

Міністерство освіти та науки України
Харківський національний університет радіоелектроніки
Кафедра програмної інженерії

Лабораторна робота №1
з дисципліни: «Безпека програм та даних»
на тему: «Створення програмної реалізації алгоритму шифрування DES»

Виконав

ст. гр. ПЗПІ-20-1

Бабанін А.К.

Перевірив

доцент кафедри ПІ

Турута О.О.

2023

Мета роботи: Ознайомитись з методами і засобами симетричної криптографії, отримати навички створювання програмних засобів з використанням криптографічних інтерфейсів.

Хід роботи

1. Створення класу-помічника BitArray.

Цей клас використовується для роботи з бітовими масивами та числовими представленнями бітів. Він містить два статичних методи: ToBits та ToDecimal, які відповідають за конвертацію чисел між десятковим та бінарним форматом та навпаки.

```
public static int[] ToBits(int decimalNumber, int numberOfBits)
{
    int[] bitArray = new int[numberOfBits];
    char[] binaryDigits = Convert.ToString(decimalNumber, 2).ToCharArray();

    for (int i = binaryDigits.Length - 1, k = numberOfBits - 1; i >= 0 && k >= 0;
--i, --k)
    {
        bitArray[k] = (binaryDigits[i] == '1') ? 1 : 0;
    }

    return bitArray;
}
```

Метод ToBits приймає ціле число та кількість бітів і конвертує число в бітовий масив, представляючи його в двійковому форматі. В результаті він повертає масив із бітами, де 1 відповідає встановленому біту у двійковому представленні числа, а 0 - нульовому біту.

```

public static int ToDecimal(int[] bitsArray)
{
    string binaryString = string.Join("", bitsArray);
    int decimalValue = Convert.ToInt32(binaryString, 2);

    return decimalValue;
}

```

Метод `ToDecimal` приймає масив бітів та конвертує його в десяткове число. Він об'єднує біти у масиві, перетворює їх у рядок і потім конвертує цей рядок у ціле число у десятковому форматі. Результатом є десяткове число, яке відповідає бітовому представленню.

2. Створення класу `Encryptor`

Клас `Encryptor` є частиною реалізації алгоритму шифрування DES. Цей клас включає в себе різні методи та структури даних, необхідні для виконання операцій шифрування та розшифрування за допомогою DES.

```

private int[] ip = new int[] { 58,50,42,34,26,18,10,2,60,52,44,36,28,20,12,4,
                                62,54,46,38,30,22,14,6,64,56,48,40,32,24,16,8,
                                57,49,41,33,25,17,9,1,59,51,43,35,27,19,11,3,
                                61,53,45,37,29,21,13,5,63,55,47,39,31,23,15,7 };

```

Рисунок 1 - Таблиця початкової перестановки

Ця таблиця визначає перестановку бітів у вхідному блоку перед початком процесу шифрування. Кожен елемент відображає вхідний біт на позицію з індексом, визначеним в цій таблиці.

```

//Circular Left shift Table
private int[] clst = new int[] { 1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 1 };

//Circular Right shift Table
private int[] crst = new int[] { 1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1 };

```

Рисунок 2 - Таблиця циклічного зсуву вліво та вправо

Перша таблиця визначає кількість бітів, на які потрібно здійснити

циркулярний зсув вліво під час кожного раунду шифрування. Кількість зсувів визначається номером раунду. Аналогічно до другої таблиці. Вона визначає кількість бітів, на які потрібно здійснити циркулярний зсув вправо під час кожного раунду дешифрування.

```
private int[] cpt = new int[] { 14,17,11,24,1,5,3,28,15,6,21,10,  
                                23,19,12,4,26,8,16,7,27,20,13,2,  
                                41,52,31,37,47,55,30,40,51,45,33,48,  
                                44,49,39,56,34,53,46,42,50,36,29,32 };
```

Рисунок 3 - Таблиця перестановок стиснення

Ця таблиця використовується для стиску після першого кроку функції Фейстеля. Вона визначає, які біти із результуючого блоку слід включити в наступний раунд.

```
private int[] ept = new int[] { 32,1,2,3,4,5,4,5,6,7,8,9,  
                                8,9,10,11,12,13,12,13,14,15,16,17,  
                                16,17,18,19,20,21,20,21,22,23,24,25,  
                                24,25,26,27,28,29,28,29,30,31,32,1 };
```

Рисунок 4 - Таблиця перестановок розширення

Ця таблиця використовується для розширення блоку даних перед використанням його в функції Фейстеля. Вона додає деякі дубльовані біти, збільшуючи розмір блоку.

```

private int[,] sbox = new int[8, 64] { { 14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,
0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8,
4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0,
15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13 },
{ 15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,
3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5,
0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15,
13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9 },
{ 10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,
13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1,
13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7,
1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12 },
{ 7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,
13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9,
10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4,
3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14 },
{ 2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,
14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6,
4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14,
11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3 },
{ 12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,
10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8,
9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6,
4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13 },
{ 4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,
13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6,
1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2,
6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12 },
{ 13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,
1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2,
7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8,
2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11 } };

private int[] pbox = new int[] { 16,7,20,21,29,12,28,17,1,15,23,26,5,18,31,10,
2,8,24,14,32,27,3,9,19,13,30,6,22,11,4,25 };

```

Рисунок 5 - Таблиці перестановок S та P

DES використовує 8 S-боксів для заміни певних бітів в результуючому блоку після функції Фейстеля. Ці S-бокси містять заміни для різних вхідних значень і виводять відповідні вихідні значення. Таблиця P використовується для перестановки бітів в результуючому блоку після використання S-боксів.

```

private int[] fp = new int[] { 40,8,48,16,56,24,64,32,39,7,47,15,55,23,63,31,
38,6,46,14,54,22,62,30,37,5,45,13,53,21,61,29,
36,4,44,12,52,20,60,28,35,3,43,11,51,19,59,27,
34,2,42,10,50,18,58,26,33,1,41,9,49,17,57,25 };

```

Рисунок 6 - Таблиця кінцевої перестановки

Ця таблиця визначає останню перестановку бітів перед завершенням процесу шифрування. Вона переставляє біти з результату фінального раунду шифрування в заздалегідь визначений порядок Метод StartEncryption виконує основну процедуру шифрування за алгоритмом DES і включає в себе наступні кроки:

- InitialPermutation: Цей крок виконує початкову перестановку бітів у вхідному тексті ptextbitslice за допомогою таблиці перестановки. Це створює початковий стан для тексту перед подальшою обробкою.
- DivideIntoLPTAndRPT: В даному кроці вхідний текст після початкової перестановки розділяється на ліву половину і праву половину, які подальше обробляться окремо.
- AssignChangedKeyToShiftedKey: Цей крок включається для копіювання ключа, який був попередньо змінений під час підготовки, до внутрішнього буфера для подальшого використання.
- SixteenRounds: Цей метод виконує шістнадцять раундів шифрування DES. Кожен раунд включає в себе операції розширення, XOR, заміни S-Box, перестановки і інші операції для обробки лівої і правої половин блоку даних.
- AttachLPTAndRPT: Цей крок об'єднує ліву і праву половини після завершення шістнадцяти раундів, створюючи об'єднаний блок attachedpt.
- FinalPermutation: У кінці шифрування виконується фінальна перестановка, яка використовує таблицю перестановки для отримання кінцевого зашифрованого тексту.

Метод SixteenRounds виконує шістнадцять раундів шифрування DES, що є основною частиною алгоритму DES. Кожен раунд включає наступні кроки:

1. Зберігання правої половини тексту ptRPT в тимчасовому масиві tempRPT.
2. Визначення кількості циклічних зсувів (n) для даного раунду згідно з таблицею циклічних зсувів clst.
3. Розділення ключа на ліву половину SKey і праву половину DKey.
4. Виконання циклічних зсувів лівої і правої половини ключа на n позицій.

5. Об'єднання лівої і правої половини ключа, створюючи підключ для даного раунду.
6. Розширення правої половини тексту на 48 бітів за допомогою таблиці розширення.
7. Виконання операції XOR між стиснутим ключем і розширеною правою половиною тексту.
8. Використання таблиці S-Box для заміни результатів XOR.
9. Виконання перестановки P-Box для отримання нової правої половини тексту.
10. Виконання операції XOR між результатом P-Box перестановки і лівою половиною тексту.
11. Заміна лівої половини тексту на тимчасову праву половину.
12. Виведення значення ентропії для оцінки якості шифрування після кожного раунду.

```

private void SixteenRounds()
{
    int n;
    Console.WriteLine($"Entropy before: {CalcEntropy()}");
    for (int i = 0; i < 16; i++)
    {
        SaveTemporaryHPT(ptRPT, tempRPT);
        n = clst[i];
        DivideIntoCKeyAndDKey();
        for (int j = 0; j < n; j++)
        {
            CircularLeftShift(CKey);
            CircularLeftShift(DKey);
        }
        AttachCKeyAndDKey();
        CompressionPermutation();
        ExpansionPermutation(ptrRPT, ptExpandedRPT);
        XOROperation(compressedkey, ptExpandedRPT, XoredRPT, 48);
        SBoxSubstitution(XoredRPT, ptSBoxRPT);
        PBoxPermutation(ptSBoxRPT, ptPBoxRPT);
        XOROperation(ptPBoxRPT, ptLPT, ptrRPT, 32);
        Swap(tempRPT, ptLPT);
        Console.WriteLine($"Entropy on round {i + 1}: {CalcEntropy()}");
    }
}

```


Наступний метод виконує процес розшифрування шифрованого тексту, використовуючи алгоритм шифрування DES. Основні кроки методу включають наступне:

1. Парсинг вхідних даних: Вхідні дані, які включають в себе шифрований текст та ключ для розшифрування, перетворюються в рядок і масив символів відповідно.
2. Перетворення тексту в бінарний формат: Вхідний шифрований текст `ciphertext` перетворюється в бінарний формат, представлений у вигляді масиву бітів. Після цього виконується додавання нулів, якщо кількість бітів не кратна 64.
3. Перетворення ключа в бінарний формат: Ключ для розшифрування перетворюється в бінарний формат і також доповнюється нулями для відповідності довжини ключа DES.
4. Видалення 8-го біта з ключа: Відбувається видалення 8-го біта з кожного байту ключа DES, так як DES використовує лише 56 бітів з 64 бітів в ключі.
5. Розшифрування кожного блоку: Шифрований текст поділяється на блоки по 64 біти кожен. Для кожного блоку виконується розшифрування.
6. Виклик методу `StartDecryption()`: Для кожного блоку шифрованого тексту виконується розшифрування за допомогою методу `StartDecryption()`, який містить основні етапи розшифрування.
7. Перетворення розшифрованого тексту в рядок: Отримані результати розшифрування конвертуються з бінарного формату назад у текстовий рядок.
8. Результат: Розшифрований текст повертається як вихідний результат методу.

```

Input instruction
5Key used (hex):0000 0000 0000 0000
Is weak:True
Key used (hex):0030 0003 0000 3000
Is weak:False
Key used (hex):0000 0101 0000 1000
Is weak:True
Key used (hex):FFFF FFFF FFFF FFFF
Is weak:True
Key used (hex):FEE0 FEE0 FEF1 FEF1
Is weak:True
Key used (hex):1F01 1F01 0E01 0E01
Is weak:True
Key used (hex):FbE0 FaE0 FEa1 FEa1
Is weak:False
Input instruction

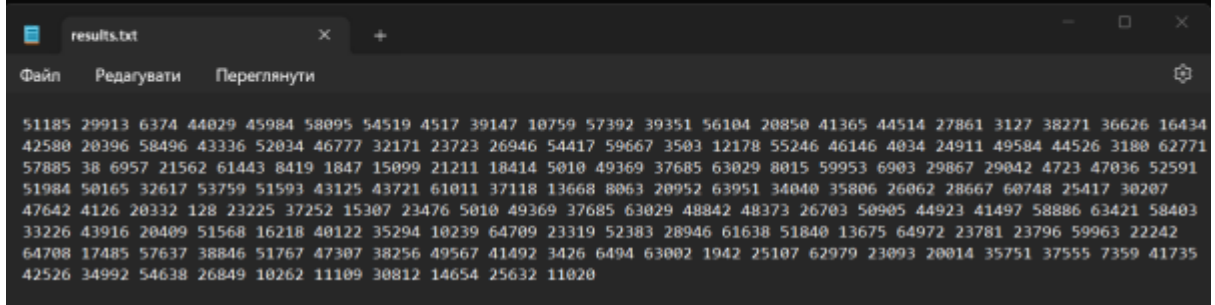
```

Рисунок 7 – Перевірка ключів на криптостійкість

```

Write a text to encrypt:
Ознайомитись з методами і засобами симетричної криптографії, отримати навички створення програмних
засобів з використанням криптографічних інтерфейсів.
-----
Write a key to use:
fdab51253100128f

```



```

51185 29913 6374 44029 45984 58095 54519 4517 39147 10759 57392 39351 56104 20850 41365 44514 27861 3127 38271 36626 16434
42580 20396 58496 43336 52034 46777 32171 23723 26946 54417 59667 3503 12178 55246 46146 4034 24911 49584 44526 3180 62771
57885 38 6957 21562 61443 8419 1847 15099 21211 18414 5010 49369 37685 63029 8015 59953 6903 29867 29042 4723 47036 52591
51984 50165 32617 53759 51593 43125 43721 61011 37118 13668 8063 20952 63951 34040 35806 26062 28667 60748 25417 30207
47642 4126 20332 128 23225 37252 15307 23476 5010 49369 37685 63029 48842 48373 26703 50905 44923 41497 58886 63421 58403
33226 43916 20409 51568 16218 40122 35294 10239 64709 23319 52383 28946 61638 51840 13675 64972 23781 23796 59963 22242
64708 17485 57637 38846 51767 47307 38256 49567 41492 3426 6494 63002 1942 25107 62979 23093 20014 35751 37555 7359 41735
42526 34992 54638 26849 10262 11109 30812 14654 25632 11020

```

Рисунок 8 – Перевірка функціонала шифрування та результат збереження у файлі

```

---- Block 4 started ----
Entropy before: 0,7281342378690551
Entropy on round 1: 0,954434002924965
Entropy on round 2: 0,9936507116910404
Entropy on round 3: 0,9936507116910404
Entropy on round 4: 0,9992954443621549
Entropy on round 5: 0,9992954443621549
Entropy on round 6: 0,9971803988942642
Entropy on round 7: 0,9886994082884974
Entropy on round 8: 0,9992954443621549
Entropy on round 9: 0,9886994082884974
Entropy on round 10: 0,9652016987500656
Entropy on round 11: 1
Entropy on round 12: 0,9936507116910404
Entropy on round 13: 0,954434002924965
Entropy on round 14: 0,9971803988942642
Entropy on round 15: 0,9744894033980523
Entropy on round 16: 0,9992954443621549
---- Block 4 ended ----

```

Рисунок 9 -Розрахунки ентропії для одного з блоків

```

Write a text to decrypt:
51185 29913 6374 44029 45984 58095 54519 4517 39147 10759 57392 39351 56104 20850 41365 44514 27861
3127 38271 36626 16434 42580 20396 58496 43336 52034 46777 32171 23723 26946 54417 59667 3503 1217
8 55246 46146 4034 24911 49584 44526 3180 62771 57885 38 6957 21562 61443 8419 1847 15099 21211 184
14 5010 49369 37685 63029 8015 59953 6903 29867 29042 4723 47036 52591 51984 50165 32617 53759 5159
3 43125 43721 61011 37118 13668 8063 20952 63951 34040 35806 26062 28667 60748 25417 30207 47642 41
26 20332 128 23225 37252 15307 23476 5010 49369 37685 63029 48842 48373 26703 50905 44923 41497 588
86 63421 58403 33226 43916 20409 51568 16218 40122 35294 10239 64709 23319 52383 28946 61638 51840
13675 64972 23781 23796 59963 22242 64708 17485 57637 38846 51767 47307 38256 49567 41492 3426 6494
63002 1942 25107 62979 23093 20014 35751 37555 7359 41735 42526 34992 54638 26849 10262 11109 3081
2 14654 25632 11020

-----
-
Write a key to use:
fdab51253100128f
-----
-
Result: Ознайомитись з методами ? засобами симетричної криптографії, отримати навички створення п
рограмних засобів з використанням криптографічних ?нтерфейсів.

```

Рисунок 10 – Перевірка розшифрування

Висновки: В ході виконання лабораторної роботи було ознайомлено з методами і засобами симетричної криптографії, отримано навички створювання програмних засобів з використанням криптографічних інтерфейсів.