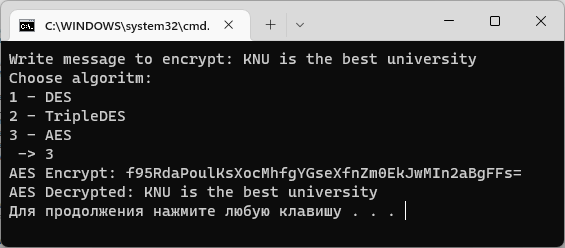
break;

}

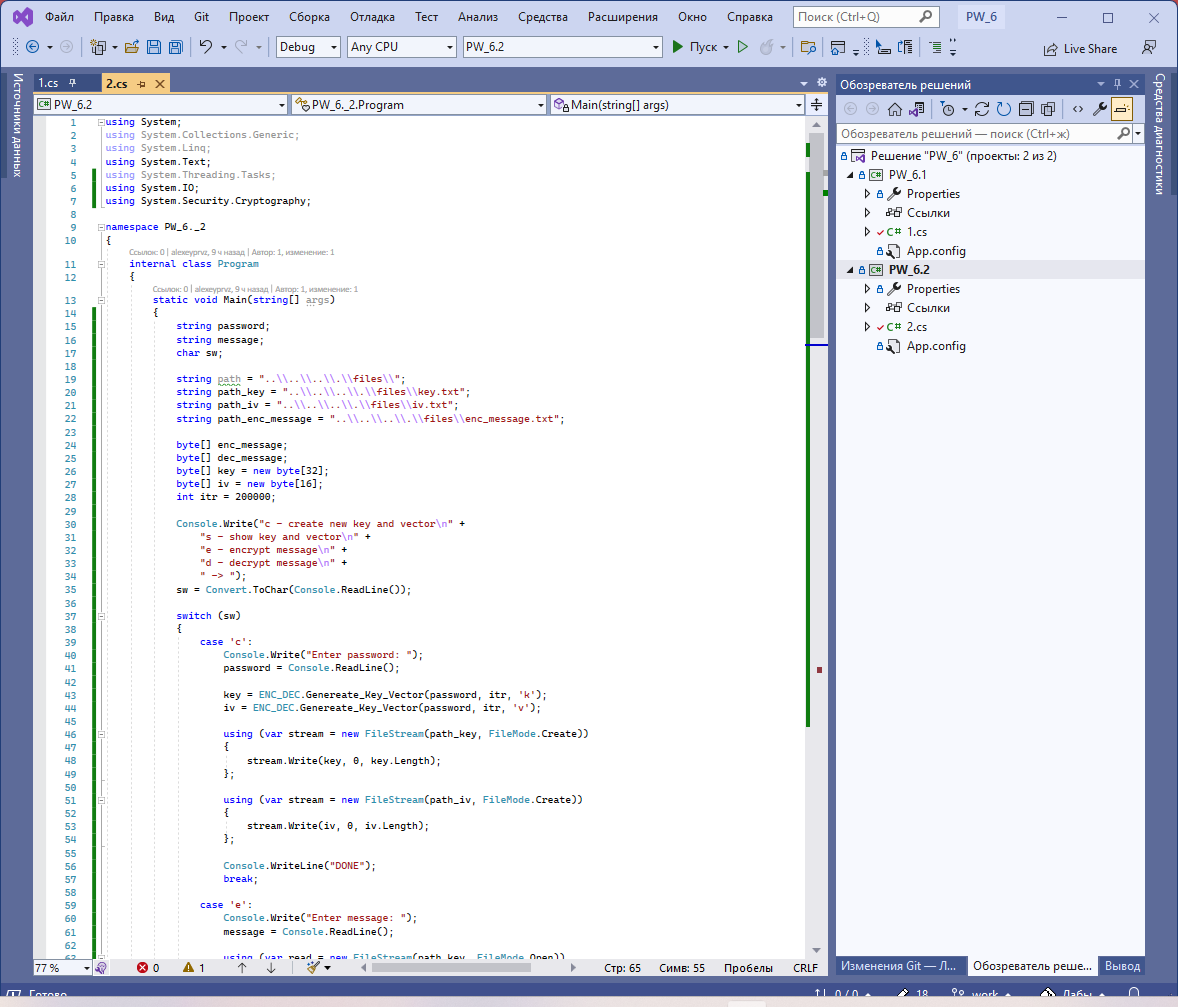
Виконання програми:

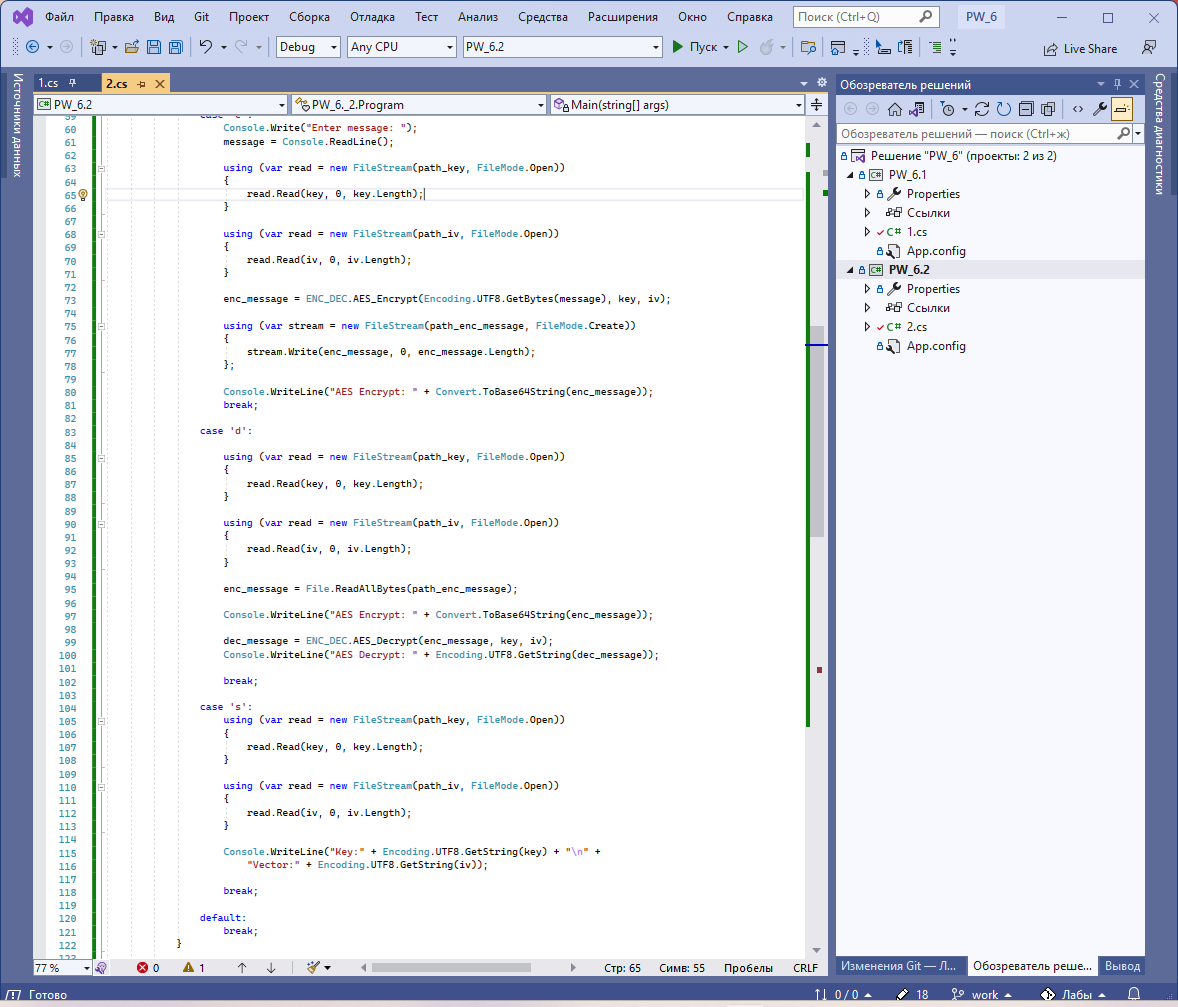


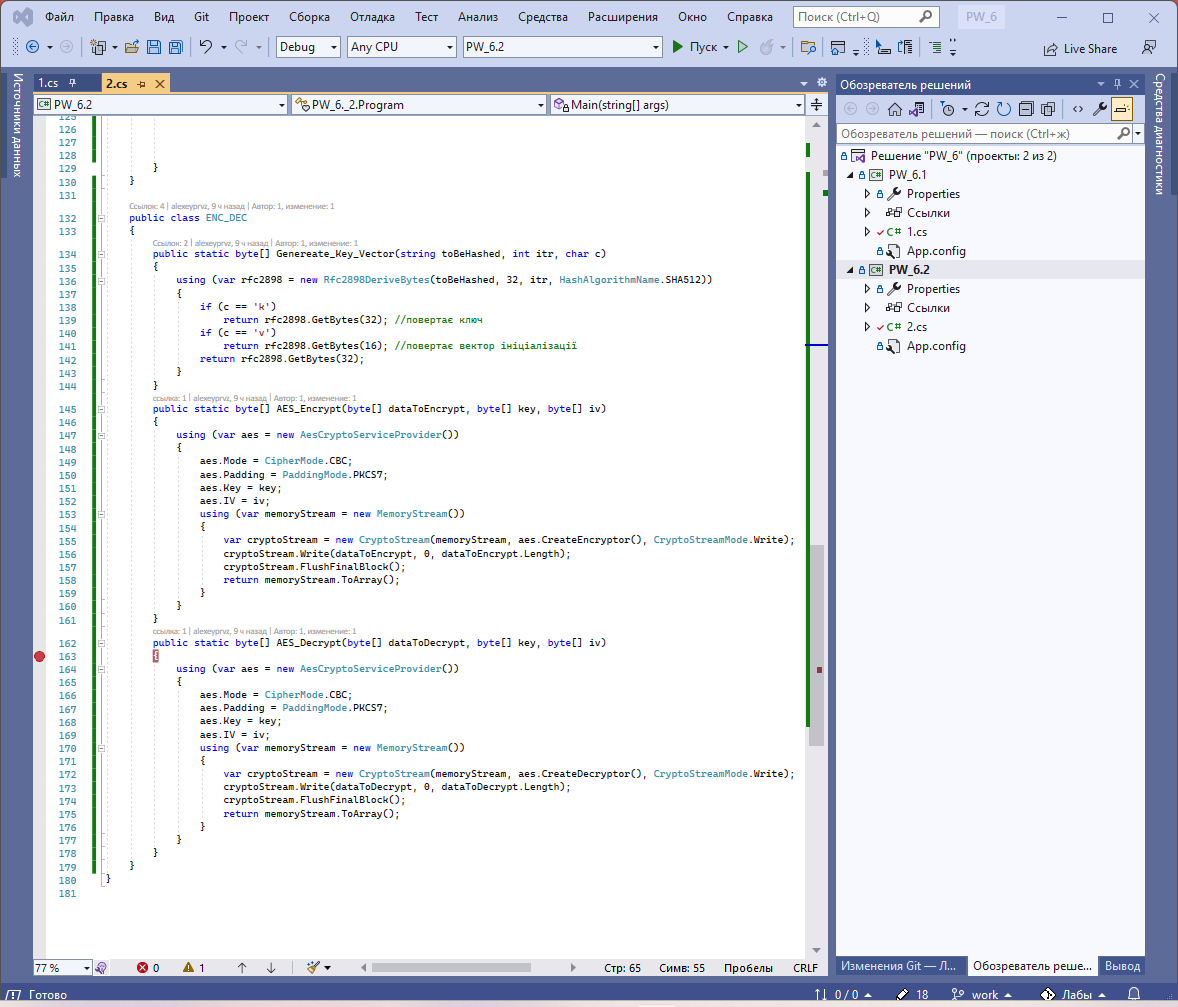




1. Для програми з п.1. реалізувати можливість задання секретного ключа та вектора ініціалізації за допомогою псевдовипадкової послідовності із використанням пароля. «Сіль» генерувати як випадкову послідовність байтів. Число ітерацій = номер варіанта \* 10’000. (Варіант 20, число ітерацій 200000) Підказка: використовувати клас Rfc2898DeriveBytes







Pозглянемо для початку class ENC\_DEC:

Так як метод шифрування за допомогою AES алгоритму тут такий самий, то не буду на цьому зипунятись. Але секретний ключ та вектор ініціалізації для шифрування генеруємо, використовуючи клас Rfc2898DeriveBytes. У функцію передаємо, введений користувачем, пароль, кількість ітерацій та змінну char c щоб повернути потрібну кількість байт (32 байти для ключа, 16 – вектора ініціалізації). Сіль генеруємо за допомогою перевантаженої функції Rfc2898DeriveBytes, в якій вказуємо довжину солі.

public static byte[] Genereate\_Key\_Vector(string toBeHashed, int itr, char c)

{

using (var rfc2898 = new Rfc2898DeriveBytes(toBeHashed, 32, itr, HashAlgorithmName.SHA512))

{

if (c == 'k')

return rfc2898.GetBytes(32); //повертає ключ

if (c == 'v')

return rfc2898.GetBytes(16); //повертає вектор ініціалізації

return rfc2898.GetBytes(32);

}

}

Далі розберемо основний код.

Для початку ініціалізуємо потрібні змінні:

string password;

string message;

char sw;

*1 – введення паролю для генерацію ключа та вектора ініцалізації, 2 – для введення повідомлення, яке буде зашифровано, 3 – для вибору виконання операції*

string path = "..\\..\\..\\.\\files\\";

string path\_key = "..\\..\\..\\.\\files\\key.txt";

string path\_iv = "..\\..\\..\\.\\files\\iv.txt";

string path\_enc\_message = "..\\..\\..\\.\\files\\enc\_message.txt";

*1 – шлях до папки з файлами, 2 - шлях до файлу з секретним ключем, 3 - шлях до файлу з вектором ініціалізації, 4 - шлях до файлу з зашифрованим повідомленням*

byte[] enc\_message;

byte[] dec\_message;

byte[] key = new byte[32];

byte[] iv = new byte[16];

int itr = 200000;

*1 – для запису зашифрованого повідомлення, 2 - для запису розшифрованого повідомлення, 3 – для запису згенерованого секретного ключа, 4 – для запису згенерованого вектора ініціалізації, 5 – кількість ітерацій.*

Далі запитуємо у користувача яку операцію потрібно виконати:

Console.Write("c - create new key and vector\n" +

"s - show key and vector\n" +

"e - encrypt message\n" +

"d - decrypt message\n" +

" -> ");

sw = Convert.ToChar(Console.ReadLine());

Відповідно розглянемо по порядку варіанти роботи програми.

Перший варіант створює новий секретний ключ та вектор ініціалізації, використовуючи введений пароль, так записує їх у відповідні файли, створивши їх або перезаписавши їх минулий вміст.

case 'c':

Console.Write("Enter password: ");

password = Console.ReadLine();

key = ENC\_DEC.Genereate\_Key\_Vector(password, itr, 'k');

iv = ENC\_DEC.Genereate\_Key\_Vector(password, itr, 'v');

using (var stream = new FileStream(path\_key, FileMode.Create))

{

stream.Write(key, 0, key.Length);

};

using (var stream = new FileStream(path\_iv, FileMode.Create))

{

stream.Write(iv, 0, iv.Length);

};

Console.WriteLine("DONE");

break;

Щоб отримати потрібну кількість байт ми передаємо ще одну змінну, яку перевірять умовні оператори у функції:

key = ENC\_DEC.Genereate\_Key\_Vector(password, itr, 'k');

iv = ENC\_DEC.Genereate\_Key\_Vector(password, itr, 'v');

if (c == 'k')

return rfc2898.GetBytes(32); //повертає ключ

if (c == 'v')

return rfc2898.GetBytes(16); //повертає вектор ініціалізації

Далі записуємо у файли key.txt та iv.txt секретний ключ та вектор ініціалізації відповідно:

using (var stream = new FileStream(path\_key, FileMode.Create))

{

stream.Write(key, 0, key.Length);

};

using (var stream = new FileStream(path\_iv, FileMode.Create))

{

stream.Write(iv, 0, iv.Length);

};

Також можна вивести їх у консоль:

case 's':

using (var read = new FileStream(path\_key, FileMode.Open))

{

read.Read(key, 0, key.Length);

}

using (var read = new FileStream(path\_iv, FileMode.Open))

{

read.Read(iv, 0, iv.Length);

}

Console.WriteLine("Key:" + Encoding.UTF8.GetString(key) + "\n" +

"Vector:" + Encoding.UTF8.GetString(iv));

break;

Розглянемо сам процес шифрування:

case 'e':

Console.Write("Enter message: ");

message = Console.ReadLine();

using (var read = new FileStream(path\_key, FileMode.Open))

{

read.Read(key, 0, key.Length);

}

using (var read = new FileStream(path\_iv, FileMode.Open))

{

read.Read(iv, 0, iv.Length);

}

enc\_message = ENC\_DEC.AES\_Encrypt(Encoding.UTF8.GetBytes(message), key, iv);

using (var stream = new FileStream(path\_enc\_message, FileMode.Create))

{

stream.Write(enc\_message, 0, enc\_message.Length);

};

Console.WriteLine("AES Encrypt: " + Convert.ToBase64String(enc\_message));

break;

Спочатку робимо запит на введення повідомлення для шифрування, а потім зберігаємо у змінній message:

Console.Write("Enter message: ");

message = Console.ReadLine();

Далі читаємо вміст файлів з ключем та вектором для подальшого шифрування введеного повідомлення:

using (var read = new FileStream(path\_key, FileMode.Open))

{

read.Read(key, 0, key.Length);

}

using (var read = new FileStream(path\_iv, FileMode.Open))

{

read.Read(iv, 0, iv.Length);

}

Відповідно саме шифрування:

enc\_message = ENC\_DEC.AES\_Encrypt(Encoding.UTF8.GetBytes(message), key, iv);

Записуємо у файл:

using (var stream = new FileStream(path\_enc\_message, FileMode.Create))

{

stream.Write(enc\_message, 0, enc\_message.Length);

};

Також виводимо у консоль, конвертуючи масив байтів у рядок:

Console.WriteLine("AES Encrypt: " + Convert.ToBase64String(enc\_message));

Далі розглянемо дешифрування:

case 'd':

using (var read = new FileStream(path\_key, FileMode.Open))

{

read.Read(key, 0, key.Length);

}

using (var read = new FileStream(path\_iv, FileMode.Open))

{

read.Read(iv, 0, iv.Length);

}

enc\_message = File.ReadAllBytes(path\_enc\_message);

Console.WriteLine("AES Encrypt: " + Convert.ToBase64String(enc\_message));

dec\_message = ENC\_DEC.AES\_Decrypt(enc\_message, key, iv);

Console.WriteLine("AES Decrypt: " + Encoding.UTF8.GetString(dec\_message));

break;

Так само читаємо вміст файлів з ключем та вектором, а також файл з шифрованим повідомленням:

enc\_message = File.ReadAllBytes(path\_enc\_message);

Виводимо у такому вигляді у консоль для зручності:

Console.WriteLine("AES Encrypt: " + Convert.ToBase64String(enc\_message));

Далі відповідно дешифруємо:

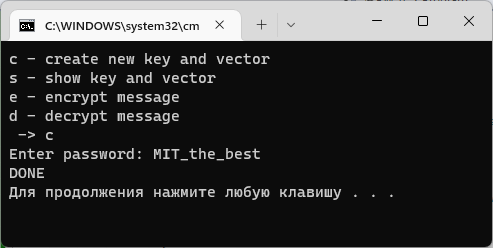
dec\_message = ENC\_DEC.AES\_Decrypt(enc\_message, key, iv);

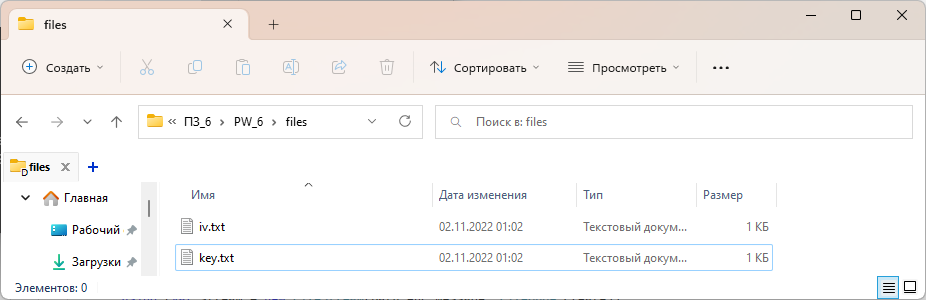
У кінці виводимо оригінальне повідомлення:

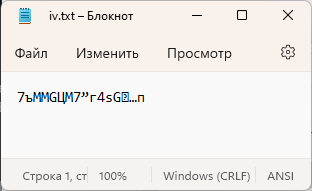
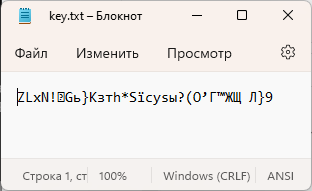
Console.WriteLine("AES Decrypt: " + Encoding.UTF8.GetString(dec\_message));

Виконання програми:

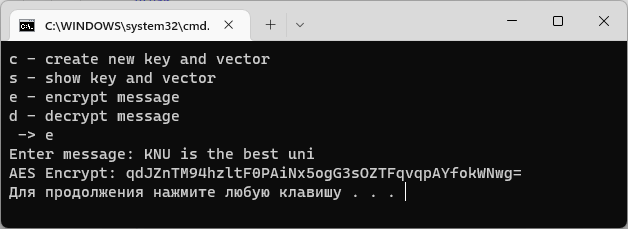
Спочатку створимо нові ключ та вектор:

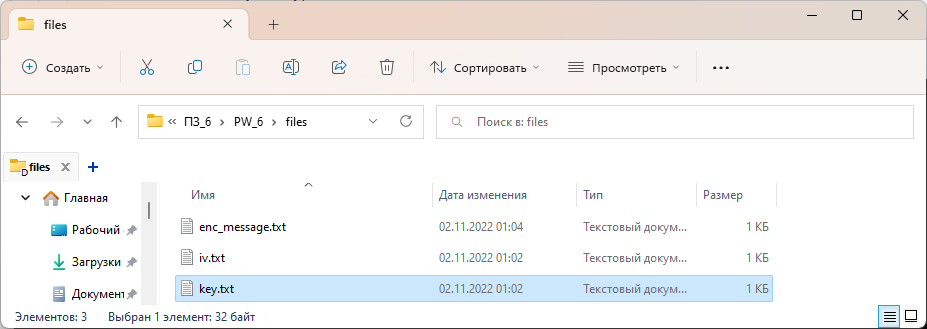


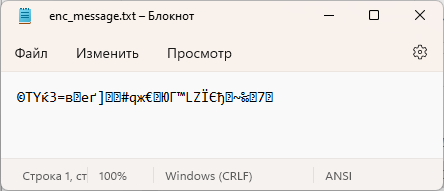


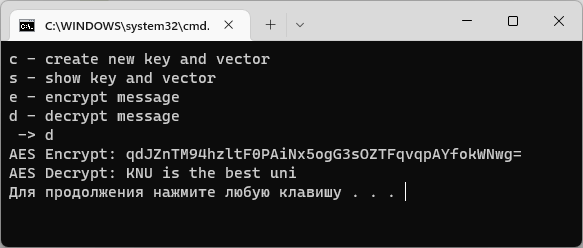
Шифруємо повідомлення:







Дешифруємо повідомлення:



Також завжди можемо переглянути ключ та вектор:

