Міністерство освіти і науки

Київський національний університет імені Тараса Григоровича Шевченка

Кафедра Мережевих та Інтернет технологій

**Звіт**

з практичної роботи **№6**

з дисципліни: “Основи інформаційної безпеки”

**«Забезпечення конфіденційності інформації з використанням симетричного шифрування»**

Виконав:

Студент групи МІТ-21

Горбун О. М.

**Результати виконання завдань роботи:**

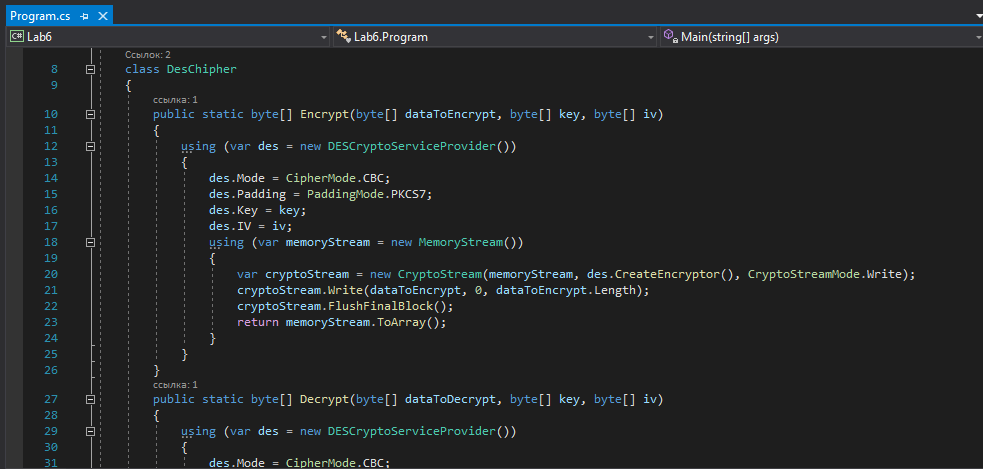
За завданнями практичної роботи було створено програми, які можна переглянути за посиланням:

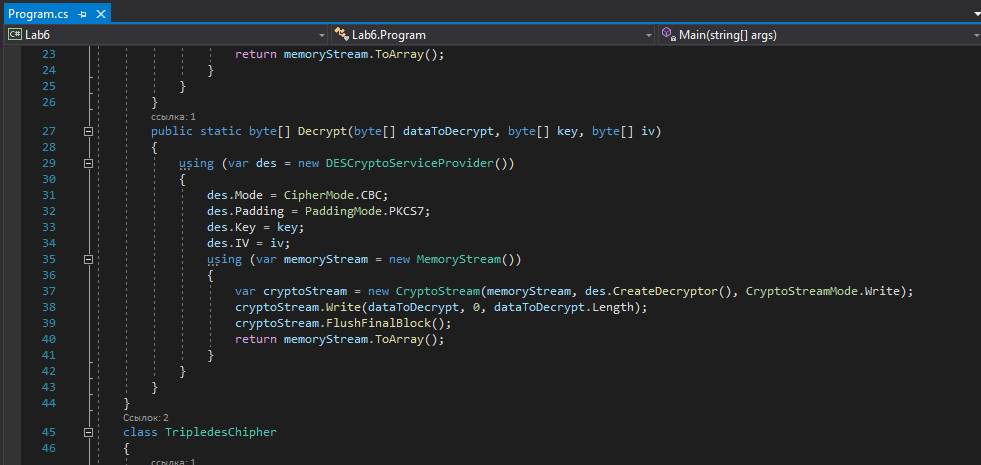
<https://github.com/OleksandrHorbun/Basics_Cybersecurity.git> (Посилання на відкритий репозиторій у веб-сервісі GitHub) завдання у вітці (Branch) Homework\_6.

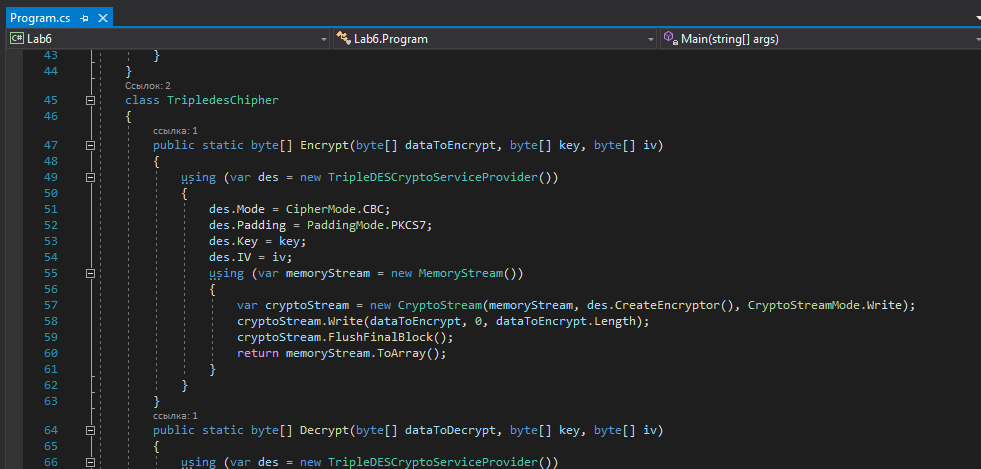
**I частина (перше завдання)**

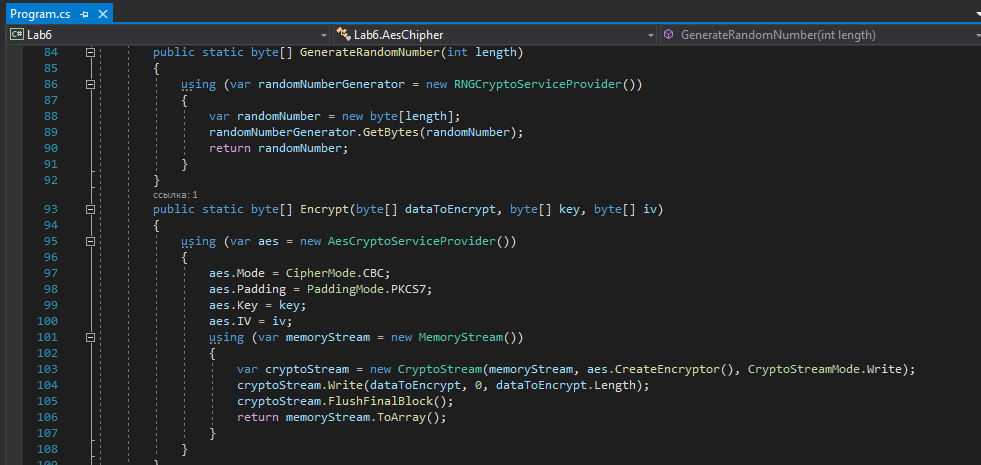
Як можна побачити, на початку програми ключ генерується псевдорандомно (за допомогою функції *GenerateRandomNumber*, яка й повертає псевдорандомне значення) й також є певний рядок, який грає роль «пароля», який треба зашифрувати. Якщо коротко: ключ та повідомлення («пароль») передаються у функцію *Encrypt* (шифрування за алгоритмом шифрування DES - Data Encryption Standard), яка знаходиться у публічному класі *DesChipher*. Результат шифрування записується у змінну в основній частин програми, яка згодом буде виведена. Після шифрування для прикладу, і щоб побачити що повідомлення при дешифруванні не змінюється, одразу ж виконується дешифрування за допомогою функції *Decrypt*. Дешифроване повідомлення теж після цього передається у основну програму для виведення на екран. Ідентично все відбувається і при шифруванні, а разом з тим – розшифруванням за алгоритмами Triple DES (той самий DES, але вже з трьома ключами) і AES (Advanced Encryption Standard). Різниця фактично лише в розмірі або кількості ключів та кількості байт у векторі ініціалізації: DES – два ключі, в обох по вісім байтів в векторі теж вісім байт, Triple DES – три ключа по 16 байтів у векторі ініціалізації вісім байтів, AES – ключі з 32 байтів і у векторі ініціалізації 16 байтів.

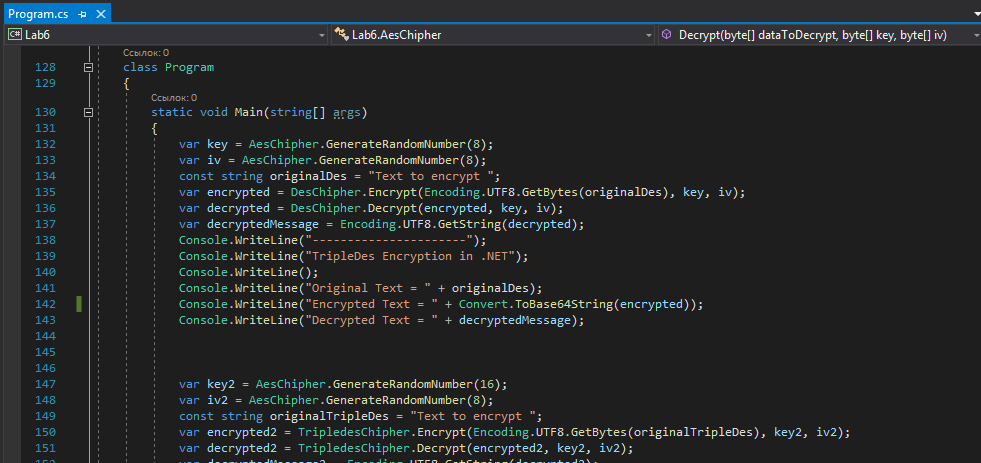
Все описане вище можна побачити на знімках екрану:

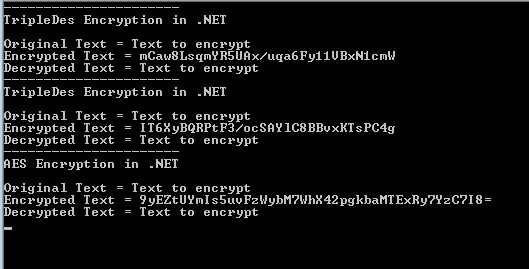








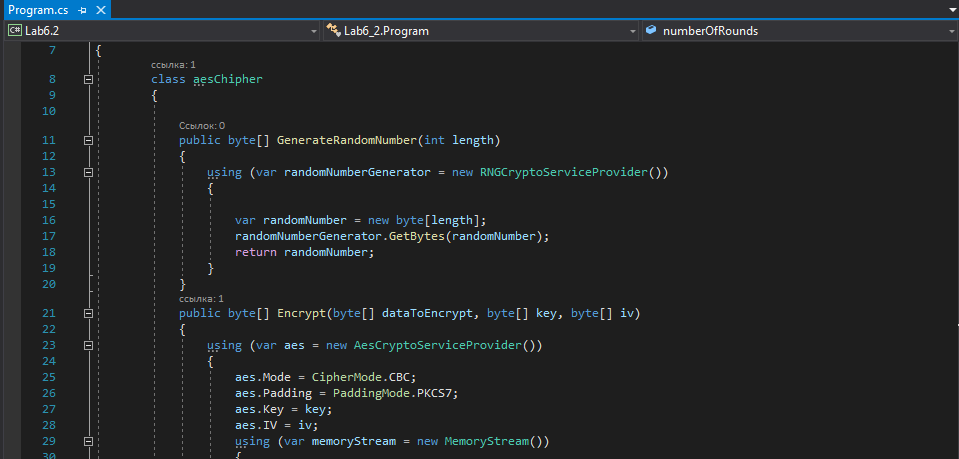


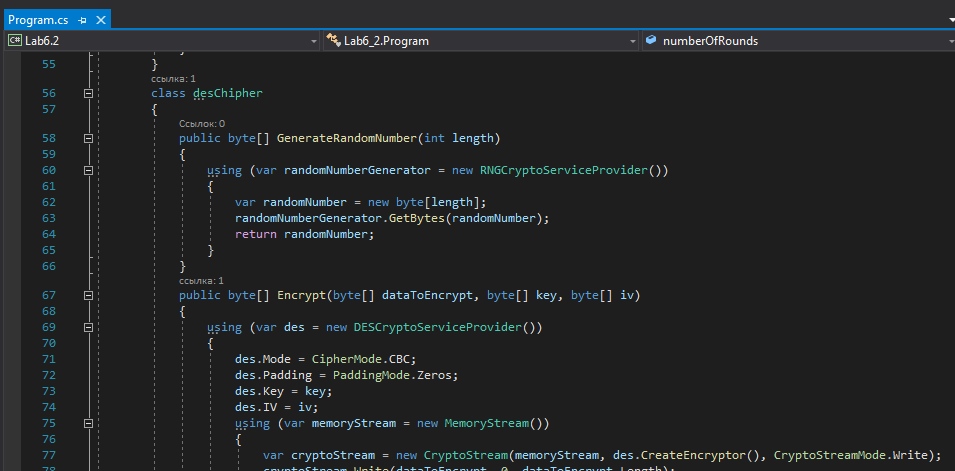


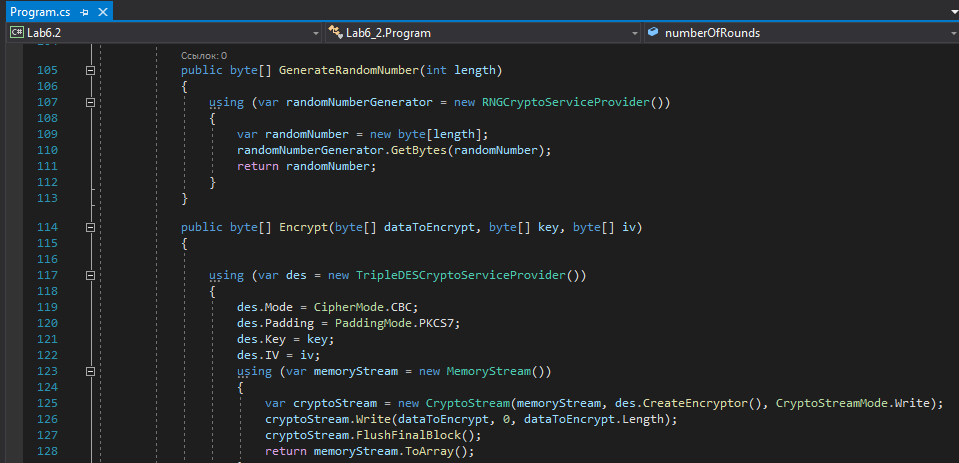
**IІ частина (друге завдання)**

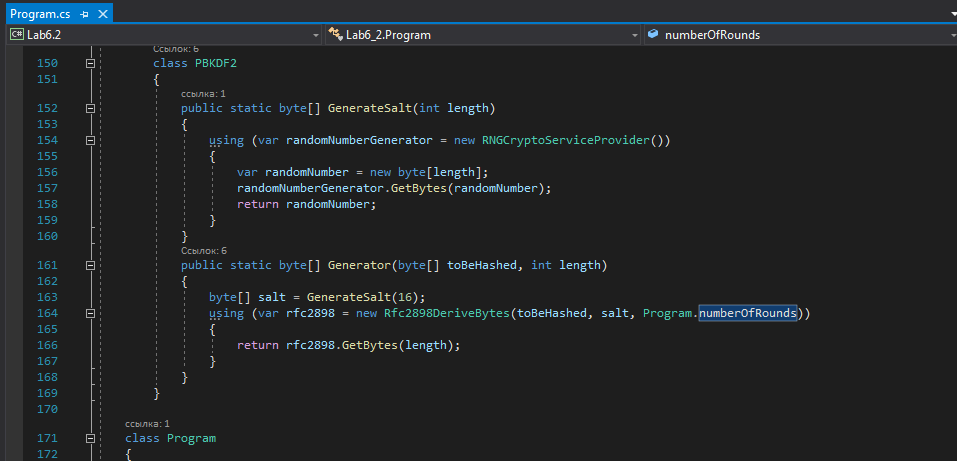
У цій частині все дуже схоже до першої частини завдання: використано ті ж функції та класи для шифрування і розшифрування, але до шифрування ще додається «сіль». «Сіль» генерується випадково за допомогою класу *PBKDF2*, який був описаний у минулих практичних роботах і використовувався для створення «солі» під час хешування. Звісно ж «сіль» - випадковий рядок чисел, додається до всього що треба згенерувати для шифрування, а саме ключа та вектора ініціалізації. Для шифрування й розшифрування використовуються ті ж самі алгоритми: DES, AES, Triple DES. Повідомлення вводиться на початку програми й передається у відповідні функції для шифрування й розшифрування. Більш того, як можна побачити, розшифрування не змінює повідомлення й можна отримати початковий «пароль». Окрім того при генерації «солі» виконується певна кількість ітерацій. У даному випадку 7 варіант за списком = 10000 \* 7 = 70000 ітерацій.

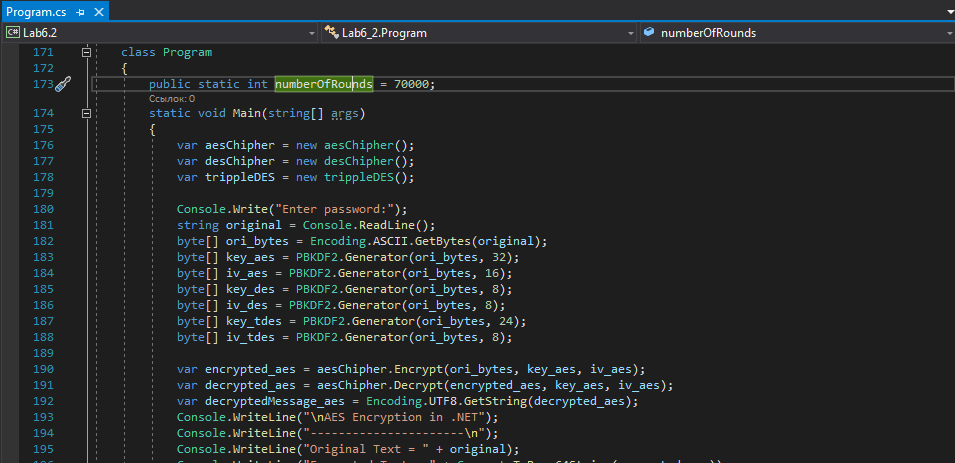
Все описане вище можна побачити на знімках екрану:

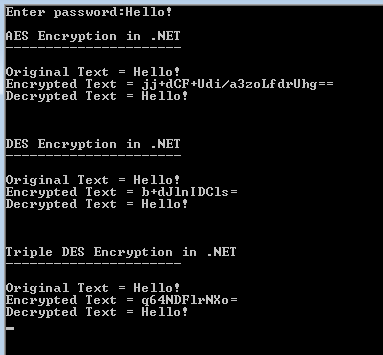












**Висновок:**

Виконуючи цю роботу я навчився зашифровувати та розшифровувати дані за допомогою симетричного шифрування, а також використовувати при цьому «сіль» під час виконання цього процесу.