Семестр ІІІ. Основи інформаційної безпеки

Методичні рекомендації до практичного заняття №5

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

Зміст

Загальні відомості про систему симетричного шифрування 1

Симетричне шифрування за алгоритмом DES і Triple DES..... 2

Алгоритм AES 4

Використання криптографічних бібліотек для здійснення симетричного шифрування..... 5

Вправи для розв'язання..... 6

Короткі теоретичні відомості.

Загальні відомості про систему симетричного шифрування

На минулому занятті ми розглянули хеш-функції, що є односторонніми. Якщо отримати одного разу хеш-код яких-небудь даних, ми не матимемо змоги відновити ці дані, знаючи лише хеш-код. Алгоритм симетричного шифрування, натомість, є двонаправленою операцією, де для зашифровування і розшифровування повідомлення використовується один і той самий секретний ключ (Рис. 1). Ми можемо відтворити оригінальне повідомлення із зашифрованого, використовуючи той самий ключ. Ось чому він називається симетричним шифруванням.

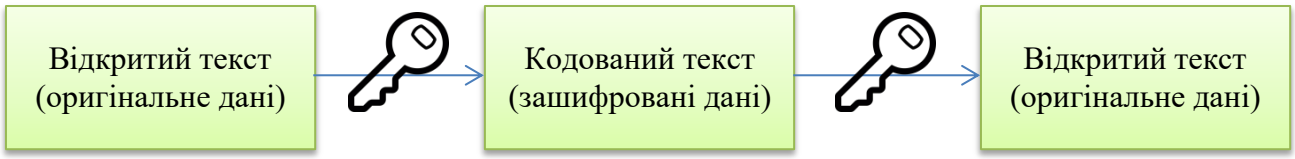

Рис. 1. Симетричне шифрування використовує один і той самий ключ для зашифровування та розшифровування повідомлення.

Симетричне шифрування має як переваги, так і недоліки.

Переваги:

- **Надзвичайно секретний**. Якщо використовувати алгоритм симетричного шифрування такий, як AES, то шифроване повідомлення є винятково секретним. Найбільш поширеною

системою симетричного шифрування у США є урядово-затверджена система Advanced Encryption Standard (AES). На даний момент ця система є не зламаною і є одним із рекомендованих алгоритмів для застосування.

- **Швидкий**. Одним із недоліків систем шифрування з відкритим ключем (наприклад, RSA) є те, що вони потребують складних математичних обчислень, що робить їх повільними. З іншої сторони, зашифровування та розшифровування за допомогою алгоритму симетричного шифрування є легким для виконання, що дає надзвичайну продуктивність зчитування/запису.

Недоліки:

- **Розподіл ключів є складним.** Однією із найважливіших проблем, що пов'язана із застосуванням алгоритмів симетричного шифрування, є те, що нам потрібно мати змогу надати секретний ключ тим особам, з якими будемо обмінюватися конфіденційною інформацією. Ключ шифрування не є звичайним текстом, як, наприклад, пароль. Він є масивом випадкових байтів, що можуть бути згенеровані за допомогою класу RNGCryptoServiceProvider, що розглядався раніше. Тому потрібно мати надійний спосіб передати секретний ключ іншій особі.
- **Небезпека у випадку компрометації ключа.** Коли хто-небудь отримає доступ до секретного ключа, він зможе розшифрувати все, що було зашифроване цим ключем. Якщо ми використовуємо симетричне шифрування для двостороннього обміну, то у випадку компрометації ключа будуть скомпрометовані обидві сторони.

Симетричне шифрування за алгоритмом DES і Triple DES

Алгоритм Data Encryption Standard (DES) був створений в 1970р. в IBM і пізніше, у 1977р., алгоритм був поданий до Національного бюро стандартів (США) і затверджений у якості урядового стандарту забезпечення конфіденційності електронних даних.

Шифрування у DES відбувається 64-бітними блоками (Рис. 2), де вхідні розділяються на 8-ми байтові (64-бітні) ланки, які зашифровуються з використанням 56-бітного секретного симетричного ключа для забезпечення конфіденційності. Хоч DES вже не є рекомендованим стандартом для шифрування на теперішній час, проте залишається багато вже розроблених

додатків, що потребують підтримки та інтеграції, які використовують DES, зокрема у банківській сфері.

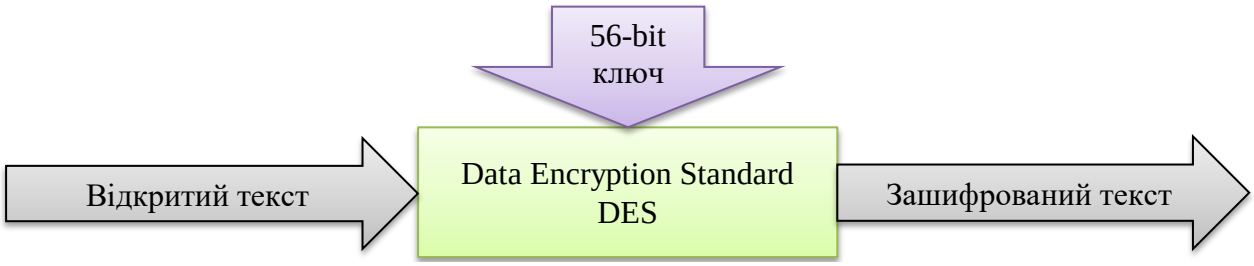

Рис. 2. DES використовує короткий 56-бітовий ключ

За часів, коли стандарт DES був розроблений, 56-bit ключ був достатнім для забезпечення стійкості алгоритму до brute-force атак. Проте із зростанням обчислювальної потужності комп'ютерів також збільшується ймовірність успішної атаки методом повного перебору.

У зв'язку з цим з'явився інший стандарт - Triple DES. Потрійне застосування DES алгоритму дозволило збільшити розмір ключа для захисту від brute-force атак без необхідності розробляти новий алгоритм.

На вхід подається відкритий текст, що зашифровується ключем 1, після чого зашифрований ключем 1 текст зашифровується ключем 2. Результат шифрування ключем 2 зашифровується ключем 3. На виході отримуємо зашифрований текст (Рис. 3).

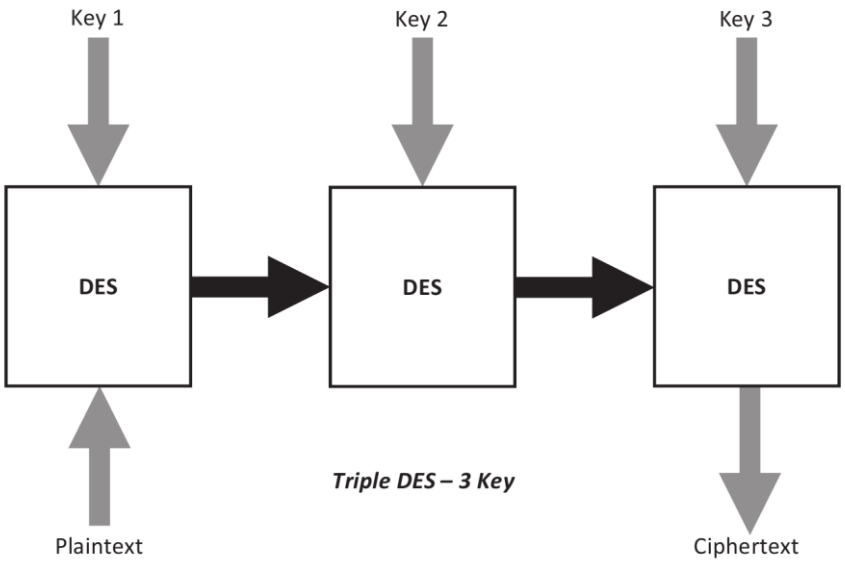

Рис. 3. Алгоритм Triple DES з використанням потрійного ключа.

Інша модифікація алгоритму Triple DES використовує подвійний ключ.

У цьому випадку відкритий текст зашифровується ключем 1, після чого зашифрований ключем 1 текст зашифровується ключем 2. Результат

шифрування ключем 2 зашифровується знову ключем 1. На виході отримуємо зашифрований текст (Рис. 4).

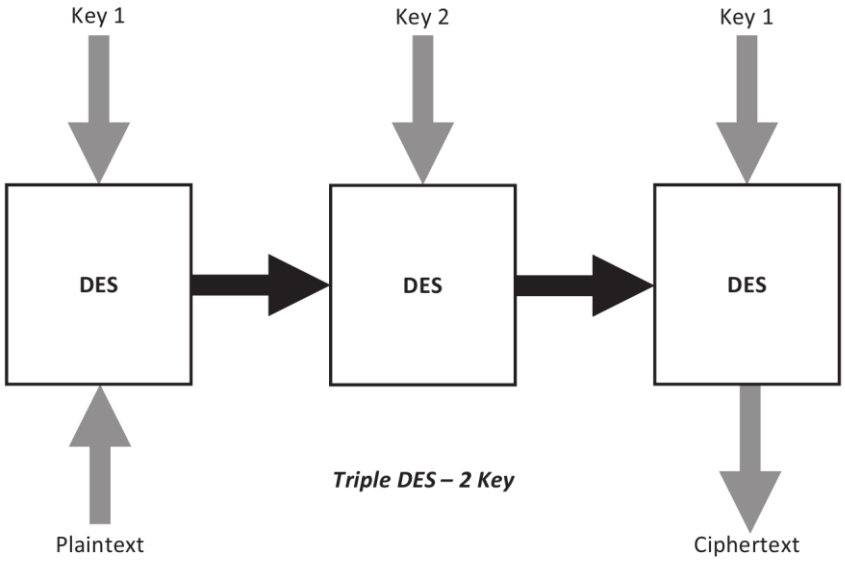

Рис. 4. Алгоритм Triple DES з використанням подвійного ключа.

Алгоритм AES

Алгоритм симетричного шифрування Advanced Encryption Standard (AES) є найактуальнішим урядовим стандартом шифрування даних у США, що прийшов на зміну DES у 2001 році. AES алгоритм базується на шифрі Rijndael (Рейндел), що розробили бельгійські спеціалісти В. Реймен (Vincent Rijmen) й Й. Демен (Joan Daemen). Він характеризується розміром блоку 128 біт, довжиною ключа 128, 192 або 256 біт й кількістю раундів 10, 12 або 14 в залежності від довжини ключа.

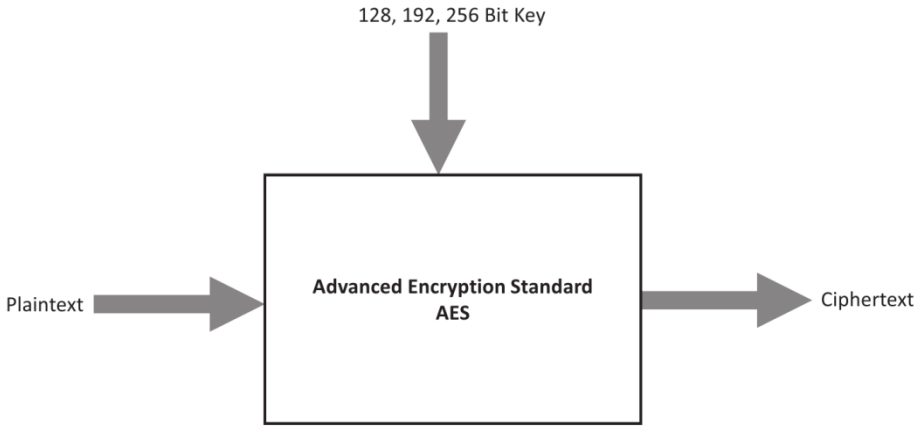

Рис. 5. AES алгоритм використовує ключі трьох різних розмірів

Використання криптографічних бібліотек для здійснення симетричного шифрування

Для нових додатків рекомендується використовувати AES алгоритм для здійснення симетричного шифрування.

У допоміжному класі створюємо метод для генерації секретного ключа та вектора ініціалізації:

```
public byte[] GenerateRandomNumber(int length)
{
  using (var randomNumberGenerator = new RNGCryptoServiceProvider())
  {
    var randomNumber = new byte[length];
    randomNumberGenerator.GetBytes(randomNumber);
    return randomNumber;
  }
}
```

Метод для зашифровування може мати вигляд:

```
public byte[] 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();
        }
      }
}
```

Метод для розшифровування аналогічний до попереднього, за винятком параметра `CreateEncryptor()`.

Зашифровування та розшифровування може бути здійснено наступним чином:

```
var key = aesChipher.GenerateRandomNumber(32);
 var iv = aesChipher.GenerateRandomNumber(16);
 const string original = "Text to encrypt";
 var encrypted = aesChipher.Encrypt(
                Encoding.UTF8.GetBytes(original), key, iv);
 var decrypted = aesChipher.Decrypt(encrypted, key, iv);
 var decryptedMessage = Encoding.UTF8.GetString(decrypted);
 Console.WriteLine("AES Encryption in .NET");
 Console.WriteLine("----------------------");
 Console.WriteLine();
 Console.WriteLine("Original Text = " + original);
 Console.WriteLine("Encrypted Text = " +
                   Convert.ToBase64String(encrypted));
 Console.WriteLine("Decrypted Text = " + decryptedMessage);
 Console.ReadKey();
```

Вправи для розв’язання.

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.
2. 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. **Підказка**: використовувати клас *Rfc2898DeriveBytes*)