***Практичне заняття 3, 4***

*“Хеш-функції та перевірка цілісності інформації”*

Черевач Юрій, МІТ-21

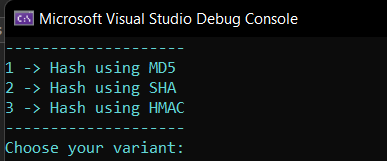
Мета: розібратися із хеш-функцією, алгоритмом MD5, сімейством алгоритмів SHA, навчитися писати програми для роботи з цими методами.

Хід роботи:

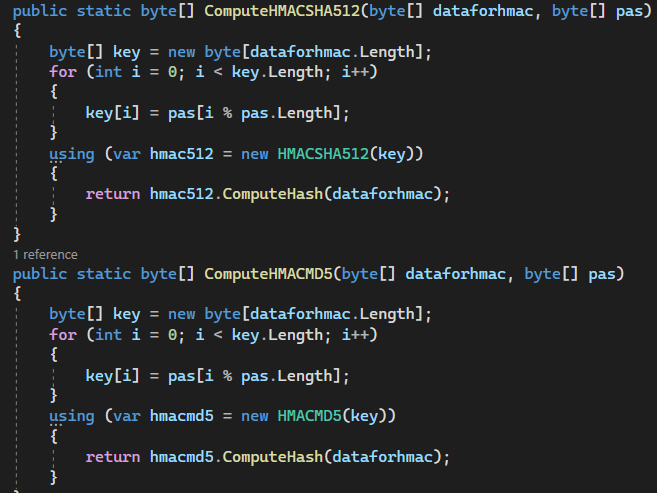
1. Напишемо програму за всіма відомими алгоритмами для заданих даних.

Напишемо методи знаходження різних хешів - MD5, SHA, HMAC. Для цього підключимо необхідні простори імен - System.Security.Cryptography, System.Text.

Створюємо меню, де користувач зможе обрати зручний для нього метод. В мене воно має такий вигляд:



Створимо також функції для обрахування хешу, за кожним методом. Всі вони виглядають на кшталт цих:



Далі використовуємо меню та обраховуємо хеш за кожним значенням.

Для перевірки використовую слова Kyiv та Berlin, і перевіряю їхній хеш, отриманий кожним методом.

|  |  |  |
| --- | --- | --- |
| **Метод/слова** | **Kyiv** | **Berlin** |
| **MDA5** | blKr/K1+NsTuIZyJaoNJAw== | xTOMnMUTXbj9SCi8ilrTJA== |
| **SHA256** | yegDpGLr7PzuvFo9faJ9rgFSkXu4fmf9L5kUnQOzyGk= | s7biCouuN0lhHaRVeDsGUez23hyTP5+58IbbVOyw0Lw= |
| **HMAC1** | cM1w9qGmWkreYbg3icpL0M1aKfk= | eZe2j+1U7GqdcRM8GveJqpnmAGQ= |
| **HMAC256** | hXa4MNUW73J/M/Ph4jCsO9my08m8Je65wdE7Y1oxg3c= | vMs5SVvQtbdui9vWBef7ji3vkTACXZd5Y/a9ry+9eFw= |
| **HMAC512** | 6ppzl4OAYuX+dZxU+4xE9U7TugzP5nDagtImg/t8mEdbKwbuODlm8InBjeWFxw9ozJSrQorlMUfiI7b9TV3f2A== | hn9S7sFMYtu98+4zyAguroOmL+E6WyIkZGYOM98Ygb9wjh4Byfw0jZ7JECU4+YwYELmpGg7Vbk+a26i6+kZaRA== |
| **HMACMD5** | IcRvBZlWB9pC5ztts+k2Tg== | NL2ZPA78xPNbb71cn90D8w== |

Як бачимо, хеш різниться довжиною. Він абсолютно різних у кожному випадку, як і має бути.

Також порівняємо хеш слова Kyiv, отриманий методом HMACMD5 з різними паролями

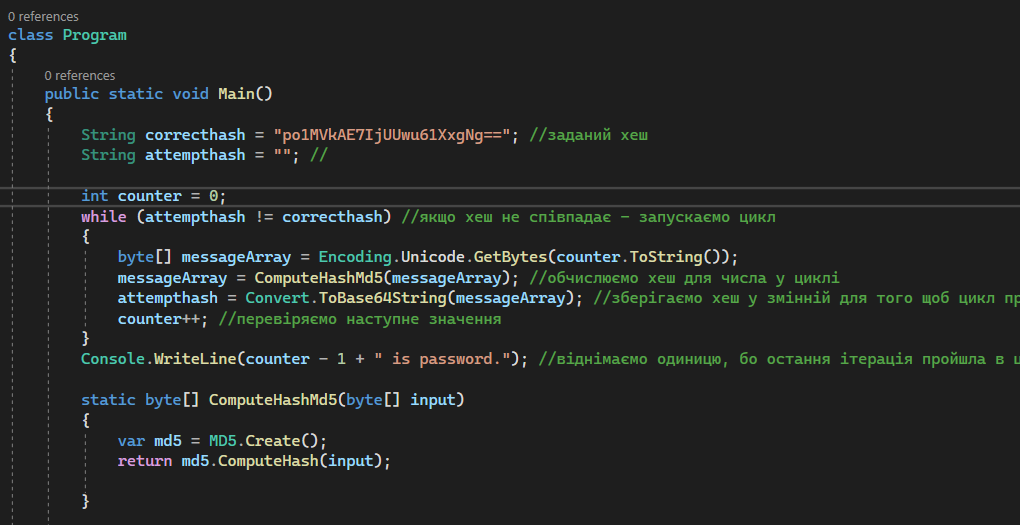
|  |  |  |
| --- | --- | --- |
| **Method/Parols** | **qwerty** | **ukraine00** |
| **HMAC1** | cM1w9qGmWkreYbg3icpL0M1aKfk= | lBX178+s6MboNZENHmrIoSCwl/s= |
| **HMAC256** | hXa4MNUW73J/M/Ph4jCsO9my08m8Je65wdE7Y1oxg3c= | h9DWYlnoW0KGYtwdyPaMuFsMCqJ/qUIoGILsGpf4ATE= |
| **HMAC512** | 6ppzl4OAYuX+dZxU+4xE9U7TugzP5nDagtImg/t8mEdbKwbuODlm8InBjeWFxw9ozJSrQorlMUfiI7b9TV3f2A== | 6i/7rL6lZmqrP9zlh4gnCOoAdbDkWcYo+rWcVnhj7nDVHaldAit2q03hLG1LcRMXAc96smHBR9c2gDJ+iD8zMw== |
| **HMACMD5** | IcRvBZlWB9pC5ztts+k2Tg== | wR5R9tneaupYPSunpadiZg== |

Як ми бачимо, маючи різні паролі окрім довжини хеш не має жодних спільних ознак між собою. Це може бути ознакою вищої надійності розрахунку хешу з паролем.

1. Знайти пароль за заданим хешем.

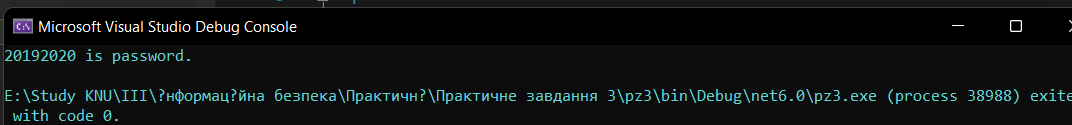
В завданні ми отримали хеш паролю, і нам запропоновано знайти його методом грубої сили. Сам хеш - po1MVkAE7IjUUwu61XxgNg==

І так, код:



Все дуже просто - ми маємо заданий хеш, маємо змінну attempthash для порівняння (в цю змінну ми будемо записувати хеш всіх чисел в ході роботи циклу). Обраховуємо HashMD5 для кожного числа, доки не зустрінемо необхідний нам хеш, і так власне і знайдемо пароль.

Запустивши код, нам необхідно зачекати. У мене все зайняло близько сорока секунд, поки я отримав це повідомлення:



Тобто, 20192020 - шуканий пароль. Можемо зробити висновки, що він **АБСОЛЮТНО НЕ НАДІЙНИЙ.**

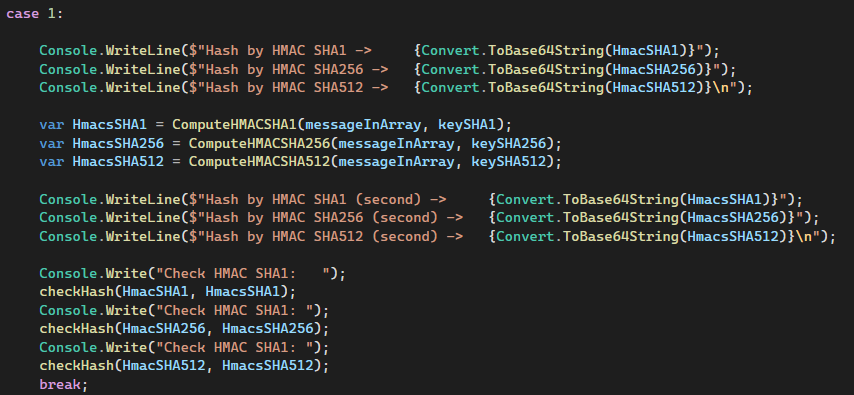
1. **Автентифікація**

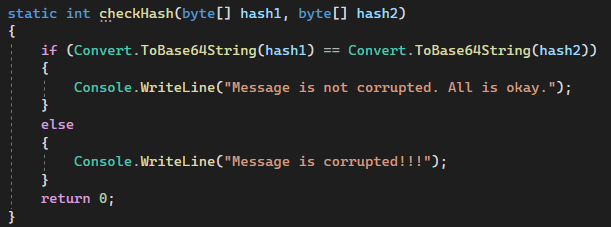
Для автентифікації повідомлення я вирішив використати метод порівняння хешів із секретними ключами.

Спочатку я запитую повідомлення, яке ми будемо перевіряти, а також генерую ключі для трьох методів творення хешу. Потім, власне, створюю хеш із секретним ключем, і вже потім викликаю меню.



Цифра один просто запустить типовий процес автентифікації.



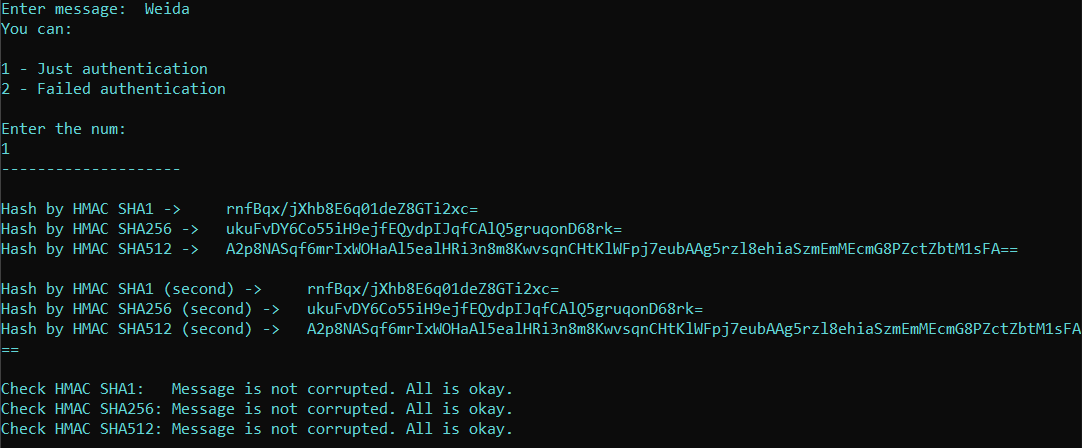


Оскільки в нашій програмі ніяких змін/втрат даних не передбачається, я також описав другий варіант в меню. І тут вже цікавіше.

Я так само знаходжу для повідомлення хеш із секретним ключем, проте, після першого знаходження я змінюю секретні ключі.



І, коли ми викликаємо першу команду в меню, відповідну отримуємо очікуваний результат:



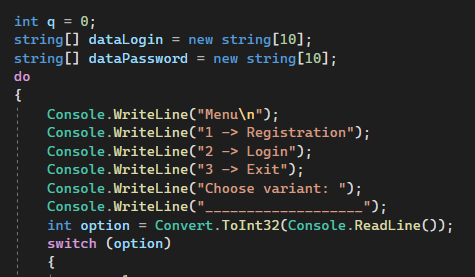
Проте, обравши другий варіант, де ми змінюємо секретні ключі для другої серії хешу, ми отримуємо ось таке:



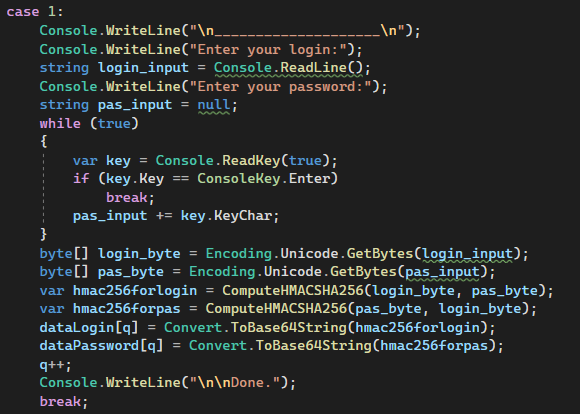
Тобто, цілісність компонентів хешування порушена, тому повідомлення не є захищеним. Це не дивно, адже ми прописали в програмі зміну секретного ключа.

1. Реєстрація користувача за логіном та паролем

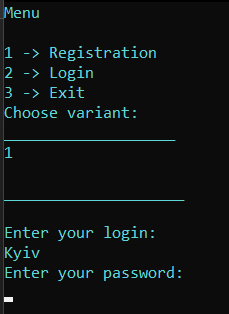
Cтворюємо місце, де будемо зберігати паролі - dataLogin i dataPassword. Також створюємо меню для роботи. Одиниця - реєстрація нового аккаунту (логін + пароль), друге - власне, перевірка логіну та паролю. На трійку ми виходимо з меню. Код меню нижче.



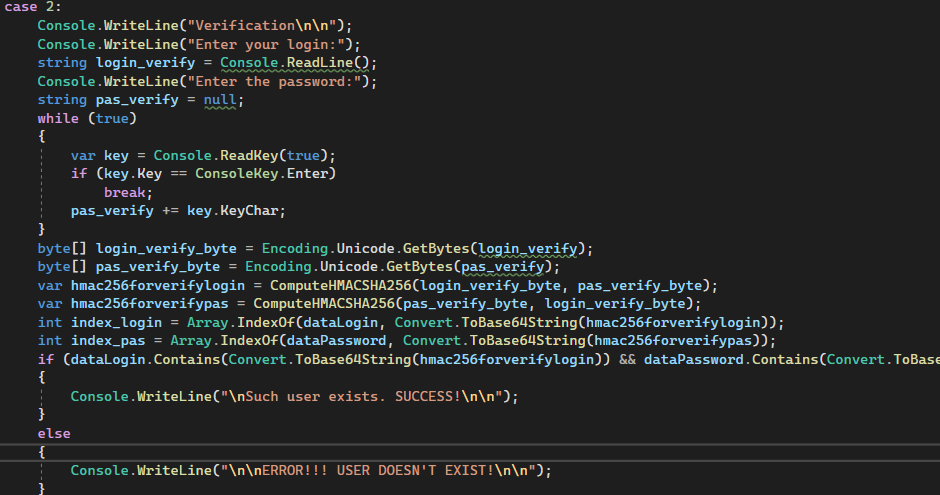
При виборі 1 ми отримаємо від користувача дані - логін та пароль. Далі ми хешуємо ці дані і зберігаємо їх в місці, що прописали перед меню. Збільшуємо q - це як індекс наших даних.



В роботі виглядає це все так:



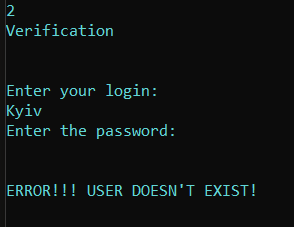
2 відповідає за авторизацію. Ось код:



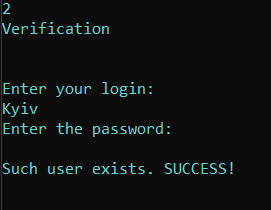
Як ми бачимо, ми так само запитуємо у користувача дані про логін та пароль, вважаючи що попередньо він вже пройшов процедуру реєстрації (1). Далі аналогічно як і у першому пункті ми хешуємо дані, і далі шукаємо даний хеш у dataLogin i dataPassword відповідно. Якщо умова збіжності хешу логіна та пароля, а також індексів цих хешів співпадає, ми розуміємо, що користувач попередньо проходив процедуру реєстрації, і ввів правильні дані, тому він отримає повідомлення про те, що користувач зареєстрований в системі.

Якщо дані не збігаються, виникає помилка, і наш користувач бачить повідомлення про те, що користувача із введеними ним даними у системі не існує.

Зверху ми ввели логін Kyiv та пароль qwerty при реєстрації. Ввівши правильний логін, але неправильний пароль [пароль - “1”] , ми отримуємо наступне повідомлення:



Якщо ввести правильний пароль [пароль - “qwerty”], повідомлення буде таким:



Отже, робимо висновки, що все працює так, як і було задумано спочатку.

Висновки: Виконуючи практичне заняття 3 та 4, я ознайомився з поняттям хеш-функції та областями її застосування, навчився використовувати різні методи знаходження хеш-функцій, а також використовувати їх для перевірки цілісності повідомлень, автентифікації, а також для створення простих систем реєстрації за логіном та паролем.