# ЛАБОРАТОРНАЯ РАБОТА № 4

# СУЧАСНІ АЛГОРИТМИ СИМЕТРИЧНОГО ШИФРУВАННЯ

**Мета роботи:** ознайомитись з різними шифрами простої заміни (шифрами підстановки) та методами їх дешифрування.

# Короткі теоретичні відомості

## Алгоритм шифрування DES

Алгоритм DES використовує комбінацію підстановок та перестановок. DES здійснює шифрування 64-бітових блоків даних за допомогою 64-бітового ключа, в якому значущими є 56 біт (інші 8 біт - перевірочні біти для контролю на парність). Дешифрування в DES є операцією, що зворотна шифрування, і виконується шляхом виконання операції шифрування у зворотній послідовності. Процес шифрування полягає в початковій перестановці бітів 64-бітового блоку, 16 циклів шифрування і, нарешті, кінцевій перестановці бітів.

Ключ має довжину 56 біт (як правило, в джерельному вигляді ключ має довжину 64 біти, де кожний 8-й біт є бітом паритету, крім того, ці контрольні біти можуть бути винесені в останній байт ключа). Ключем може бути довільна 64-бітна комбінація, яка може бути змінена в будь-який момент години. Частина цих комбінацій вважається слабкими ключами, оскільки може бути легко визначена. Безпечність алгоритму базується на безпечності ключа.

На найнижчому рівні алгоритм є ніщо інше, ніж поєднання двох базових технік шифрування: перемішування та підстановки. Цикл алгоритму, з якого складається DES є комбінацією цих технік, коли як об'єкти перемішування виступають біти тексту, ключа і блоків підстановок.

Тепер DES вважається нестійким, оскільки: Розмір ключа — 56 бітів — замалий, тому існує реальна загроза пошуку ключа лобовою атакою (послідовним перебором). DES нестійкий до лінійного криптоаналізу (тобто лінійна атака дозволяє знайти ключ DES швидше, ніж послідовний перебір). В той же час, повний 16-раундовий DES стійкий до диференційного криптоаналізу. Через високу розповсюдженість DES було запропоновано багато ідей щодо підвищення його безпеки, зокрема, замінити S-блоки DES новими, стійкими до лінійної атаки. Однак, широке практичне застосування жодна з видозмінених версій DES не набула. Винятком є потрійний DES, однак, це не видозміна алгоритму, а лише особливий режим шифрування звичайним DES.

## Алгоритм шифрування IDEA

IDEA (англ. International Data Encryption Algorithm, міжнародний алгоритм шифрування даних) – симетричний блоковий алгоритм шифрування даних, запатентований швейцарською фірмою Ascom. Відомий тим, що застосовується в пакеті програм шифрування PGP та його вільній альтернативі GnuPG. У листопаді 2000 року IDEA був представлений кандидатом у проекті NESSIE в рамках програми Європейської Комісії IST (англ. Information Societes Technology, інформаційні суспільні технології).

Оскільки IDEA використовує 128-бітний ключ і 64-бітний розмір блоку, відкритий текст розбивається на блоки 64 біт. Якщо таке розбиття неможливо, використовуються різні режими шифрування. Кожен вихідний незашифрований 64-бітний блок ділиться на чотири підблоки по 16 біт кожен, оскільки всі операції алгебри, що використовуються в процесі шифрування, здійснюються над 16-бітними числами. Для шифрування та розшифрування IDEA використовує той самий алгоритм.

Фундаментальним нововведенням в алгоритмі є використання операцій з різних груп алгебри, а саме:

- додавання за модулем ;

- множення за модулем ;

- побітове що виключає АБО (XOR).

Ці три операції несумісні тому, що жодна з них не задовольняють дистрибутивний закон та асоціативний закон.

На рисунку 1.2 зображено приклад шифрування

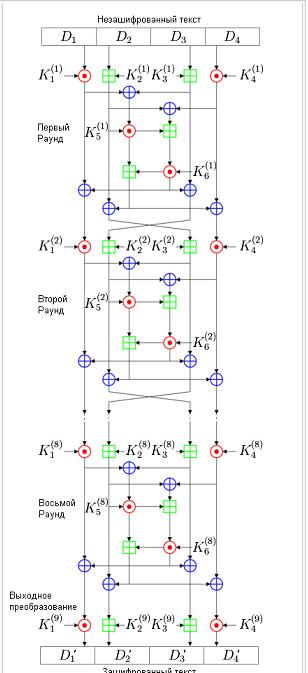


Рисунок 1.2 – Шифрування

Процес шифрування складається з восьми однакових раундів шифрування та одного вихідного перетворення. Початковий незашифрований текст ділиться на блоки по 64 біти. Кожен такий блок ділиться на чотири підблоки по 16 біт кожен. На малюнку ці підблоки позначені D1, D2, D3, D4. У кожному раунді використовуються свої підключення згідно з таблицею підключень. Над 16-бітовими підключами та підблоками незашифрованого тексту виконуються такі операції:

- додавання за модулем ;

- множення за модулем ;

- побітове що виключає АБО (XOR).

В кінці кожного раунду шифрування є чотири 16-бітових підблоки, які потім використовуються як вхідні підблоки для наступного раунду шифрування. Вихідне перетворення являє собою укорочений раунд, а саме, чотири 16-бітних підблоки на виході восьмого раунду та чотири відповідні підключи піддаються операціям:

- додавання за модулем ;

- множення за модулем .

Після виконання вихідного перетворення конкатенація підблоків D1', D2', D3' та D4' є зашифрованим текстом. Потім береться наступний 64-бітний блок незашифрованого тексту та алгоритм шифрування повторюється. Так триває доти, доки не зашифруються всі 64-бітові блоки вихідного тексту.

## Хід роботи

1. Статична модель згідно з варіантом зображено на рисунку 1.3.

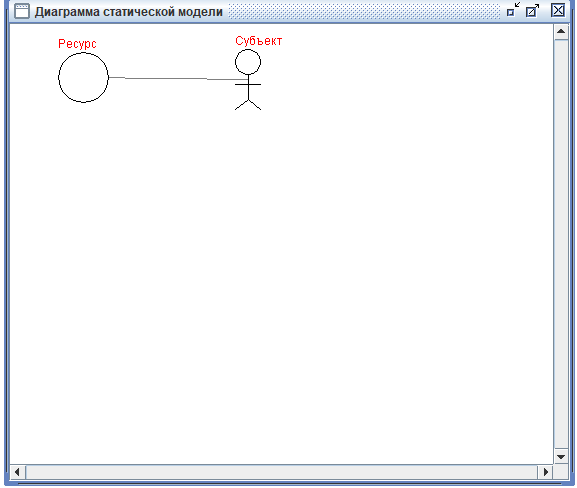


Рисунок 1.3 – Статична модель

2. Вікна редагування ресурсів, суб'єктів, дерева динамічної моделі зображено на рисунку 1.4 – 1.6.

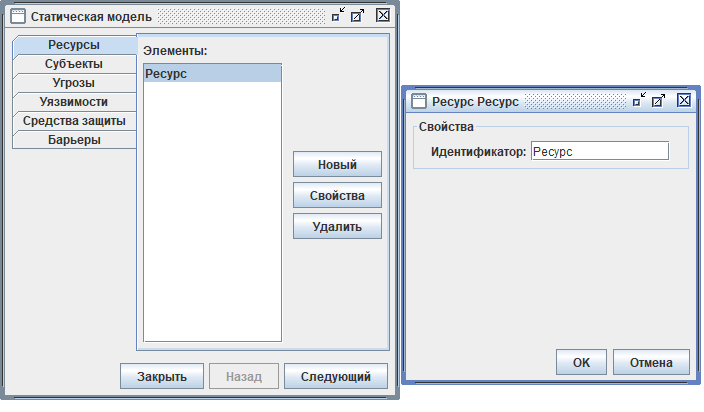


Рисунок 1.4 – Модель ресурсу

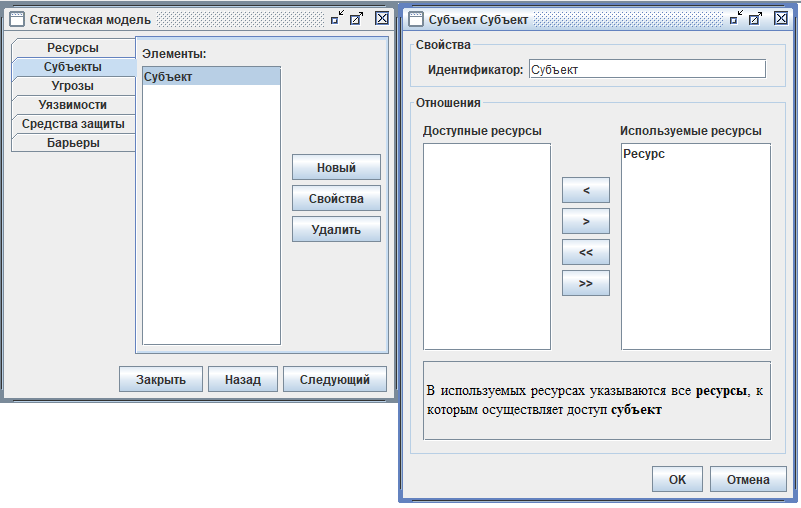


Рисунок 1.5 – Модель суб’єкту

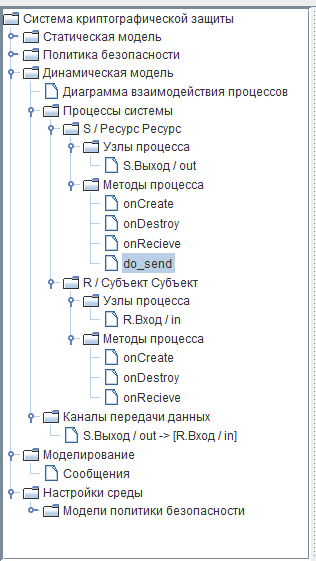


Рисунок 1.6 – Дерево динамической моделі

Програма призначена для кодування та декодування слів шифром DES.

Метод do\_send призначений для шифрування тексту. Код методу наведено в лістингу 1.7.

Лістинг 1.7

DES crypter=new DES();

crypter.setIV(Binary.setFromHex("7836 ECD6 C5F0 37B6"));

crypter.setKey(Binary.setFromHex("9B58 086D 9BF9 CD96 C6EA 3381 B1B4 F637"));

//Crypting

int iWholeTime = 0;

int iDataPtr = 0;

TimeUtil t=new TimeUtil();

while (iDataPtr < iDataSize) {

byte data16[] = new byte[16];

int iLocalI = 0, iLocalI2;

while ((iLocalI < 16) && (iDataPtr < iDataSize)) {

data16[iLocalI] = data[iDataPtr];

iDataPtr++;

iLocalI++;

}

t.start();

byte code[]=crypter.encodeData(data16,crypter.MODE\_CFB);

t.finish();

iWholeTime += t.millisec();

iDataPtr -= iLocalI;

for (int i = 0; i < iLocalI; i++) {

data[iDataPtr] = code[i];

iDataPtr++;

}

}

logDataMessage("Время шифрования ", iWholeTime);

logDataMessage("Зашифрованный текст",data);

send("Выход",data);

Метод onRecieve призначений для розшифрування тексту. Код методу наведено в лістингу 1.8.

Лістинг 1.8

DES crypter=new DES();

crypter.setIV(Binary.setFromHex("7836 ECD6 C5F0 37B6"));

crypter.setKey(Binary.setFromHex("9B58 086D 9BF9 CD96 C6EA 3381 B1B4 F637"));

byte data[]=(byte[])recv("Вход");

int iDataSize = data.length;

//DeCrypting

int iWholeTime = 0;

int iDataPtr = 0;

TimeUtil t=new TimeUtil();

while (iDataPtr < iDataSize) {

byte data16[] = new byte[16];

int iLocalI = 0, iLocalI2;

while ((iLocalI < 16) && (iDataPtr < iDataSize)) {

data16[iLocalI] = data[iDataPtr];

iDataPtr++;

iLocalI++;

}

t.start();

byte code[]=crypter.decodeData(data16,crypter.MODE\_CFB);

t.finish();

iWholeTime += t.millisec();

iDataPtr -= iLocalI;

for (int i = 0; i < iLocalI; i++) {

data[iDataPtr] = code[i];

iDataPtr++;

}

}

logDataMessage("Время расшифрования",iWholeTime);

logDataMessage("Расшифрованный текст",data);

int iDataSize = 1000;

byte data[]=Binary.random(iDataSize);

logDataMessage("Открытый текст",data);

**Висновок:** ознайомились з шифрами IDEA, DES.