DOI 10.37882/2223–2966.2021.03.05

ПРОГРАММНЫЙ АНАЛИЗ ТИПИЧНЫХ АТАК НА АЛГОРИТМ DES

PROGRAM ANALYSIS OF TYPICAL ATTACKS ON THE DES ALGORITHM

A. Berezovsky
A. Romanenkov

Summary. Methods of attacks on the DES algorithm and algorithms that are implemented on the Feistel permutation network are considered. The paper considers algorithms for organizing typical attacks and presents their software implementations. Computational experiments are presented.

Keywords: cryptanalysis, differential cryptanalysis, linear cryptanalysis, LAT tables.

Березовский Александр Дмитриевич
Московский авиационный институт (национальный исследовательский университет), Москва
sashulber@mail.ru
Романенков Александр Михайлович
К.т.н., доцент, Московский авиационный институт (национальный исследовательский университет); с.н.с., ФИЦ «Информатика и управление» Российской Академии Наук, Москва
romanaleks@gmail.com

Аннотация. Рассмотрены методы атак на алгоритм DES и алгоритмы, которые реализованы на подстановочно-перестановочной сети Файстеля. В работе рассмотрены алгоритмы организации типичных атак и представлены их программные реализации. Приведены вычислительные эксперименты.

Ключевые слова: криптоанализ, дифференциальный криптоанализ, линейный криптоанализ, LAT таблицы.

Введение

Алгоритм DES (Data Encryption Standard) — это один из распространённых алгоритмов. На протяжении нескольких лет (1977–1980 г.) был национальным стандартом шифрования США. На замену этого алгоритма пришёл новый алгоритм шифрования данных Rijndail, который является стандартом шифрования США и по сей день. На основе DES предложен новый алгоритм, который называется 3DES, разработанный в 1978 году [1].

При взломе DES в основном использовались следующие атаки:

1. *Метод полного перебора.* Требует одну известную пару шифрованного и расшифрованного текста, его выполнение требует около 2^{55} шагов.
2. *Дифференциальный криптоанализ.* Первую такую атаку на DES заявили Бихам и Шамир. Эта атака требует шифрования 2^{47} открытых текстов, выбранных нападающим, и для её выполнения нужны примерно 2^{47} шагов. Но оказалась не практичной.
3. *Линейный криптоанализ.* Разработан Matsui. Этот метод позволяет восстановить ключ DES с помощью анализа 2^{43} известных открытых текстов. Первый экспериментальный криптоанализ DES, основанный на открытии Matsui, был успешно выполнен в течение 50 дней на автоматизированных рабочих местах 12 HP 9735 [2].
4. *Атака встречей по середине.* Основной принцип атаки заключается в разделении раундов на верхнюю и нижнюю части. Главное условие, от которого отталкивается атакующий — это равенство по середине разделённых раундов [8].

Описание алгоритма DES

DES шифрует *блоки* длиною 64 бит. Алгоритм является типичной сетью Фейстеля с 16 раундами. В каждом раунде используется свой *раундовый ключ,* которые генерируются из основного ключа, вводимого пользователем. В алгоритме DES16 раундовых ключей, которые имеет длину 48 бит [2].

Основная функция шифрования:

- расширяющая перестановка E,
- сложение по модулю 2 с раундовым ключом k_i.
- преобразование S с помощью восьми S блоков $(S_1, S_2, \ldots, S_8)$.
- перестановка P.

Механизм работы сети Фейстеля заключается в преобразовании блоков, которые разбиваются на две части (левая и правая). Правая R_i копируется в левую часть следующего блока L_{i+1}, а L_i складывается по модулю 2 с R_i прошедшую заданные преобразования ра-

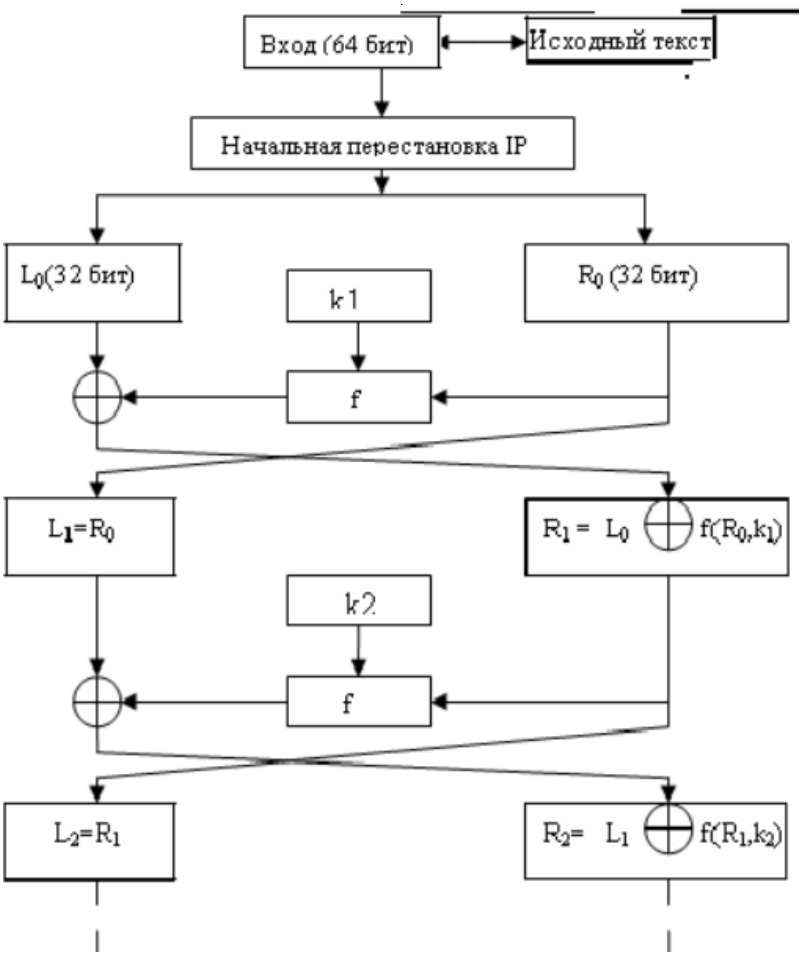

Рис. 1. Схема двух раундов DES

унда f и присваивается правой части следующего блока R_{i+1}. Таким образом циклично преобразуются несколько блоков [3].

Криптоанализ

Атака грубой силой

Всеми известная атака «брутфорс» или по-другому атака перебором ключа. Данная атака не уместна к алгоритмам шифрования подобным шифру Вермана. Так как все перебираемые ключи также перебирают всё возможные открытые тексты [4].

Основная суть атаки заключается в последовательном переборе ключа. Нужно чтобы подбираемый ключ совпал с секретным. Об этом должно сообщить появление определённого сигнала. Сигнал зависит от условий в которых и при которых проходит атака.

Атака грубой силой в своём подавляющем большинстве гарантирует, что данный шифр будет взломан, за некоторое количество времени [4]. Была создана машина DES Cracker организацией Electronic Frontier Foundation, которой удалось взломать алгоритм DES меньше чем за неделю.

Атаке грубой силой невозможно противостоять, не расширяя количество секретных бит. Как и поступили создатели 3DES. Ключ в данном алгоритме составляет 168 бит (эффективные лишь 112 бит), что в разы больше, чем 56 бит обычного DES [1].

Атака линейным анализом

Суть метода состоит в нахождении линейной зависимости начального текста, шифротекста и ключа. После нахождения зависимостей выбирается наиболее эффективная и подбираются оставшиеся биты прямым перебором.

Прежде чем приступить к DES, рассмотрим подстановочно-перестановочную сеть (SP-$сеть$). Пусть x, y — это двоичные вектора размерностью n. Тогда скалярное произведение этих векторов определим формулой:

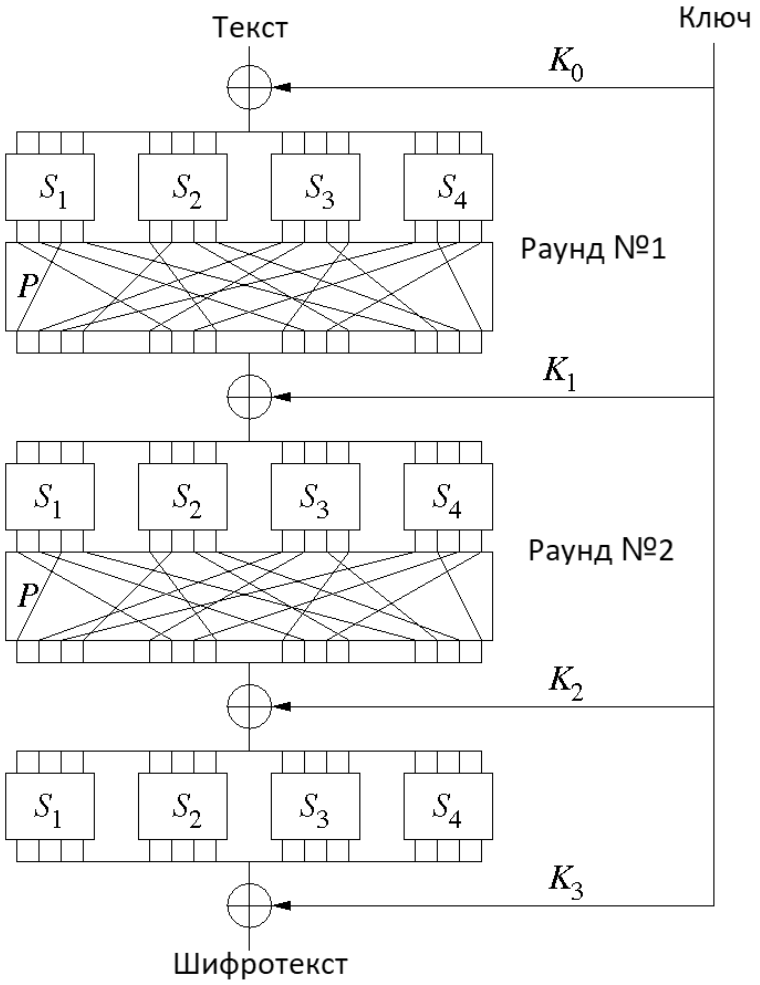

Рис. 2. Схема SP-сети с 4-мя S блоками

$$< x, y >= x_1 \cdot y_1 \oplus \ldots \oplus x_n \cdot y_n \quad (1)$$

где операция (·) — это поразрядная конъюнкция, а (⊕) — это поразрядное исключающие «или» (XOR).

Пусть, далее, P — открытый текст, C — шифротекст, K — ключ, ϵ — *преобладание* линейного соотношения, α, β, γ — выбранные двоичные векторы, которые будем называть *масками*. *Линейным приближением* шифра будем называть:

$$< P, \alpha > \oplus < C, \beta >= < K, \gamma >, \quad (2)$$

который будет выполняться с вероятностью 1/2 + ϵ. Чем больше $|\epsilon|$, тем успешнее будет криптоанализ.

Для наглядности рассмотрим раунд № 1, в котором S блоки $S_1 = S_2 = S_3 = S_4$. Значения для всех S возьмём равные (0,1,13,12,11,10,9,8,7,6,5,4,3,2,14,15). Рассмотрим все возможные приближения S блока $< a, x >$ и $< b, y >$, где a и $b \in Z_2^4$. Теперь рассмотрим линейное приближение, например:

$$X_1 \oplus X_2 = Y_1 \oplus Y_2 \oplus Y_3$$

то есть здесь масками являются вектора $a = (1100) = 12, b = (1110) = 14$. В итоге можно получить таблицу всех возможных вариаций входных данных и выходных. В этой таблице -столбцы — это все возможные вариации левой части линейного приближения, Y-столбцы — это все возможные вариации правой части линейного приближения (рис. 3).

При анализе значений в предпоследнем и последнем столбцах, заметим, что равенство выполняется в 10-ти строчках из 16-ти. Получается, что равенство выполняется с вероятностью в

$$\epsilon_1 = \frac{10}{16} - \frac{1}{2} = \frac{2}{16}.$$

X1	X2	X3	X4	Y1	Y2	Y3	Y4	X1+X2	Y1 + Y2+ Y3
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0
0	0	1	0	1	1	0	1	0	0
0	0	1	1	1	1	0	0	0	0
0	1	0	0	1	0	1	1	1	0
0	1	0	1	1	0	1	0	1	0
0	1	1	0	1	0	0	1	1	1
0	1	1	1	1	0	0	0	1	1
1	0	0	0	0	1	1	1	1	0
1	0	0	1	0	1	1	0	1	0
1	0	1	0	0	1	0	1	1	1
1	0	1	1	0	1	0	0	1	1
1	1	0	0	0	0	1	1	0	1
1	1	0	1	0	0	1	0	0	1
1	1	1	0	1	1	1	0	0	0
1	1	1	1	1	1	1	1	0	0

Рис. 3. Таблица истинности для линейного приближения

a\b	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
2	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
3	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
4	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
5	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
6	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
7	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
8	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
9	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
10	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
11	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
12	0	…	…	…	…	…	…	…	…	…	…	…	…	…	2	…	…
13	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
14	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
15	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
16	0	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

Рис. 4. LAT таблица

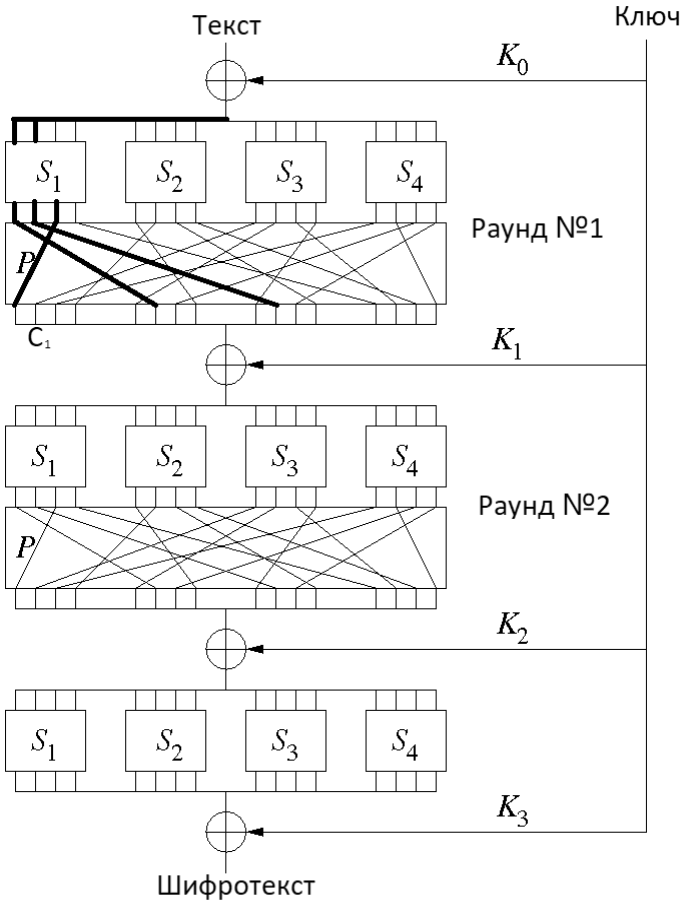

Рис. 5. Схема SP-сети с 4-мя S блоками

Далее строится так называемая LAT таблица для S-блока (*таблица линейного приближения*). В общем случае для *S*-блока это *n*×*n* таблица будет размера 2^n на 2^n, где строки — это векторы a, а столбцы это векторы b (рис. 4), где значения –8, 8, 0 не будут интересовать. Эффективнее брать 4,—4, 6,—6 для дальнейших вычислений. Знак перед значением из ячейки не важен, так как при отрицательном ϵ в линейном приближении (в правой части) добавляется $\oplus 1$.

Рассмотрим структуру шифра 1-го раунда (рис. 5).

Как видно при входе получается $P[1,2]$ (для 1-го S блока), $K_0[1,2]$, а на выходе $C_1[1,6,10]$. Далее на вход в следующий раунд подаётся $C_1[1,6,10]$. В итоге выходит линейное приближение для 1-го раунда:

$$P[1,2] \oplus K_0[1,2] = C_1[1,6,10]$$

это приближение только для одного раунда. Таким образом итерируются эти операции на всех раундах.

Следующим шагом будет рис. 6, то есть система разобьётся на

$$C_1[1] \oplus K_1[1] = C_2[n_1, m_1]$$

$$C_1[6] \oplus K_1[6] = C_2[n_2, m_2]$$

$$C_1[10] \oplus K_1[10] = C_2[n_3, m_3]$$

где n, m нужно посчитать тем же самым способом. Далее формируется линейное приближение уже для 2-го раунда:

$$P[1,2] \oplus K_0[1,2] \oplus K_1[1,6,10] = \\ = C_2[n_1, m_1, n_2, m_2, n_3, m_3]$$

и результирующая вероятность, с которой выполняется данное неравенство ищется с помощью *леммы Мацуи* «о набегании знаков»:

Лемма Мацуи [5]. *Пусть X_i, где — независимые случайные величины, принимающие значения из Z_2. Пусть*

$$P\{X_i = 0\} = \frac{1}{2} + \epsilon_i \qquad (3)$$

где $0 \le \epsilon_i \le \frac{1}{2}$.

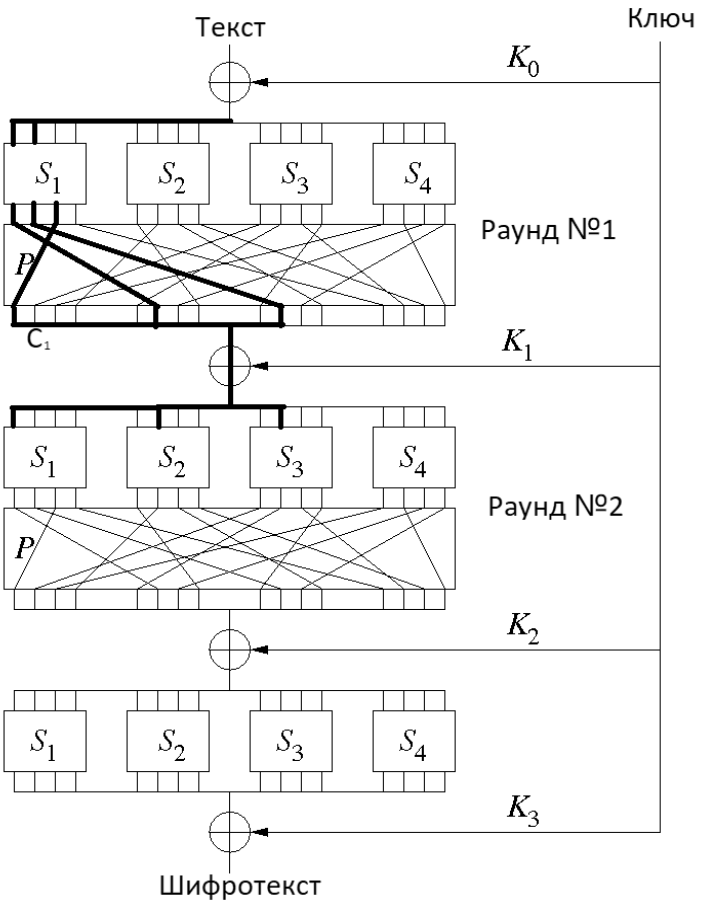

Рис. 6. Схема SP-сети с 4-мя S блоками

Тогда случайная величина $X_1 \oplus X_2 \oplus \ldots \oplus X_n$ *принимает значение 0 с вероятностью*

$\frac{1}{2} + \epsilon$, *где* $\epsilon = 2^{n-1} \prod_{i=1}^{n} \epsilon_i$.

Вероятность для 2-го раунда будет равна

$\frac{1}{2} + 2^3 \left(\frac{2}{16}\right)\left(\frac{t}{p}\right)$, где $\frac{t}{p} -$

это вероятность выполнения равенства во втором раунде, где t — числитель, который взят из LAT-таблицы. И завершающий этап алгоритма имеет несколько своих вариаций. Рассмотрим следующую:

После нахождения окончательного линейного соотношения (назовём его PCK (2), а также вероятности с которой оно выполняется

$$p = \frac{1}{2} + \epsilon$$

для каждой пары P, C вычисляется значение левой части соотношения PCK. Пусть N_0, N_1 будут количество пар статистики, для которых левая часть соотношения равна 0 и 1 (очевидно $N_0 + N_1 = N$).

Положим $< K, \gamma > = 0$, если $(N_0 - N_1) * \epsilon > 0$ и $< K, \gamma > = 1$, если $(N_0 - N_1) \cdot \epsilon \leq 0$. И с учётом этого условия подбирается ключ грубой силой [5].

Теперь отобразим те же действия только **с учётом этапов шифрования DES**:

Первая проблема, с которой можно столкнутся это то, что в начале идёт не объединение с ключом, а преобразование правой части текста матрицей расширения E. Поэтому можно сказать, что вместо R получается E(R), где E — это преобразование матрицей расширения и R — это правая часть текста P, т.е. R → E(R).

Далее будет ровно такая же процедура сложения по модулю 2 с раундовым ключом.

Вторая проблема, с которой можно столкнутся, это необычный вход на S блок:

Входят 6 битов, а выходит всего 4, то есть все возможные вариации входа будут не 4 столбца, а 6. Но для выходов все равно будет 4 столбца. Строк в этом случае будет 64 (это все возможные вариации выхода из S блока).

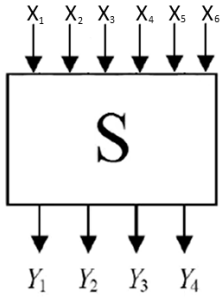

Рис. 7. S блок в алгоритме DES

3	4	5	6	7	8	9	10	11	12	13	14	15
-2	0	-2	4	-6	2	4	2	4	-2	-8	10	-4
2	0	2	-4	6	-6	0	2	-8	-2	-4	10	0
4	2	-4	0	2	2	0	-4	-6	0	-2	-2	4
8	-2	8	0	2	-2	4	4	-6	8	-2	2	0
0	2	2	6	-10	-2	2	2	-2	0	4	8	20
-8	-2	2	-2	2	2	2	2	-6	0	0	-4	4
-2	0	4	-2	-6	-2	6	-4	4	-2	6	-8	0
6	0	-4	-2	2	6	6	-4	-4	-2	-2	0	0
0	-4	4	0	-4	4	-4	0	4	0	4	0	0
-8	0	4	8	0	-4	0	-4	-4	4	-4	8	-4

Выбранная строчка: 16 Самое эффективное значение в строчке: 20 Самое эффективное значение в таблице: 20

Рис. 8. LAT таблица 5-го S блока

Потом выходные биты просто складываются по модулю 2 с другой 32-х битной частью. Т.е. в итогом соотношении пропадает P и C, а появляются R, F, L. Учитывается, что R_{i-1} становится L_i}, а также $F(R_{i-1}, K_i)[m,\dots] \oplus L_{i-1}$, становится R_i. Линейное приближение будет иметь вид:

$$R_{i-1}[n,\dots] \oplus F(R_{i-1}, K_i)[m,\dots] = K_i[t,\dots]$$

для 1-го раунда.

Известное линейное соотношение **эффективной атаки на DES**:

Для нахождения соотношения Мацуи использовал следующие приближения S блоков:

$$S_5: x_2 = y_1 \oplus y_2 \oplus y_3$$

$$S_1: x_2 = y_1 \oplus 1$$

$$S_5: x_2 = y_1 \oplus y_2 \oplus y_3 \oplus y_4 \oplus 1$$

и в итоге получает вероятности

$\frac{42}{64}, \frac{34}{64}, \frac{52}{64}$.

Тем самым прийдя к результату:

$\epsilon_i = \frac{52}{64} - \frac{1}{2} = \frac{20}{64}$ благодаря слабости S_5 блока.

Мацуи выстроил специальную схему приближения, для нахождения итогового линейного соотношения:

$$R_1[8,14,25] \oplus L_{15}[17] R_1[8,14,25,3] = \\ = K_3[26] \oplus K_4[4] \oplus K_5[26] \oplus K_7[26] \oplus \\ \oplus K_8[4] \oplus K_9[26] \oplus K_{11}[26] \oplus K_{12}[4] \oplus \\ \oplus K_{13}[26] \oplus K_{15}[26]$$

и получил итоговое преобладание ϵ = 0,00000057, что практически полностью уничтожило *криптостойкость* шифра алгоритмом DES [6]. Чтобы убедиться, в 5-м S блоке действительно есть слабость, воспользуемся программой (написанной на C#) и убедимся, что такое значение существует (рис. 8).

Действительно, при значениях a = 010000 и b = 1111 получается самый эффективный проход через S_5 блок.

Рассмотрим код, который может вызвать затруднения в реализации. Случай, когда подготавливается правая часть линейного приближения (а точнее все возможные варианты), это Y_1, Y_2, Y_3, Y_4. По сути, просто

Рис. 9. LAT таблица 1-го S блока

достаются из выбранной матрицы S все возможные варианты, которые заданы константами, где ($y_1 y_2 y_3 y_4$) это значения от 0b0000 до 0b1111:

```
public ulong[] OutputColumns(int[][] s_Matrix)
{
ulong[] Y = new ulong[4] {0, 0, 0, 0};
for (int i = 63; i >= 0; i—)
{
for (int j = 0; j < 4; j++)
{
Y[j] <<= 1;
Y[j] = Y[j] | GetBit((uint)s_Matrix[Glue((uint)i, 6, 1)]
[CuttingMiddleBit((uint)i, 6, 1)], j);
}
}
return Y;
}
// GetBit — ставит n-ый бит на 0-й порядок
// Glue — склеивает левую и правую части битового
массива
// CuttingMiddleBit — вырезает левую и правую
части битового массива
```

С использованием вычислительного эксперимента можно убедиться, что S_5 является самый уязвимый S блок среди всех 8-ми.

Пример таблицы для S_1 блока — рис. 9.

Одной из главных функций является функция вычисления линейного преобразования правой и левой частей (по таблице истинности). Реализация для правой части выглядит следующим образом:

```
public ulong LinearApproximation_RightPart(uint
rightPart,
int[][] s_Matrix)
{
ulong[] Y = OutputColumns(s_Matrix);
ulong answer = 0;
for (int i = 0; i < Y. Length; i++)
{
if (GetBit(rightPart, i) == 1)
{
answer = answer ^ Y[Y. Length — 1 — i];
}
}
return answer;
}
// где answer — это вектор битов правой части
// линейного приближения (уже посчитанный).
```

После получения двух векторов левой и правой части линейного преобразования, значения векторов сравниваются. Количество совпавших значений сохраняется в n. Всего имеется $m = 64$ значений. Подсчитывается значение для одной ячейки в LAT таблице

$$\left(\frac{n}{m}\right).$$

Такие действия выполняются для всех ячеек LAT таблицы. Для определения самого большого значения в таблице (исключая 32), воспользуемся функцией FindEffectiveValuePerTable:

```
public static int FindEffectiveValuePerTable(int[][]
LAT_table)
{
```

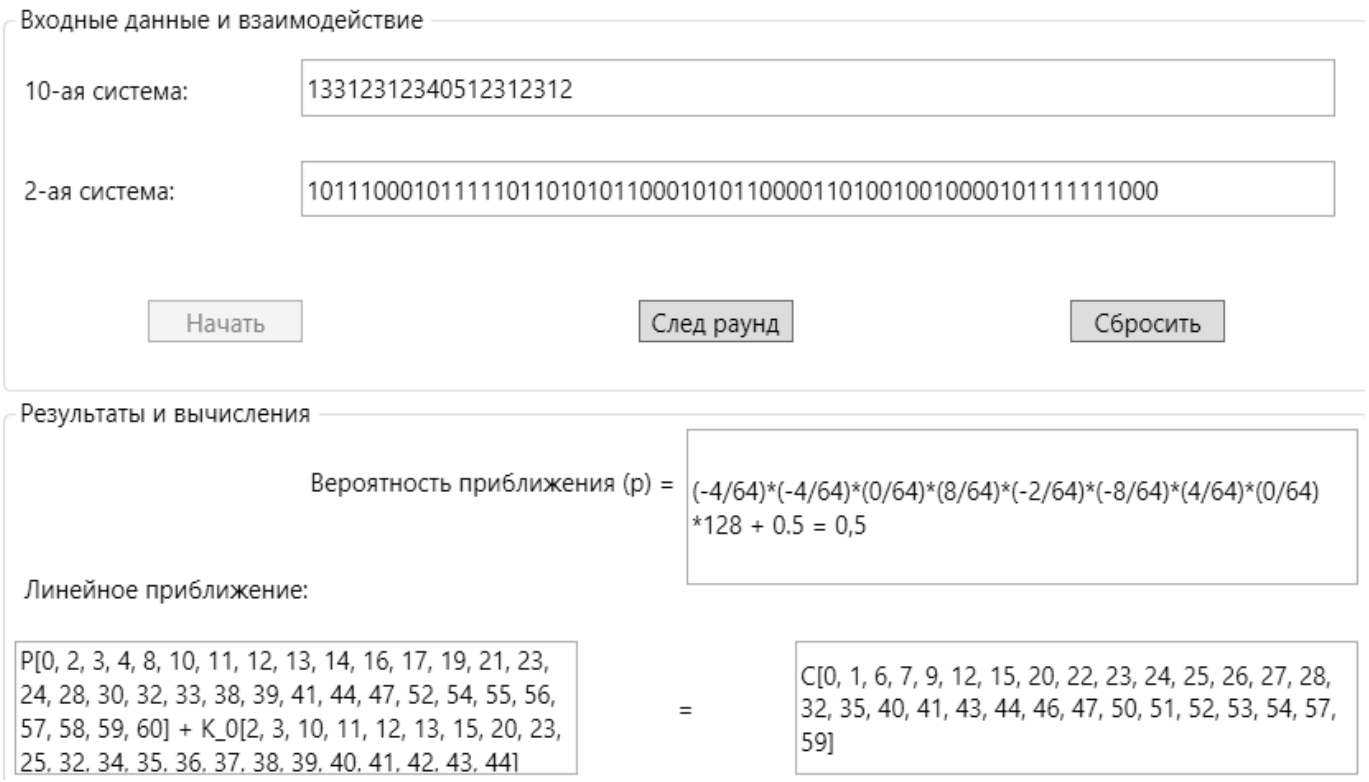

Рис. 10. Вычисление линейной зависимости

```
int answer = 0;
for (int i = 0; i < 64; i++)
{
for (int j = 0; j < 16; j++)
{
if ((Math.Abs(LAT_table[i][j]) > Math.Abs(answer)) &&
(Math.Abs(LAT_table[i][j])!= 32))
{
answer = LAT_table[i][j];
}
}
}
return answer;
}
```

Таким образом можно наглядно увидеть все возможные уязвимости в S блоках. Чтобы вычислить линейную зависимость входного текста и выходного через линейное приближение можно обратиться к другому программному средству. В этом приложении вводится исходный текст и выполняется трассировка по раундам, с вычислением линейного приближения и вероятности линейного приближения. Не учитываются IP перестановки, так как программа работает непосредственно с переходами между раундами DES. Эти перестановки обратимы. Поэтому узнав информацию о входном *раундовом* тексте легко можно применить обратное IP преобразование, чтобы найти входной *первоначальный* текст (рис. 10).

В вероятности приближения (p) два множителя являются нулевыми, поэтому линейное приближение не подходит.

Данная программа не разделяет входные и выходные данные на левую и правую части. Они сразу склеиваются, образуя входные и выходные данные нужного размера в 64 бита.

Рассмотрим главную функцию программы. Эта функция получает входной текст, помещает его в переменную tmp_P, далее в массив ключей _K сохраняются все биты tmp_P, а результаты всех преобразований сохраняются в переменную _C:

```
public ulong LinearApproximation_ForOneRound(
ulong inputText64Bit)
{
ulong tmp_P = inputText64Bit;
uint[] L = new uint[2];
uint[] R = new uint[2];
L[0] = (uint)(tmp_P >> 32);
R[0] = (uint)(tmp_P & (((ulong)1 << 32) — 1));
uint[][][] s_Matrixs = new uint[8][][] {
```

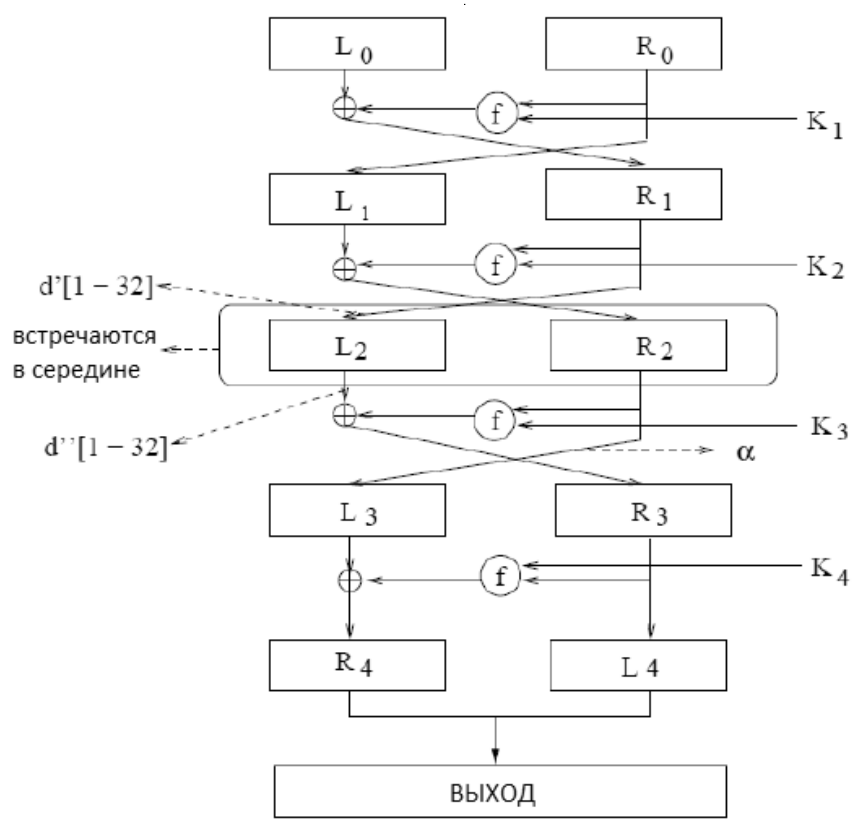

Рис. 11. Схема 4-х раундов DES

```
_S1, _S2,_S3, _S4,
_S5, _S6, _S7,_S8};
tmp_P = BitsSwap(R[0], _E, 32);
R[1] = L[0] ^ P_Transformation(
S_Transformation(tmp_P, s_Matrixs));
L[1] = R[0];
_K.Add(tmp_P); // массив ключей
_C = ((ulong)L[1] << 32) | R[1];
return _C;
}
```

Остальные функции взяты из программы по вычислению LAT таблицы.

Чтобы защититься от данной атаки нужно добавить новые S-блоки для алгоритма с учётом слабостей S_5 блока, а также увеличить размер ключа. Данная атака является самой эффективной для взлома алгоритма DES [7].

Атака встречей по середине

Суть атаки идёт по принципу «разделяй и властвуй», шифр делится на две части (верх и низ), после чего начинаем подбирать ключи с главным условием, что в середине интересующие биты из L_{middle} или R_{middle} [8].

1. Рассмотрим следующую простую схему двойного шифрования, которая вычисляет сообщение зашифрованного текста C из текстового сообщения P используя два ключа K_1, а также K_2:

$$C = E'_{K_2}(E_{K_1}(P)) \quad (4)$$

$$P = D'_{K_1}(D_{K_2}(C)) \quad (5)$$

Путём несложных вычислений можно вывести:

$$C = E'_{K_2}(E_{K_1}(P))$$

$$D'_{K_2}(C) = D'_{K_2}(E'_{K_2}(E_{K_1}(P)))$$

$$D'_{K_2}(C) = E_{K_1}(P) \quad (6)$$

Это и есть основной вывод для использования атаки встречей по середине.

2. Рассмотрим более сложную схему. Пусть M обозначает пространство сообщений, а K обозначает пространство ключей:

Предположим, что $G_k, H_k : M \times K \to M$ — двухблочный шифр $F_k = G_k \circ H_k$. Злоумышленник пытается

вывести K из заданной пары зашифрованного текста $c = F_k(p)$ пытаясь решить

$$G\_k(p) = H_k^{-1}(c) \quad (7)$$

Пусть $d'[1-m] = G\_k(p)\,, d''[1-m] = H_k^{-1}(c),$

G_k состоит из первых двух раундов DES
H_k содержит раунды 3 и 4.

Рассмотрим $d'[9-12]$ и $d''[9-12]$ (в скобках нумерация битов), достаточно угадать только 37 битов ключа. Если $d'[9-12] != d''[9-12]$, ключевое предположение не может быть правильным и отброшено.

Основное наблюдение заключается в том, что значения $d'[9-12]$ и $d''[9-12]$ могут быть вычислены путём угадывания меньшего количества ключевых бит в обмен на угадывание внутренних битов (E - функция расширения).

$$d'[9-12] = L\_0[9-12] \oplus \\ \oplus\ S\_3[E(R\_0)[13-18] \oplus K\_1[13-18]]$$

$$d''[9-12] = L\_4[9-12] \oplus \\ \oplus\ S\_3[E(L\_3)[13-18] \oplus K\_3[13-18]]$$

Замечание: Устанавливается зависимость битов [9–12] и [13–18] после функции расширения текста (и функции сжатия ключа). Так в 32-ух битах текста такие биты под номерами [9–12] будут находится между [13–18] в 48 расширенных битах текста.

Пусть $L_3 = [\alpha_1 \ldots\ldots \alpha_{32}]$, тогда
$E(L_3)[13-18] = [\alpha_{17}\alpha_1\alpha_{15}\alpha_{23}\alpha_{26}\alpha_5]$.

Рассмотрим α_{17}. можно было угадать все 37 предложенных ключевых битов, кроме 6 битов, составляющих $K_4[25-30]$. Для каждого предположения из 31 бита ключа злоумышленник пробует две возможности α_{17}. Если для обоих значений равенство $d'[9-12] = d''[9-12]$ не достигается, то предположение о 31 бите обязательно неверно [9].

Алгоритм хорошо себя показал при атаке на 3DES (нежели чем на DES), тем самым уменьшив эффективность его ключа с 168 до 112 бит [1]. Степень защиты атаки «встреча по середине» напрямую зависит от количества раундов. Увеличив количество раундов, увеличится и стойкость от таких атак.

Атака дифференциальным анализом

Основная идея данного метода состоит в изучении преобразований *дифференциалов (разностей)* между шифруемыми значениями на различным этапах шифрования блоков. Дифференциал в данном случае — это результат сложения по модулю 2 некоторых параметров.

Характеристика — это пара дифференциалов, один из которых образован входными значениями некоторого преобразования, а второй — выходными значениями этого же преобразования. Дифференциал на входе преобразования также часто называют входной разностью, а дифференциал на выходе преобразования — выходной разностью. **Входной дифференциал** — это разность, которая подаётся на S блок. **Выходной дифференциал** — это разность, которая получается после выхода с S блока.

Злоумышленник при использовании криптосистемы свободно оперирует с входными и выходными данными.

Непосредственная атака заключается в использовании заранее подготовленных текстах M_1 и M_2, где:

$$\Delta M = M_1 \oplus M_2 \quad (8)$$

и с помощью ΔM пытается определить каким должен быть дифференциал шифротекстов:

$$\Delta C = C_1 \oplus C_2 \quad (9)$$

Данный криптоанализ никогда не обеспечивает достоверное получение дифференциала шифротекстов. В основном это преобразование используется для вскрытия значительной части ключа с достаточной вероятностью, чтобы облегчить грубую атаку прямым перебором.

Предположим, что злоумышленник решил проверить дифференциал 0b10101010. Для этого выполняется генерация произвольного байта X_1, и вычисление:

$$X_2 = X_1 \oplus 10101010$$

Далее атакующий прогоняет X_1 и X_2 через функцию S_{box1} и получает значения Y_1 и Y_2. Для каждой такой пары X_1 и X_2, дифференциал которых равен 0b10101010, атакующий в состоянии получить дифференциал ΔY. Анализируя полученные значения, атакующий выбирает такое значение ΔY, которое имеет большую вероятность возникновения.

Предположим, что из всех m пар X_1 и X_2, в n случаях $Y_1 \oplus Y_2 = 101100$. Таким образом, вероятность того, что при заданном $\Delta X = 0b10101010$, значение $\Delta Y = 101100$, составляет n/m. Это в свою очередь означает, что при заданном

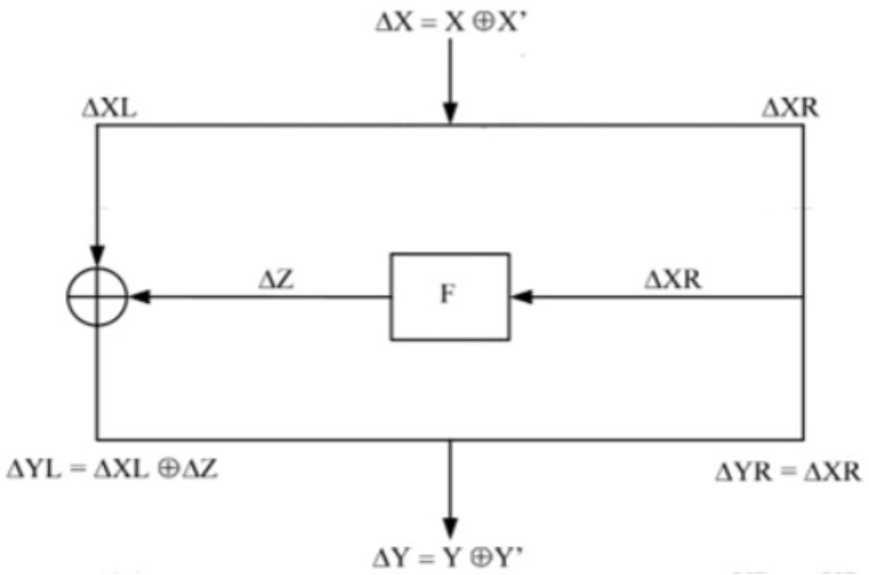

Рис. 12. Дифференциальный анализ 1-го раунда

$$\Delta X = 10101010$$

с вероятностью

$$P_1 = \frac{n}{m}$$

на вход второго раунда попадут два значения U_1 и U_2, такие что $\Delta U = 101100$.

При следующих вычислениях число 101100 подаётся как входной дифференциал для следующего блока S_{box2}. Проводятся те же вычисления для нахождения выходного дифференциала. Итоговая вероятность будет равна

$$P = P_1 \cdot P_2 = \frac{n_1}{m_1} \cdot \frac{n_2}{m_2} \quad (10)$$

Обладая этим фактом, атакующий генерирует несколько пар текстов таких, что $\Delta M = M_1; M_2; M_3; \ldots$ где каждая $M_n = 10101010$ — это и есть блок по 64 бита. Приступает к побайтовому подбору подключа. Вычисляется

$$U\_1 = S_{box}(C_1 \oplus key[0])\ U_2 = S_{box}(C\_2 \oplus key[0]),$$

где key[0] — это первый байт третьего раунда. Подобрав таким образом наиболее вероятный первый байт подключа, атакующий может перейти ко второму байту и действуя аналогичным образом вскрыть весь ключ третьего раунда [10].

Эта атака непрактична из-за чрезмерных требований к подбору данных и сложности организации атаки по выбранному открытому тексту (понадобится много памяти, а также относительно большая скорость обработки). Очевидная защита от таких атак заключается в увеличении размера раундового ключа, а также количества преобразований подобных S блокам [2].

Заключение

В данной работе рассмотрены алгоритмы организации атак на алгоритм DES. Предложены программные решения, которые непосредственно реализуют эти алгоритмы. С помощью этих приложений выполнена визуализация этапов атаки линейным анализом, что позволяет самостоятельно определять уязвимости у классического алгоритма DES. Сгенерированы LAT таблицы для всех S блоков. А также были программно найдены исследованные Мацуи уязвимости.

Подробно проведён анализ 4-х распространённых атак на DES. Рассмотрены возможные варианты защиты от подобных атак.

ЛИТЕРАТУРА

1. 3DES // Википедия. Режим доступа: https://ru.wikipedia.org/wiki/Triple_DES (дата обращения: 24.09.2020).
2. DES // Википедия. Режим доступа: https://ru.wikipedia.org/wiki/DES (дата обращения: 25.09.2020).
3. Biryukov A., Christophe D. Data Encryption Standard (DES). In Encyclopedia of Cryptography and Security. Netherlands: 2011. Vol. 3. 691 p.

4. Смарт Н. Криптография. М.: Техносфера, 2006. 528 с.
5. Matsui M. Linear cryptanalysis method for DES cipher // Computer & Information Systems Laboratory. 1998. Pp. 1–12.
6. P. Patil, P. Narayankar, D.G. Narayan, S.M. Meena. A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowsh // Procedia Computer Science. 2016. Pp. 1–8.
7. Pascal J. On the Complexity of Matsui's Attack // Security and Cryptography Laboratory. 2001. Pp. 1–13.
8. Stephane M. Meet-in-the-Middle Attacks: Autoref. Dis. … Doctor of Engineering Sciences: "Analytical". Thousand Oaks, California, U.S., 2010. 2 p.
9. Li J. Attack on DES: Autoref. Dis. … Doctor of Engineering Sciences: "Analyst". California, 1985. 33 p.
10. Howard M. Linear and Differential Cryptanalysis: Autoref. Dis. … Doctor of Engineering Sciences: "Cryptographer". St. John's, NL, Canada, 2002. 33 p.
11. Grabbe J. The DES algorithm illustrated // Laissez Faire City Times. 1992. Pp. 1–13.

Московский авиационный институт