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.

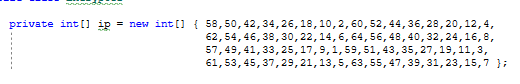


Рисунок 1 - Таблиця початкової перестановки

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.

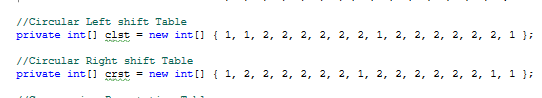


Рисунок 2 - Таблиця циклічного зсуву вліво та вправо

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.

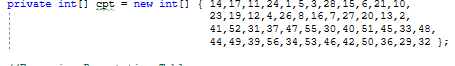


Рисунок 3 - Таблиця перестановок стиснення

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



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.



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.

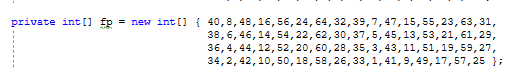


Рисунок 6 - Таблиця кінцевої перестановки

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 CKey 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(ptRPT, ptExpandedRPT);

XOROperation(compressedkey, ptExpandedRPT, XoredRPT, 48);

SBoxSubstitution(XoredRPT, ptSBoxRPT);

PBoxPermutation(ptSBoxRPT, ptPBoxRPT);

XOROperation(ptPBoxRPT, ptLPT, ptRPT, 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.

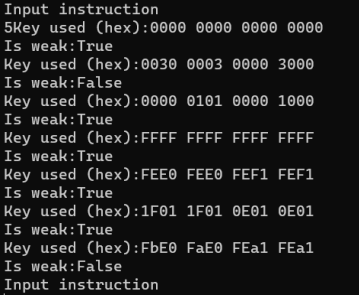


Рисунок 7 – Перевірка ключів на криптостійкість

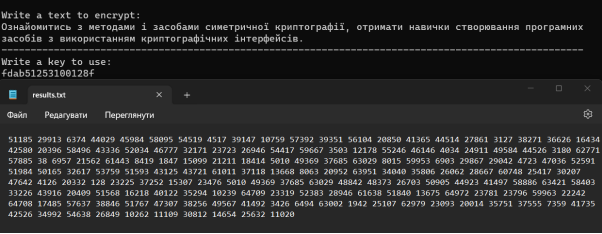


Рисунок 8 – Перевірка функціонала шифрування та результат збереження у файлі

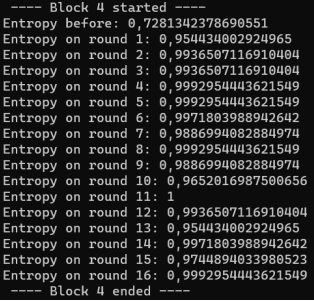


Рисунок 9 -Розрахунки ентропії для одного з блоків

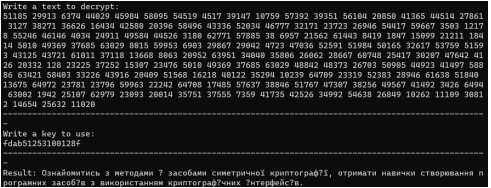


Рисунок 10 – Перевірка розшифрування

**Висновки:** В ході виконання лабораторної роботи було ознайомлено з

методами і засобами симетричної криптографії, отримано навички

створювання програмних засобів з використанням криптографічних

інтерфейсів.