# 4. ДОСЛІДЖЕННЯ З ЗАСТОСУВАННЯМ ДИНАМІЧНОГО МЕТОДУ

# 4.1 Опис експерименту та застосованої програмної розробки

Динамічний метод відновлення алгоритмів захисту заснований на використанні програмних налагоджувальних засобів (Зневадников). Відладчик — це програма, яка завантажує в пам'ять іншу програму і надає користувачу можливість спостерігати за ходом виконання цієї програми.

Аналіз програми динамічним методом розбивається на три основних етапи:

1) пошук підходів до потрібних функцій програми;

2) пошук вакансій функцій програми (пройти цей самий

шлях);

3) аналіз вакансій функцій програми.

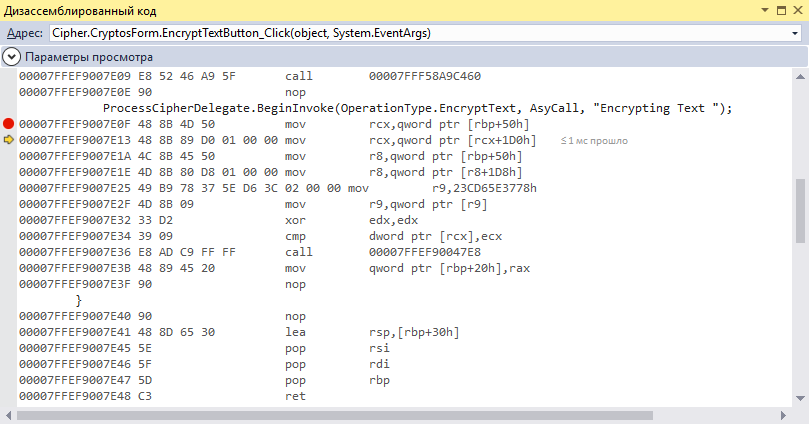
При аналізі машинного коду завданням першого етапу є виявлення в неозорому масиві машинного коду програми деяких «зачіпок», що дозволяють досить швидко дібратися до потрібних функцій. Існують два методи рішення цієї задачі: метод маяків і метод Step-Trace першого етапу.

# 4.2 Результати дослідження з застосуванням динамічного методу

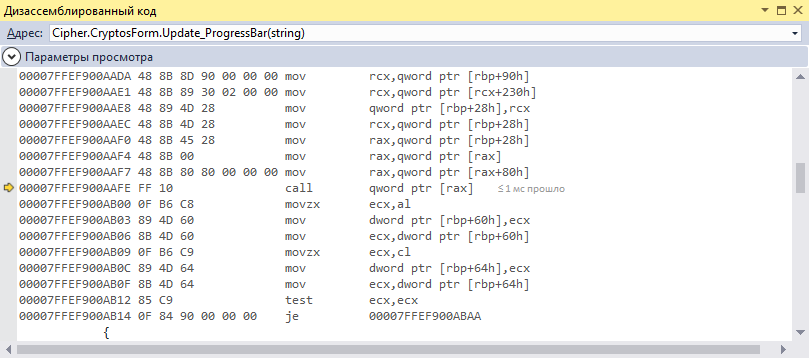
# 4.2.1 Метод маяків

Метод маяків заснований на встановленні точок зупину на так звані маяки — точки програми, в яких програма виконує дії, легко розуміються без знання контексту, в якому ці дії виконуються. У ролі маяків зазвичай виступають системні виклики. Для прикладу розглянемо фрагмент лістингу коду, отриманий за допомогою вбудованого дизассемблера відладчика Microsoft Visual Studio:

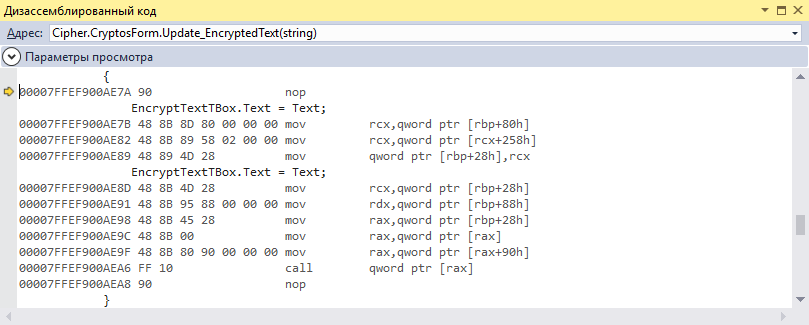
При огляді дизасмленного коду бачимо що відбувається виклик функції що відповідає за передачу тексту та ключа шифрування до криптомодуля. Який після отримання тексту, та ключа шифрування починає шифрувати текст. Зображення 4.2.1 і 4.2.2 відповідно.



Зображення 4.2.1.1 Відправлення тексту та ключу шифрування

 Зображення 4.2.1.2 Шифрування тексту

Після шифрування отриманого тексту функція повертає вже зашифрований текст. Зображення 4.3



Зображення 4.2.1.3 Відображення зашифрованного тексту

# 4.2.2. Дослідження методом Step-Trace першого етапу

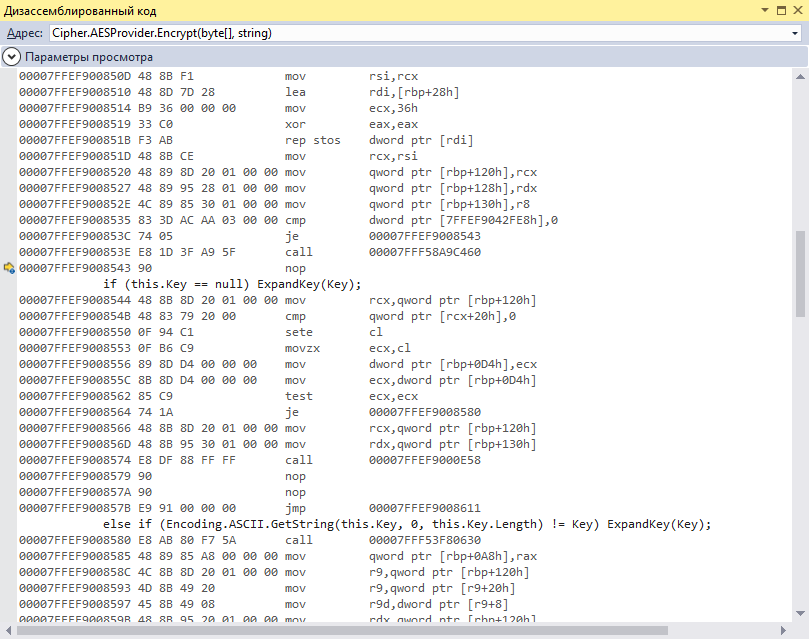
Метод Step-Trace отримав свою назву по двох режимів трасування програми відладчиком: з входом у вкладені функції програми (Trace) і без входу (Step). Суть методу полягає в покроковому проході досліджуваної програми з поперемінним використанням обох режимів. Метод може бути використаний для пошуку в аналізованій програмі функції х, що задовольняє наступним умовам:

а) функція х реалізує цікавлять аналітика алгоритми;

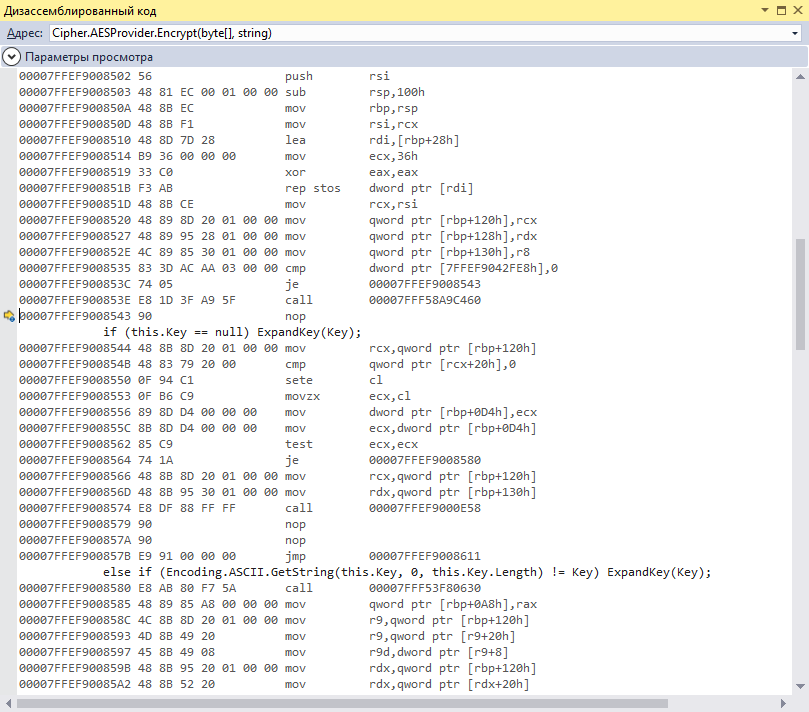
б) виклик функції х легко виявляється аналітиком з зовнішнім

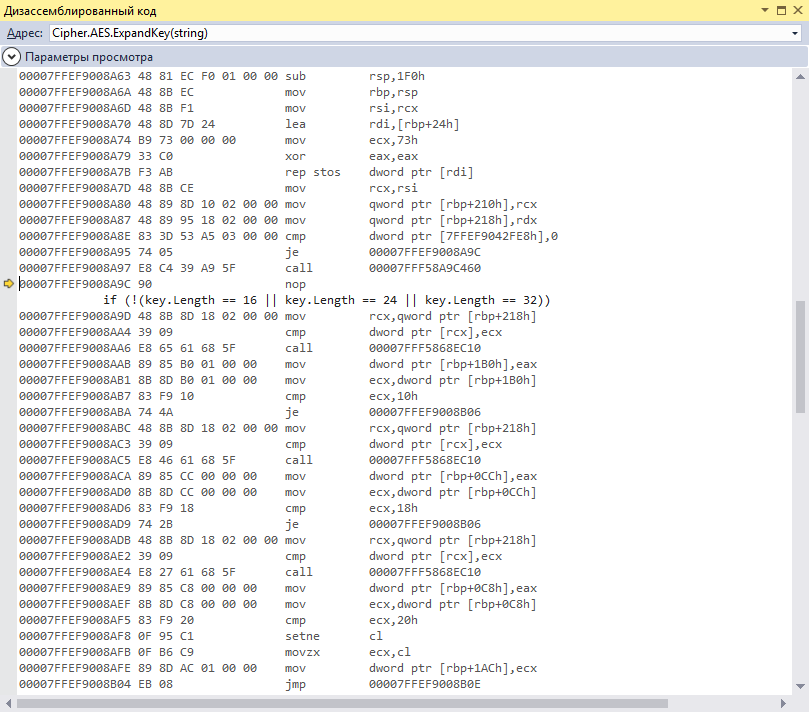
проявам програми.

Для початку шукаємо функцію яка запрашує код шифрування або текст для шифрування (зображення 4.2.2.1), назвемо цю функцію - f1. Далі функціїя f1 відповідає за виклик функції процесу шифрування, назвемо її функція f2 (зображення 4.2.2.2), та передачі до цієї функції ключа шифрування та тексту що треба зашифрувати. При наступному запуску программи в функцію f2. При виконані функції я побачив що, ця функція викликає інші функції, назвемо її f3 (зображення 4.2.2.3). Далі функція f3 викликає функцію яка відповідає за, створення ключа шифрування (зображення 4.2.2.4)

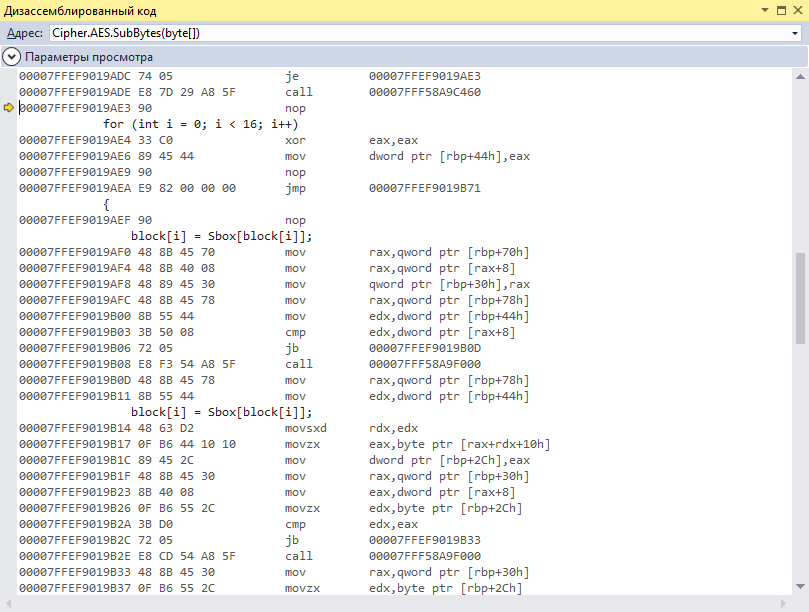


Зображення 4.2.2.1 – Функція f1

Зображення 4.2.2.2 – Функція f2



Зображення 4.2.2.3 – Функція f3



Зображення 4.2.2.4 – Функція створення ключа шифрування

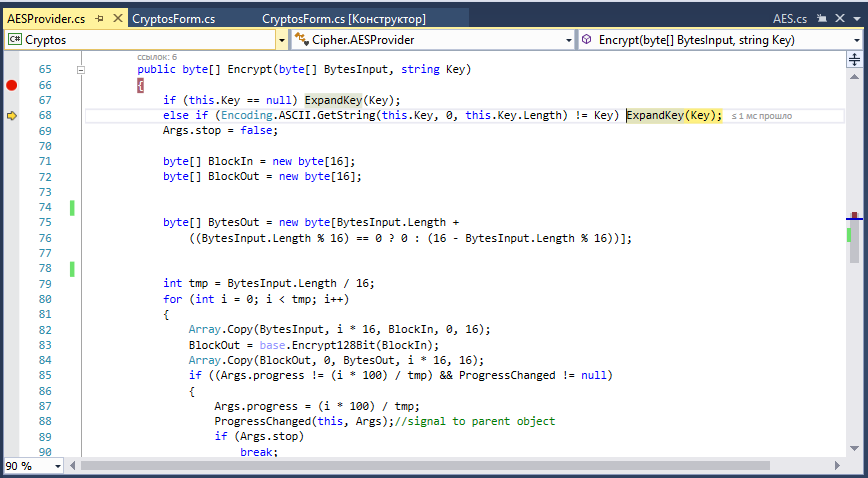
# 4.2.3. Результати застосування метод апаратної точки зупинки уник – 23,31

Цей метод доцільно використовувати в якості продовження

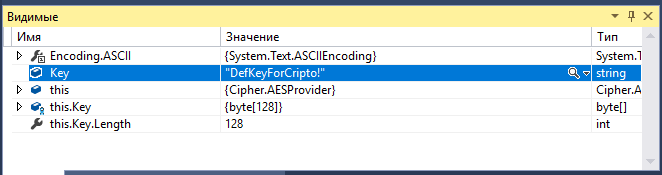
методу маяків. По закінченні першого етапу нам вже відомо розташування в пам'яті буферів з важливими даними в певні моменти виконання програми. Потрібно виявити в програмі функції, що заповнюють чи аналізують ці буфера.

Суть методу полягає в установці на цікавлять аналітика дані апаратної точки зупину, що спрацьовує при читанні даних з буфера або запису даних в буфер залежно від того, що відбувається з буфером в процесі роботи аналізованої програми. Як тільки дані змінені або прочитані процесором, відладчик зупиняється на команди, наступної за тією, яка змінила або використовувала цікаві дані. Залишається з'ясувати характер програми роботи з важливими даними і, якщо це нецікаво, продовжити відстеження перетворень цих даних.

Я встановив точку зупинки на функції отримання ключа. (Зображення 4.2.3.1), побачив там такі дані(Зображення 4.2.3.2)



Зображення 4.2.3.1

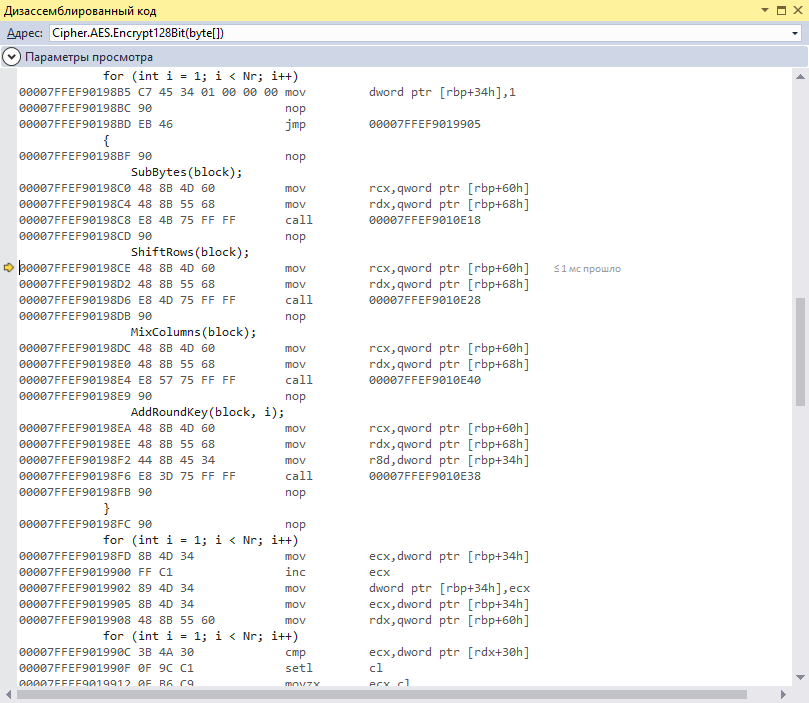


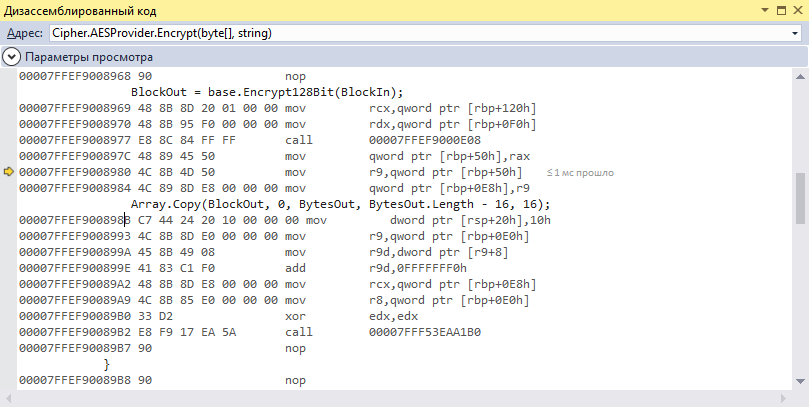
(Зображення 4.2.3.2)

# 4.2.4. Дослідження методом Step-Trace другого етапу

При використанні даного методу на другому етапі аналізу програми трасування починається не з початку програми, а з моменту або проходу маяка, або завершення трасування на першому етапі. Той факт, що функція х була викликана всередині деякого виклику функції, встановлюється не за зовнішнім виглядом екрану комп'ютера, а станом цікавлять нас буферів пам'яті. Якщо в момент проходу виклику деякої функції у режимі Step вміст буферів змінилося, це означає, що функція х викликається всередині даного виклику.

Знаходимо функцію що відповідаэ за шифрування. Це ми бачимо по чотирьом аналогічним блокам які пересилають однакові дані в чотири функції. Зображення4.2.4.1. Після виконаня яких ми маємо змінені дані. Зображення4.2.4.1

Зображення 4.2.4.1- функція шифрування



Зображення 4.2.4.2- Дані після шифрування