esse 2

Захар Назаров

January 2024

Содержание

1. Введение

В данной работе дается описание и некоторые аспекты безопасности следующих поточных шифров: DES и 3DES, AES, ГОСТ 34.12-2018 "Магма", ГОСТ 34.12-2018 "Кузнечик", IDEA и IDEA NXT, RC6. Устройство шифра "Магма"и его особенности рассматриваются более детально, также описаны атаки на него.

2. Термины и определения

Определение 2.1. Сеть Фейстеля - это одна из архитектур блочного шифра. Исходный текст делится на блоки, и над каждым блоком производится одно и тоже преобразование. На вход этому преобразованию подается блок и ключ. Вот так выглядит это преобразование:

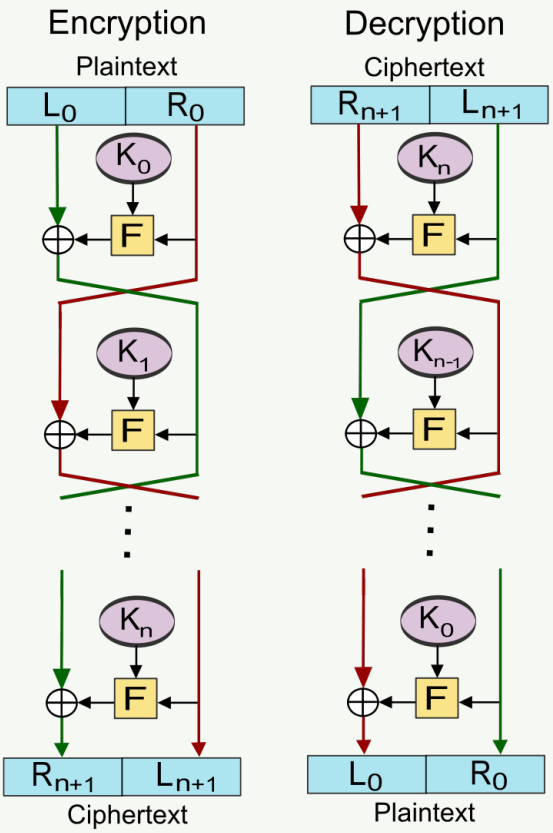

Рис. 1. Сеть Фейстеля [17]

Исходный текст (блок) делится на две части: R_0, L_0. Из ключа создаются подключи, количество которых ровно количеству раундов. После раунда (L_i, R_i) изменяется следующим образом:

1) $L_{i+1} = R_i$
2) $R_{i+1} = L_i \oplus F(R_i, K_i)$

Происходит N раундов, и (L_n, R_N) является выходом преобразования. Причем последний раунд происходит без перестановки:

1) $L_{i+1} = L_i \oplus F(R_i, K_i)$

2) $R_{i+1} = R_i$

Функция F может быть разной. Расшифрование происходит в обратном порядке.

Определение 2.2. S-блок (блок подстановоки, англ. S-box или substitution-box) - математическо преобразование, которое переводит n битов в m битов. n и m могут быть любыми. Обычно это преобразование представляется в виде таблицы.

Определение 2.3. P-блок (блок перестановки, англ. S-box или permutation-box) - математическое преобразование, которое перемешивает биты входной последовательности. По сути, это математическая перестановка.

Определение 2.4. SP-сеть (англ. Substitution–permutation network) - это композиция из идущих друг за другом S-блоков и P-блоков, которая используется в блочном шифровании. Обычно раундом блочного шифра, использующего классическую SP-сеть, называется преобразование блока данных с помощью одного S-блока, одного P-блока и ключа раунда.

Определение 2.5. Атака отражения (reflection attack) - это атака на сеть Фейстеля, которая использует сходство между функциями раунда, и с помощью их строит неподвижные точки полученной функции $f: x : f(X) = x$.

Определение 2.6. Атака "человек посередине"(man-in-the-middle attack, MITM)- атака, предложеная Диффи Хеллманом [13], которая направлена на шифры со следующей конструкцией: $C = enc_{k_2}(enc_{k_1}(p))$. Основная идея атаки, это перебор ключей k_1, k_2 и поиск пар таких что: $enc^{-1}_{k_2}(C) = enc_{k_1}(P)$.

Определение 2.7. Сдвиговая атака (slide attack) - это атака на блочные шифры [14], на основе пар открытого текста и шифротекста, которая ищет такие пары (C, P) и C', P', что $F(P) = P'$, $F(C) = C'$, где F - функция раунда. Далее ищется возможность восстановить весь секретный ключ или его часть из этой пары.

3. Шифры DES и 3DES

3.1. Шифр DES

Шифр DES (Data Encryption Standard, стандарт шифрования данных) - это шифр, выбранный в качестве официального федерального стандарта обработки информации (FIPS) для США в 1976 году и впоследствии получивший широкое распространение на международном уровне.

3.1.1. Описание шифра DES

Блочный шифр DES принимает на вход 2 объекта: 64-битных блок данных и 64-битный секретный ключ. Схема шифра выглядит следующим образом:

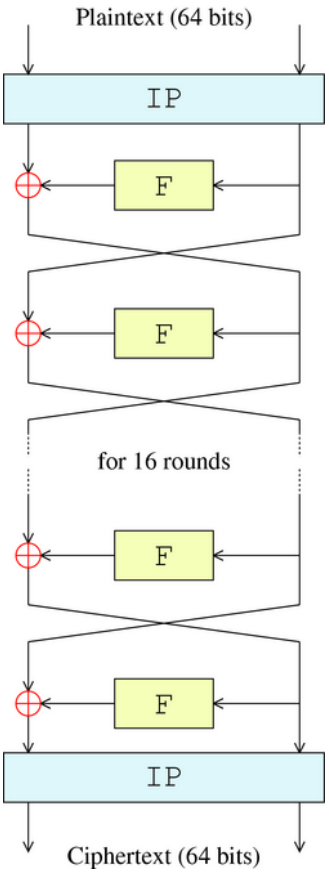

Рис. 2. Устройство DES [1]

Из схемы видно, что шифр состоит из блока инициализирующей перестановки, сети Фейстеля с 16 раундами и финальной перестановки. Рассмотрим функцию F из сети Фейстеля.

Функция F сети Фейстеля

F принимает на вход 32-битный блок данных и 48-битный ключ раунда. Схема фукнции выглядит следующим образом:

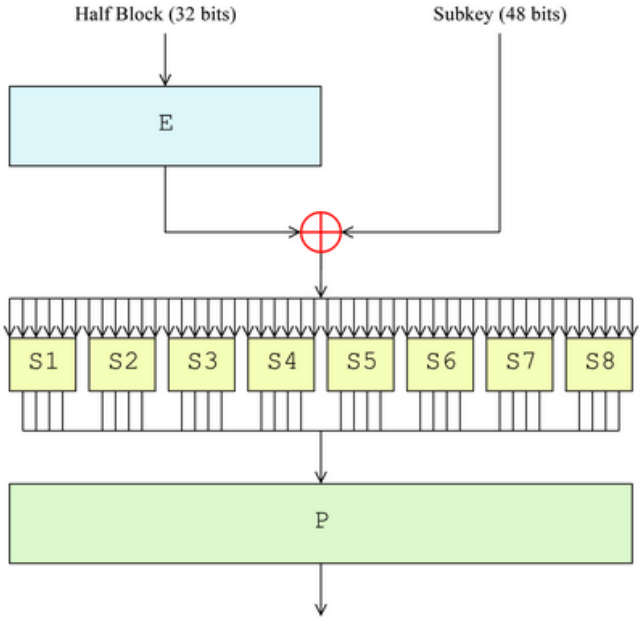

Рис. 3. Функция F шифра DES [2]

Сначала 32-битный блок данных преобразуется с помощью блока расширения E (схема в Приложении этой работы) в 48-битный блок данных, который ксориться (xor) с 48-битным ключом раунда. Затем полученный блок делится на 8 6-битный частей, каждый из которых подается в соответствующий S-box S_i, который преобразует 6 битов в 4 бита. Далее 8 4-битных частей снова собираются в один 32-битный блок, который пропускается через блок перестановки P (схема в Приложении этой работы), выходом которого является 32-битный блок. Полученный блок является выходом функции F.

Вычисление ключей раунда

В DES подается 64-битный секретный ключ, из которого далее для каждого раунда вычисляется ключа раунда. Схема этого вычисления выглядит следующим образом:

Сначала 64-битный ключ подается в блок "выбора из перестановки 1"(Permuted Choice 1, PC1, схема в Приложении этой работы), который выдает 56-битный блок данных. Этот блок делится на 2 28-битных блока KL и KR. Затем каждый раунд ключ раунда $ROUNDKEY$ считается следующим образом:

1) $KL = KL <<< 1$
2) $KR = KR <<< 1$
3) $ROUNDKEY = PC2(KL||KR)$

Блок "выбора из перестановки 2"(Permuted Choice 2, PC2, схема в Приложении этой работы), преобразует 56-битный блок данных, в 48-битный раунд ключа $ROUNDKEY$.

3.1.2. Безопасность шифра

Несмотря на большое количество работ с криптоанализом шифра DES, наилучшей атакой является метод грубой силы. Однако на сегодняшний день такая маленькая длина ключа делает его небезопасным. В 1998 году организация Electronic Frontier Foundation, используя специальный компьютер DES Cracker, вскрыла DES за 3 дня.

3.1.3. Шифр 3DES

Шифр 3DES был создан в 1978 году на основе алгоритма DES с целью устранения главного недостатка DES — малой длины ключа.

Входом шифра является 168-битный ключ k, который разбит на 3 равных ключа (k_1, k_2, k_3). Шифрование 64-битного блока P происходит следующим образом:

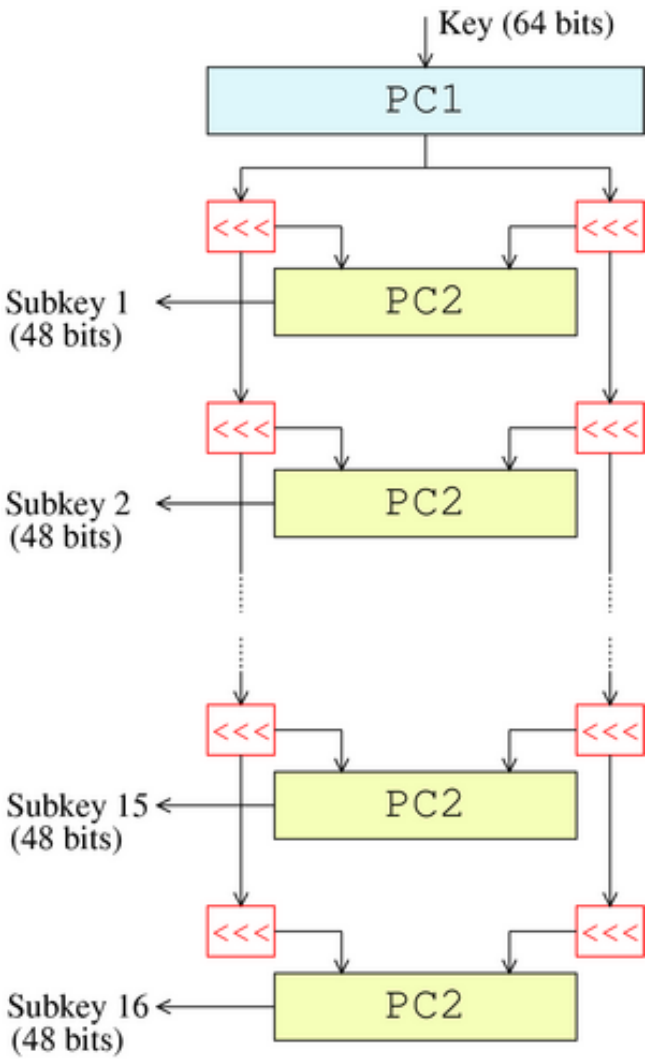

Рис. 4. Расписание ключей шифра DES [3]

$$C = DES_{k_3}(DES^{-1}_{k_2}(DES_{k_1}(P))) \tag{3.1}$$

DES^{-1} это работа алгоритма DES в режиме расшифровки.

Безопасность 3DES

Достаточно старый 3DES остается невзломанным на сегодняшний день. По оценкам НИСТ этот шифр должен остаться надежным до 2030-х годов. Однако у него есть недостаток - это низкая скорость шифрования.

4. Шифр AES

2 января 1997 года NIST начинает искать преемника для DES, являвшегося американским стандартом с 1977 года. 2 октября 2000 года было объявлено, что победителем конкурса стал алгоритм Rijndael, и началась процедура стандартизации. AES (Advanced Encryption Standard) - стандарт шифрования в основе, которого лежит стандартизованный шифр Rijndael.

4.0.1. Описание шифра AES

Симметричный алгоритм блочного шифрования AES [4] - SP-сеть с размером блока в 128 бит и ключом 128/192/256 бит.

Алгоритм оперирует матрицами 4x4, которые получены из 16-байтного (128-битного блока) набора $b_0, b_1, \ldots, b_{15}$. Такая матрица в алгоритме называется состоянием (state):

$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix} \tag{4.1}$$

Количество раундов зависит от длины ключа:

- 10 раундов для 128-битного ключа
- 12 раундов для 192-битного ключа
- 14 раундов для 256-битного ключа

Ознакомимся с базовыми функциями AES: SubBytes. ShiftRows,MixColumns,AddRoundKey.

SubBytes

На вход подается фукнции подается 1 байт (8 бит) из состояния. Многочленом $m(x)$AES в поле $GF(2^8)$ является: $m(x) = x^8 + x^4 + x^3 + x + 1$. Далее входное значение проходит 2 этапа:

1) Находится обратный мультипликативный элемент к входному значению (8 битов интерпретируется как коэффициенты многочлена) в поле $GF(2^8)$ относительно $m(x)$.

2) Пусть $x_0, x_1, \ldots, x_7$ результат с прошлого шага. Далее происходит следующее преобразование:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{4.2}$$

Вектор $(y_0, y_1, \ldots, y_7)$ является выходом функции SubBytes.

ShiftRows

На вход функции подается состояние (state). Первая строка состояния остается неизменной. Вторая, третья и четвертая строка сдвигаются циклически влево на 1 байт, 2 байта и 3 байта соотвественно. Выходом функции является полученное "сдвинутое"состояние.

MixColumns

На вход функции подается состояние (state). Каждый столбец рассматривается как полином 4ой степени: $b_3 * x^3 + b_2 * x^2 + b_1$ в поле $GF(2^8)$. Каждый такой столбец умножается на фиксированный полином $a(x)$ по модулю $x^4 + 1$. $a(x) = 3 * x^3 + x^2 + x + 2$. Полученное новое состояние является выходом функции MixColumns. Также умножение полиномов можно представить в следующем виде:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{4.3}$$

Все операции выполняются по модулю 2^8.

AddRoundKey

На вход функции подается состояние (state) и ключ раунда. Выходом фукнции является сумма по модулю 2 состояния и ключа раунда.

Вычисление ключей раунда

На вход подается 128/192/256 битный ключ, который преобразуется в расширенный ключ длиной равной (количество раундов +1) * (длина блока). Для 128-битного ключа получится: (10 + 1) * 128 = 1408 бит. Далее каждый раунд от этого ключа берутся по 128 бит, которые и будут ключом раунда. Для первого ключа первые 128 бит расширенного ключа, для второго - вторые 128 бит и так далее. Для 128 и 192 битного ключа алгоритм следующий:

```
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)])
{
      for(i = 0; i < Nk; i++)
            W[i] = (Key[4*i],Key[4*i+1],Key[4*i+2],Key[4*i+3]);

      for(i = Nk; i < Nb * (Nr + 1); i++)
      {
            temp = W[i - 1];
            if (i % Nk == 0)
                  temp = SubByte(RotByte(temp)) ^ Rcon[i / Nk];
            W[i] = W[i - Nk] ^ temp;
      }
}
```

Рис. 5. Алгоритм вычисления расширенного ключа из секретного ключа длиной 128 или 192 бит

Для 128 бит Nk = 4, для 192 бит Nk = 6. Для 128 бит Nr = 10, для 192 бит Nr = 12. Первые байты расширенного ключа равны секретному ключу.

Для 256 битного ключа алгоритм следующий:

```
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)])
{
      for(i = 0; i < Nk; i++)
            W[i] = (key[4*i],key[4*i+1],key[4*i+2],key[4*i+3]);

      for(i = Nk; i < Nb * (Nr + 1); i++)
      {
            temp = W[i - 1];
            if (i % Nk == 0)
                  temp = SubByte(RotByte(temp)) ^ Rcon[i / Nk];
            else if (i % Nk == 4)
                  temp = SubByte(temp);
            W[i] = W[i - Nk] ^ temp;
      }
}
```

Рис. 6. Алгоритм вычисления расширенного ключа из секретного ключа длиной 256 бит

Для 256 бит Nk = 8, Nr = 14.

Фукнция SubByte это старое название SubBytes. RotByte делает циклический сдвиг 4-байтного слова на один байт. Rcon (константа раунда) расчитывается так: $Rcon(i) = [rc(i), 00_{16}, 00_{16}, 00_{16}]$. Где rc_i это 8-битное значение, которое считается так:

$$rc_i = \begin{cases} 1 & \text{if } i = 1 \\ 2 \cdot rc_{i-1} & \text{if } i > 1 \text{ and } rc_{i-1} < 80_{16} \\ (2 \cdot rc_{i-1}) \oplus 11\,\mathrm{B}_{16} & \text{if } i > 1 \text{ and } rc_{i-1} \geqslant 80_{16} \end{cases} \tag{4.4}$$

Основной алгоритм AES

Теперь приведем основной алгоритм шифра AES [4]. На вход ему подается 128-битный блок данных (State) и 128/192/256-битный ключ (CipherKey). Функция раунда выглядит следующим образом:

```
Round(State,RoundKey)
{
ByteSub(State);
ShiftRow(State);
MixColumn(State);
AddRoundKey(State,RoundKey);
}
```

Рис. 7. Функция раунда алгоритма AES [4]

Функция финального раунда отличается тем, что в ней нет функции MixColumns. Финальный раунд выглядит следующим образом:

```
FinalRound(State,RoundKey)
{
ByteSub(State) ;
ShiftRow(State) ;
AddRoundKey(State,RoundKey);
}
```

Рис. 8. Финальный раунд алгоритма AES [4]

Основной алгоритм шифра AES выглядит следующим образом:

```
Rijndael(State,CipherKey)
{
KeyExpansion(CipherKey,ExpandedKey) ;
AddRoundKey(State,ExpandedKey);
For( i=1 ; i<Nr ; i++ ) Round(State,ExpandedKey + Nb*i) ;
FinalRound(State,ExpandedKey + Nb*Nr);
}
```

Рис. 9. Шифр AES [4]

4.0.2. Безопасность шифра

На сегодняшний день не было опубликовано реальных атак на шифр AES. В 2003 году агенство национальной безопасности США постановило, что шифр AES является достаточно надежным, чтобы использовать его для защиты сведений, составляющих государственную тайну [6].

5. Шифр ГОСТ 34.12-2018 "Магма"

В марте 1978 года после предварительного изучения опубликованного в 1976 году стандарта DES началась разработка алгоритма шифрования в рамках темы «Магма» (защита информации криптографическими методами в ЭВМ ряда Единой Системы). Позже алгоритм был утверждён постановлением Госстандарта СССР № 1409 от 2 июня 1989 года.

5.0.1. Описание шифра "Магма"

Шифр "Магма"- это сеть Фейстеля с 32 раундами, которая принимает на вход 64-битный блок данных. Длина секретного ключа - 256 бит, длина ключа раунда - 32 бита.

Функция F сети Фейстеля

Преобразование F в Магме выглядит следующим образом:

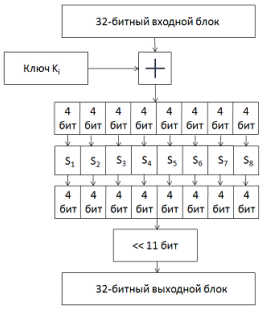

Рис. 10. Функция F в Магме [7]

В F поступают 32-битный блок и 32-битный ключ раунда. Сначала они перемешиваются с помощью побитового сложения по модулю 2, затем полученный блок бьется на 8 разных частей и к каждой части применяется свой S-box. Выходы S-box'ов обратно собираются в один блок, который сдвигается циклически влево на 11 бит и далее подается на выход F.

Вычисление ключей раунда

На вход подается 256-битный секретный ключ $K = k_0||\ldots||k_{256}$. Ключ раунда i вычисляется по следующей формуле:

$$\begin{aligned} k_{8-i} = k_{32*(i+1)-1}||\ldots||k_{32*i}, i = 0, 1, \ldots, 7 \\ k_{i+8} = k_i, i = 1, 2, \ldots, 8 \\ k_{i+16} = k_i, i = 1, 2, \ldots, 8 \\ k_{i+24} = k_{9-i}, i = 1, 2, \ldots, 8 \end{aligned} \tag{5.1}$$

5.0.2. Эксплуатационные и технические характеристики

В стандарте [8] изначально предусмотрели несколько режимов его работы. Выше описан шифр Магма, работающий в режиме простой замены.

Режим гаммирования

Блочный шифр Магма может работать как поточный шифр, создавая ключевую последовательность. Для этого необходимо задать синхропосылку 64-битную синхропосылку S, которая будет открыто передаваться вместе с шифротекстом. Алгоритм работы шифра в режиме гаммирования выглядит следующим образом:

1) Синхропосылка S заносится в регистры N_1 и N_2 шифруется с помощью Магмы в режиме простой замены с секретным ключом K. Полученное значение переписывается в вспомогательные 32-битные регистры N_3 и N_4.
2) $N_3 = N_3 + C_2 mod 2^{32}$, где $C_2 = 1010101_{16}$.
3) $N_4 = N_4 + C_1 mod(2^{32} - 1)$, где $C_1 = 1010104_{16}$.
4) Затем N_3 и N_4 в рабочие регистры шифра N_1 и N_2 (левая и правая часть 64-битного блока данных) соответственно. Этот блок шифруется с помощью шифра в режиме простой замены. Полученный блок является первой 64-битной гаммой.
5) Следующие гаммы высчитываются шагами 2-4 в соответствии с длиной шифруемого текста.

Особенностью этого режима является то, что если поменять один бит зашифрованного текста, то поменяется одни расшифврованного текста. При обычном режиме такого эффекта нет.

Режим гаммирования с обратной связью

Этот режим похож на предыдущий. Основное отличие в том, что он подмешивает в текущую гамму шифротекст с предыдущего шага. Пусть у нас есть синхропосылка S. Алгоритм выглядит так:

1) Синхропосылка S заносится в регистры N_1 и N_2

2) Содержимое регистров N_1 и N_2 шифруется в режиме простой замены. Полученный блок является гаммой.

3) $N_4 = N_4 + C_1 mod(2^{32} - 1)$, где $C_1 = 1010104_{16}$.

4) Гамма подмешивается в блок открытого текста. Полученный блок идет на выход как шифротекст, и также заносится в регистры N_1 и N_2 для расчета следующей гаммы.

5) Следующие гаммы высчитываются шагами 2-3 в соответствии с длиной шифруемого текста.

Режим выработки имитовставки Это особый режим работы, который считает преобразует шифротекст в последовательность длины L бит ($1 <= L < 32$). Этот "слепок"открытого текста может отправляться вместе с шифротекстом, полученным в одном из режимов, описанных выше, для того, чтобы проверять целостность пришедших данных. Так выглядит схема работы шифра в этом режиме:

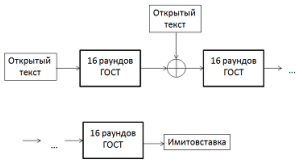

Рис. 11. Режим выработки имитовставки [6]

"Слепок"вырабатывается для более чем 1 блоков открытого текста по 64 бит. Алгоритм следующий:

1) Блок данных загружается в N_1 и N_2. После чего шифруется режиме простой замены только первые 16 раундов.

2) К полученному результату подмешивается (побитовая сумма по модулю 2) следующий блок открытых данных. Для последнего блока ничего не подмешивается.

От полученного блока берется последовательность с $32 - L$ бита по 32 (нумерация с 1). Стандарт рекомендует выбирать L так, что вероятность навязывания ложных данных равна 2^{-L}.

Выработка имитовставки может производится параллельно вычислению шифротекста. Но отправляется она в самом конце.

Возможности реализации и скорость работы

Магму можно реализовать как аппаратно, так и программно. В программной реализации ГОСТ как любой шифр сети Фейстеля медленнее современных шифров типа AES (SP-сети). Однако реализация ГОСТа явно быстрее чем его конкурент 3DES.

Если сравнивать аппаратные реализации, то для ГОСТа требутеся существенно меньше элементов для его построения, чем для того же AES[9]. В работе [10] смогли за 651 GE(gate Equivalent) построить Магму против 3000 GE для AES!Поэтому он является лидером в нише недорогих и безопасных решений.

Однако при должном количестве ресурсов реализациия SP-сети быстрее чем сеть Фейстеля, так как SP-сеть изначально подается лучшей параллелизации.

В работе [11] скорость программной реализации Магмы в среднем была 135 МБ/с.

5.1. Атаки

Более 20 лет с момента создания шифра (до 2011 года) шифр Магма считался очень надежным, несмотря на многочисленные попытки его взломать.

5.1.1. Слабые ключи

В работе [16] было показано что существуют слабые ключи для шифра Магма.

Угроза и модель нарушителя

Данная атака направлена на поиск слабых ключей из пар открытого текста и шифротекста, поэтому это атака на основе знания открытого текста (known-plaintext attack, KPA).

Описание атаки

Общий вид секретного ключа выглядит так: $K = K_1||K_2||K_3||K_4||K_5||K_6||K_7||K_8$. Слабые ключи имеет структуру:

1) K - палиндром. Порождает 8-подобный шифр.

2) $K = (A||B||C||D||D||B||C||A)$. Порождает 4-подобный шифр.

3) $K = (A||||A||||A||||A||)$. Порождает 1-подобный шифр.

Такие ключи уменьшают пространство перебора для таких ключей, также позволяют провести сдвиговую атаку с дополнением (complementation slide).

Оценка требуемых ресурсов для атаки

При атаке полным перебором размерность ключевого пространства уменьшается до 2^{128}, 2^{32} при этом шанс на такой ключ пренебрежима мала.

Возможности практической реализации атаки

Эта атака непрактична так, как очень низкая вероятность успеха. Однако она предворяет другие атаки.

Пути нейтрализации атаки. В данной атаке используется простота расписания ключей. Усложнив алгоритм вычисления ключей раунда, можно помешать данной атаке.

5.1.2. Сдвиговая атака

В 2000 году была предложена сдвиговая атака [15] на последние 20 раундов шифра Магма, в котором ключ в функции F складывается по модулю два с 32-битным блоком.

Угроза и модель нарушителя

Данная атака направлена на восстановление последних 128 битов секретного ключа из пар открытого текста и шифротекста, поэтому это атака на основе знания открытого текста (known-plaintext attack, KPA).

Описание атаки.

Применяется сдвиг с поворотом (Sliding with a twist) [14] для следующей цепочки блоков:

$$\begin{array}{cccccccccccccccccccc} K_4 & K_5 & K_6 & K_7 & K_0 & K_1 & K_2 & K_3 & K_4 & K_5 & K_6 & K_7 & K_7 & K_6 & K_5 & K_4 & K_3 & K_2 & K_1 & K_0\bullet \\ K_0 & K_1 & K_2 & K_3 & K_4 & K_5 & K_6 & K_7 & K_7 & K_6 & K_5 & K_4 & K_3 & K_2 & K_1 & K_0 & K_7 & K_6 & K_5 & K_4 \end{array} \quad (5.2)$$

Обозначим за F 4 раунда шифра с ключами $K_4, \ldots, K_7$. В $2^{64/2} = 2^{33}$ парах открытого текста и шифротекста ожидаемо найдутся 2 паrs сдвига для функции F (парадокс дня рождении). Если пара найдена, то подбор ключа $K_4, \ldots, K_7$ займет около 2^9 запусков 4-раундовой версии шифра (или 2^5 20 раундовых версии). Изучив все пары, которых $\frac{2^{33} * (2^{33}-1)}{2} \approx 2^{65}$, мы найдем для каждой своего кандидата на 128-битный ключ и сохраним в таблицу. Правильный ключ должен появится дважды.

Далее нетрудно найти $K_0, \ldots, K_3$. Мы можем отделить первые 4 раунда и в том же пуле текстов искать неподвижные точки, так как последние 16 раундов образуют палиндром (первые 8 ключей в прямом порядке, последние 8 ключей в инвертированном). Всего 2^{32} значений являвяются неподвижными точками. Из них выбирается 2 кандидата, и затем по первым 4 раундам с $K_0, \ldots, K_3$ этот же ключ и восстанавливается за те же 2^5 операции. Полученный 256-битный ключ проверяется на какой-нибудь паре.

Оценка требуемых ресурсов для атаки

Это атака требует 2^{33} пар открытого текста и шифротекста, 2^{65} блоков памяти, а также 2^{70} операции шифрования.

Возможности практической реализации атаки

Большое количество затрат по памяти делают эту атаку непрактичной.

Пути нейтрализации атаки. В данной атаке используется простота расписания ключей. Усложнив алгоритм вычисления ключей раунда, можно помешать данной атаке.

5.1.3. Атака Reflection-Meet-in-the-Middle Attack

В феврале 2011 года на конференции FSE была представлена атака нового типа Reflection-Meet-in-the-Middle Attack [12] на шифр ГОСТ. Это атака требует 2^{32} пар открытого текста и шифротекста, 2^{64} блоков памяти, а также 2^{225} операции шифрования. Заметим, что уже намного лучше атаки полного перебора, требующей 2^{256} операции шифрования.

Угроза и модель нарушителя

Данная атака направлена на восстановление секретного ключа из пар открытого текста и шифротекста, поэтому это атака на основе знания открытого текста (known-plaintext attack, KPA).

Описание атаки

Round	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Key	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7
Round	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Key	K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_7	K_6	K_5	K_4	K_3	K_2	K_1	K_0

Reverse Order!

Рис. 12. Вычисление ключей раунда шифра Магма [10]

Это атака отталкивается от расписания ключей, которое можно представить схемой:
Это значит, что схему шифрования для секретного ключа K можно записать так:

$$\begin{aligned} E_K &= S \circ F_K[25,32] \circ F_K[17,24] \circ F_K[9,16] \circ F_K[1,8] \\ &= F_K^{-1}[1,8] \circ S \circ F_K[1,8] \circ F_K[1,8] \circ F_K[1,8]. \end{aligned} \tag{5.3}$$

Где $F[i,j]_k$ - это функция F схемы Фейстеля, а S - перестановка двух 32-битных слов 64-битного блока. В атаке отражения утверждается, что существует 2^{32} входных блоков x, таких что $F_K^{-1}[1,8] \circ S \circ F_K[1,8](x) = x$. То есть вероятность, что случайный блок будет таким равна: $P_{ref} = \frac{2^{32}}{2^{64}} = 2^{-32}$. Таким образом, для этих неподвижных точек существенными является только первые 16 раундов. Такая точка точно есть в $2^{32} + 1$ случайных блоках. Поэтому считаем, что у нас найдена такая точка, осталось разобраться с первыми 16 раундами.

Из-за того, что в первых 16 раунда ключами раунда по сути являются 2 подряд идущих секретных ключа, поделенные на блоки, то можно записать схему шифрования так:

$$E_K = F_K[1,8] \circ F_K[1,8] = F_K[5,8] \circ F_K[1,4] \circ F_K[5,8] \circ F_K[1,4] \tag{5.4}$$

Определение 5.1. Множество Z эквивалентных ключей на $F_k[i,j]$ для пары $x, y \in 0,1^{64}$ это:

$$Z(F_k[i,j],x,y) = ek \in 0,1^{256} : F_{ek}[i,j](x) = y \tag{5.5}$$

У шифра Магма есть следующее свойство:

Замечание 5.1. Для заданных x, y легко найти $Z(F_k[1,4],x,y)$ и $Z(F_k^{-1}[5,8],x,y)$, мощность каждого множества равна 2^{64}.

Доказательство приводится в [12]. Коротко идея заключается в том, что если мы выберем x, y и первые два ключа раунда k_1, k_2, то мы легко можем однозначно получить ключи k_3, k_4, такие что $k_1||k_2||k_3||k_4 \in Z(F_k[1,4],x,y)$ (причем мы получим все ключи Z). Аналогично для $Z(F_k^{-1}[5,8],x,y)$ будут найдены все $k_5||k_6||k_7||k_8$. Мы смогли разбить секретный ключ $k = k_1||k_2||k_3||k_4||k_5||k_6||k_7||k_8$ на все левые его части K_a и все правые части K_b на два независимых подмножества $Z_{K_a}(F_k[1,4],x,y)$, $Z_{K_b}(F_k^{-1}[5,8],x,y)$.

Нам дана пара открытого текста P и шифротекста C. Пусть $S = F_K[1,4](P)$ и $T = F_K^{-1}[5,8](C)$. Для них мы можем легко найти $Z_{K_a}(F_k[1,4],P,S)$, $Z_{K_b}(F_k^{-1}[5,8],C,T)$. Теперь мы можем провести MITM атаку, игнорируя первые 4 раунда и последние 4 раунда, между $F_K^{-1}[5,8](S)$ и $F^{K[1,4](T)}$. Схема атаки выглядит следующим образом:

Пусть дана пара открытого текста и шифротекста P и C. Алгоритм атаки следующий:

1) Выбираем произвольные S и T.
2) Считаем множества $Z_{K_a}(F_k[1,4],P,S)$, $Z_{K_b}(F_k^{-1}[5,8],C,T)$.
3) Для каждого ключа K_b из $Z_{K_b}(F_k^{-1}[5,8],C,T)$ считаем $v = F_{K_b[5,8](S)}$ и запоминаем в таблицу (v, K_b).
4) Для каждого ключа K_a из $Z_{K_a}(F_k[1,4],P,S)$ считаем $u = F_{K_a^{-1}[1,4](T)}$.
5) Если $u = v$, то запоминаем ключ $K = K_a||K_b$ как кандидата на истинный ключ. Число таких ключей будет 2^{64}.
6) Повторяем шаги $2-5$ для всевозмонжных S и T (2^{128}).

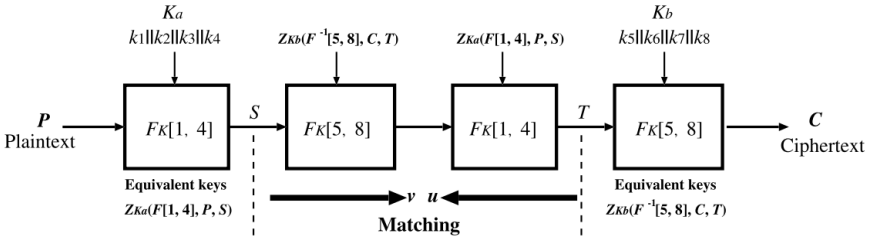

Fig. 8. R-MITM stage using equivalent keys.

Рис. 13. R-MITM атака на Магму [12]

После этого останется $2^{128} * 2^{64} = 2^{192}$ ключей-кандидатов. К этому моменту мы отсеяли ключи, которые точно не могут быть истиным ключом, поэтому в этом списке точно будет истиный ключ. Дальше он ищется в этом списке полным перебором по другим известным парам открытого текста и шифротекста.

$Z_{K_a}(F_k[1,4], P, S)$ содержит 2^{64} ключей различных ключей. Причем для разных P также разные множества ключей, так как атака предполагает что S-box'ы биективны. Количество P также равно 2^{64}, поэтому в алгоритме мы переберем всевозмножные левые части K_a секретного ключа: $2^{64} * 2^{64}$. Также для K_b, поэтому будут учетные всевозможные ключи.

Осталось не забыть, что этот алгоритм делается для каждой пары P и из известных нам пар, которых 2^{64}. Таким образом сложность атаки равна:

$$C_{\text{comp}} = \underbrace{(\left(2^{128}\left(2^{64}+2^{64}\right)\right)}_{\text{R-MITM stage}} + \underbrace{\left(2^{256-64}+2^{256-128}+\cdots\right))}_{\text{Key testing stage}} \times 2^{32} = 2^{225}. \tag{5.6}$$

Оценка требуемых ресурсов для атаки

Это атака требует 2^{32} пар открытого текста и шифротекста, 2^{64} блоков памяти, а также 2^{225} операции шифрования.

Возможности практической реализации атаки

Такое количество операции на сегодняшний день невозможно выполнить за обозримое время, также нельзя собрать такое количество пар открытого текста и шифротекста в реальной жизни. Эта атака имеет теоритическую ценность, а не практическую.

Пути нейтрализации атаки. В данной атаке используется простота расписания ключей. Усложнив алгоритм вычисления ключей раунда, можно помешать данной атаке.

5.2. Заключение

Шифр Магма довольно долго и усердно разрабывался. Полученная схема шифрования очень компактно реализуется на аппаратном уровне, при это обеспечивая высокую скорость. Более 20 лет специалисты не могли подступиться к данному шифру, но уже начали появляться атаки более быстрые чем полный перебор. Основным его недостатком является это довольно простое расписание ключей. При этом атаки требуют достаточно специфичных условий. Это может быть нецелесообразно в случаях, когда шифр используется как недорогой и быстрый вариант.

6. Шифр ГОСТ 34.12-2018 "Кузнечик"

Шифр "Кузнечик"был впервые утверждён в качестве стандарта в ГОСТ Р 34.12-2015 «Информационная технология. Криптографическая защита информации. Блочные шифры». В 2018 году он получил международную стандартизацию [17].

6.0.1. Описание шифра "Кузнечик"

"Кузнечик"- симметричный алгоритм блочного шифрования с 128-битным блоком и 256-битным ключом, основанный на SP-сети с 10 раундами.

Для описания шифра потребуется ввести следующие обозначения:

1) F - конечное поле $GF(2)[x]/p(x)$, где $p(x) = x^8 + x^7 + x^6 + x + 1$. Элементы поля представляются целыми числами: коэффициенты при степенях двойки это коэффициенты многочлена.
2) $\delta : V_8 -> F$ ставит в соотвествие двоичной строке элемент из F, δ^{-1} - отображение, обратное к δ.
3) $N(a) = S(a_{15})|| \ldots ||S(a_0)$
4) S - нелинейное преобразование заданное в виде конкретной подстановки (схема в Приложении этой работы).
5) $G(a) = \gamma(a_{15}, \ldots, a_0)||a_15|| \ldots ||a_1$.
6) $H(a) = G^{16}(a)$.
7) $Add_2[k](a)k \oplus a$.

$$\begin{aligned}\gamma\left(a_{15}, \ldots, a_0\right) = \delta^{-1}\left(148 * \delta\left(a_{15}\right) + 32 * \delta\left(a_{14}\right) + 133 * \delta\left(a_{13}\right) + 16 * \delta\left(a_{12}\right) + \right. \\ 194 * \delta\left(a_{11}\right) + 192 * \delta\left(a_{10}\right) + 1 * \delta\left(a_9\right) + 251 * \delta\left(a_8\right) + 1 * \delta\left(a_7\right) + 192 * \delta\left(a_6\right) + \\ \left. 194 * \delta\left(a_5\right) + 16 * \delta\left(a_4\right) + 133 * \delta\left(a_3\right) + 32 * \delta\left(a_2\right) + 148 * \delta\left(a_1\right) + 1 * \delta\left(a_0\right)\right),\end{aligned} \tag{6.1}$$

Шифр "Кузнечик"

$$Enc(a) = Add_2[K_10]HNAdd_2[K_9] \ldots HNAdd_2[K_2]HNAdd_2[K_1](), a128 - . \tag{6.2}$$

Расшифрование происходит в обратном порядке и с обратными фукнциями.

Вычисление ключей раунда

На вход подается 256-битный ключ $K = k_{256}|| \ldots ||k_0$. Генерируются итерационные контсанты $C_i = H(Bin_{128}(i))$, $i = 1, 2, \ldots, 32$. Bin_{128} превращает число в его 128-битное двоичное представление. Далее ключи считаются так:

1) $K_1 = K_{255}|| \ldots ||K_{128}$
2) $K_2 = K_{127}|| \ldots ||K_0$
3) $(K_{2i+1}, K_{2i+2} = F[C_{8(i-1)+8}][C_{8(i-1)+1}](K_{2i-1}, K_{2i})$, $i = 1, 2, 3, 4$.

6.0.2. Безопасность шифра

В [18] было показано, что нелинейный этап алгоритма можно декомпозировать. Также есть MITM атака [19] на "Кузнечик", требующая 2^{140} операции, 2^{153} памяти и 2^{113} данных.

7. Шифры IDEA и IDEA NXT

7.1. Шифр IDEA

В 1991 году компанией Ascom был создан шифр IDEA(International Data Encryption Algorithm).

7.1.1. Описание шифра IDEA

Шифр IDEA - это симметричный блочный шифр, который является модификацией сети Фейстеля с размером блока в 64 бит, 128-битным ключом и 8.5 раундами. Схема шифрование выглядит следующим образом:

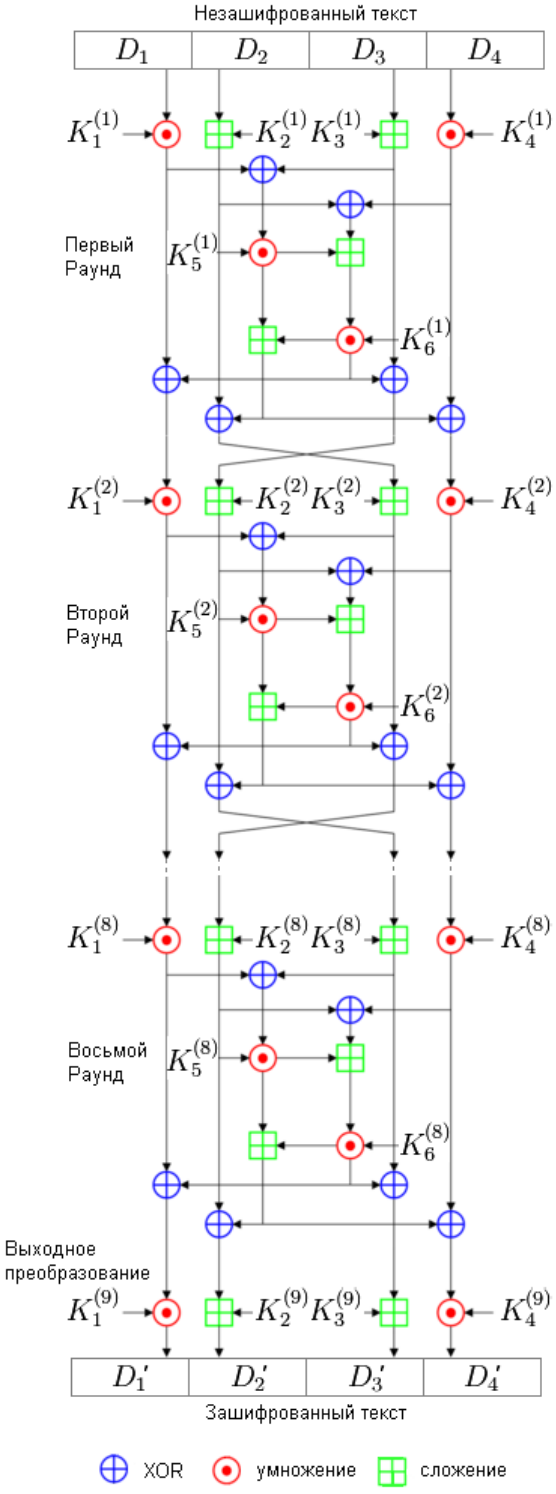

Рис. 14. Шифр IDEA [20]

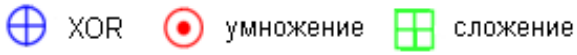

Рис. 15. Операции из схемы шифра IDEA [20]

Таким образом, исходный 64-битнный блок делится на 4 равные части D_1, D_2, D_3, D_4 и проходит через 8.5 раундов, перемешиваясь с ключами раунда при помощи операции. После 8 одинаковых раундов происходит отдельный финальный раунд, который показан на схеме. Матиматически описание для первых 8 раундов ($i = 1, \ldots, 8$) выглядит так:

$$
\begin{aligned}
A^{(i)} &= D_1^{(i-1)} * K_1^{(i)} \\
B^{(i)} &= D_2^{(i-1)} + K_2^{(i)} \\
C^{(i)} &= D_3^{(i-1)} + K_3^{(i)} \\
D^{(i)} &= D_4^{(i-1)} * K_4^{(i)} \\
E^{(i)} &= A^{(i)} \oplus C^{(i)} \\
F^{(i)} &= B^{(i)} \oplus D^{(i)} \\
D_1^{(i)} &= A^{(i)} \oplus \left(\left(F^{(i)} + E^{(i)} * K_5^{(i)}\right) * K_6^{(i)}\right) \\
D_2^{(i)} &= C^{(i)} \oplus \left(\left(F^{(i)} + E^{(i)} * K_5^{(i)}\right) * K_6^{(i)}\right) \\
D_3^{(i)} &= B^{(i)} \oplus \left(E^{(i)} * K_5^{(i)} + \left(F^{(i)} + E^{(i)} * K_5^{(i)}\right) * K_6^{(i)}\right) \\
D_4^{(i)} &= D^{(i)} \oplus \left(E^{(i)} * K_5^{(i)} + \left(F^{(i)} + E^{(i)} * K_5^{(i)}\right) * K_6^{(i)}\right)
\end{aligned}
\tag{7.1}
$$

Для последнего раунда:

$$
\begin{aligned}
D_1^{(9)} &= D_1^{(8)} * K_1^{(9)} \\
D_2^{(9)} &= D_3^{(8)} + K_2^{(9)} \\
D_3^{(9)} &= D_2^{(8)} + K_3^{(9)} \\
D_4^{(9)} &= D_4^{(8)} * K_4^{(9)}
\end{aligned}
\tag{7.2}
$$

Вычисление ключей раунда

На вход поступает 128-битный ключ, из которого для каждого из 8 раундов генерируется по 6 16-битных ключей, для финального раунда 4.

Сначала 128-битный ключ разбивается на 8 16-битных блоков: $K^(1)_1, K^(1)_2, K^(1)_3, K^(1)_4, K^(1)_5, K^(1)_6, K^(2)_1, K^(2)_2$. Затем этот ключ циклически сдвигается влево на 25 битов. После чего получаем следующие 8 16-битных ключей. Затем снова сдвигаем ключи, делаем это до тех пор пока не получим все 52 ключей раунда.

7.1.2. Безопасность шифра

У алгоритма существует большие классы слабых ключей. Есть примеры взлома IDEA с количеством раундов равным 6 [21].

7.2. Шифр IDEA NXT

В 2003 году институт EFPL представил преемника шифра IDEA - IDEA NXT, который также использует схему Лая-Месси (Lai–Massey scheme).

7.2.1. Описание шифра IDEA NXT

Шифр IDEA NXT [22] - это симметричный блочный шифр, который является модификацией сети Фейстеля с размером блока в 64/128 бит, [0-128]-битным ключом и 16 раундами. Схема шифрование выглядит следующим образом. Опишем 64-битную версию, ее раунд выгядит так:

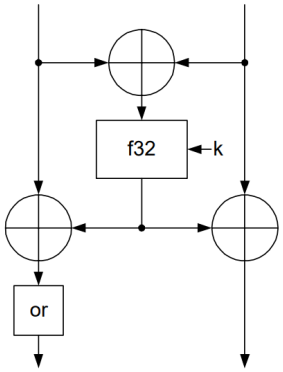

Рис. 16. Схема раунда IDEA NXT64 [22]

Функция f32 принимает на вход 32-битный блок. Схема функции f32 выглядит следуюзим образом:

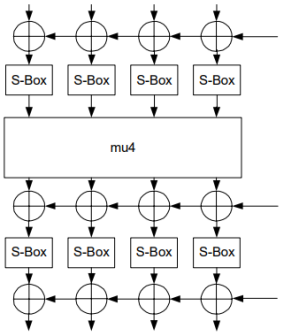

Рис. 17. Функция f32 [22]

Следовательно f32 состоит из блока подстановки, блока подстановки и блока добавление ключа раунда.

Вычисление ключей раунда Алгоритм на вход получает [0-256]-битный ключ с кратностью 8. В зависимости от длины ключа применяется конкретный алгоритм. Однако у всех общая схема:

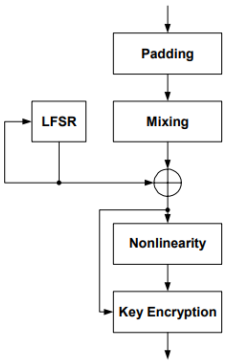

Рис. 18. Схема вычисления ключей раунда [22]

Сначала ключ расширяется до длины своей категории. Затем перемешивается с помощью рекурсии Фиббоначи. Затем идет диверсификационная часть, которая реализована с помощью LSFR и функции нелинейности. Конкретная реализация в документе [22] не приведена.

7.2.2. Безопасность шифра

В работе [23] описана атака на 64-битную версию. Он смогли атаковать 5/6/7 раундов с количеством требуемых операции 2^{71}, 2^{135}, 2^{199}.

8. Шифр RC6

Шифр RC6 был создан и подан в 1998 году в NIST как один из кандидатов на роль Advanced Encryption Standard (AES).

8.0.1. Описание шифра RC6

Шифр RC6 - это симметричный блочный шифр, который является 2-ым типом сети Фейстеля. Будем рассматривать версию алгоритма, которая работает с 128-битным блоком данных и 128/192/256-битным ключом.

Вычисление ключей раунда

На вход алгоритму подается 128/192/256битный ключ, который загружен в массив L длиной c(4/6/8) по 32-бита и количество раундов r. Выходом является массив S длины $2r + 3$ с ключами раунда. При вычислении ключей раунда используются константы: $Q_{32} = 9E3779B9$ и $P_{32} = B7E15163$. Алгоритм вычисления ключей раунда выглядит следующим образом:

Key schedule for RC6-$w/r/b$

Input: User-supplied b byte key preloaded into the c-word array $L[0, \ldots, c-1]$
Number r of rounds

Output: w-bit round keys $S[0, \ldots, 2r+3]$

Procedure:

$S[0] = P_w$

for $i = 1$ **to** $2r + 3$ **do**
$S[i] = S[i-1] + Q_w$

$A = B = i = j = 0$

$v = 3 \times \max\{c, 2r + 4\}$
for $s = 1$ **to** v **do**
{
$A = S[i] = (S[i] + A + B) \lll 3$
$B = L[j] = (L[j] + A + B) \lll (A + B)$
$i = (i + 1) \bmod (2r + 4)$
$j = (j + 1) \bmod c$
}

Рис. 19. Расписание ключей шифра RC6 [5]

Шифр RC6

Этот алгоритм шифрования работает с 4 32-битными регистрами (A, B, C, D), в которые загружается исходный 128-битный блок (в A загружается наименее значимый байт, в D - наиболее значимый байт). После работы алгоритма там будет находится шифротекст. Входом алгоритма являются количество раундов r и вычисленные ключи раунда, находящиеся в S. Алгоритм выглядит следующим образом:

Переменная $w = 32$.

Encryption with RC6-$w/r/b$

Input: Plaintext stored in four w-bit input registers A, B, C, D
Number r of rounds
w-bit round keys $S[0, \ldots, 2r+3]$

Output: Ciphertext stored in A, B, C, D

Procedure:
$B = B + S[0]$
$D = D + S[1]$
for $i = 1$ **to** r **do**
{
$t = (B \times (2B+1)) \lll \lg w$
$u = (D \times (2D+1)) \lll \lg w$
$A = ((A \oplus t) \lll u) + S[2i]$
$C = ((C \oplus u) \lll t) + S[2i+1]$
$(A, B, C, D) = (B, C, D, A)$
}
$A = A + S[2r+2]$
$C = C + S[2r+3]$

Рис. 20. Шифр RC6 [5]

8.0.2. Безопасность шифра

На сегодняшний день не было опубликовано реальных атак на шифр RC6. Лучшей атакой на него является полный перебор секретного ключа.

9. Список литературы

[1] URL: https://academickids.com/encyclopedia/index.php/Image:DES-main-network.png (дата обращения: 13.01.2024)

[2] URL: https://academickids.com/encyclopedia/index.php/Image:DES-f-function.png (дата обращения: 13.01.2024)

[3] URL: https://academickids.com/encyclopedia/index.php/Image:DES-key-schedule.png (дата обращения: 13.01.2024)

[4] Daemen J., Rijmen V. AES proposal: Rijndael. – 1999.

[5] Rivest R. L. et al. The RC6TM block cipher //First advanced encryption standard (AES) conference. – 1998. – С. 16.

[6] URL: https://ru.wikipedia.org/wiki/ГОСТ_28147-89Достоинства_стандарта (дата обращения: 16.01.2024)

[7] URL: https://ru.wikipedia.org/wiki/ГОСТ_28147-89/media/Файл:Feistel_function_GOST.png (дата обращения: 16.01.2024)

[8] Государственный стандарт СССР ГОСТ 28147-89. URL: https://files.stroyinf.ru/Data2/1/4294826/4294826631.pdf (дата обращения: 16.01.2024)

[9] URL: https://www.securitylab.ru/news/405678.php(дата обращения: 17.01.2024)

[10] URL: https://iacr.org/workshops/ches/ches2010/presentations/CHES2010_Session05_Talk02.pdf (дата обращения: 17.01.2024)

[11] URL: https://moluch.ru/archive/358/80031/ (дата обращения: 17.01.2024)

[12] Isobe T. A single-key attack on the full GOST block cipher //Journal of cryptology. – 2013. – Т. 26. – С. 172-189.

[13] Standard D. E. et al. Data encryption standard //Federal Information Processing Standards Publication. – 1999. – Т. 112.

[14] Biryukov A., Wagner D. Slide attacks //International Workshop on Fast Software Encryption. – Berlin, Heidelberg : Springer Berlin Heidelberg, 1999. – С. 245-259.

[15] Biryukov A., Wagner D. Advanced slide attacks //International conference on the theory and applications of cryptographic techniques. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2000. – С. 589-606.

[16] Ростовцев А. Г. и др. О стойкости ГОСТ 28147-89 //Проблемы информационной безопасности. Компьютерные системы. – 2003. – №. 1. – С. 75-83.

[17] URL: https://files.stroyinf.ru/Data2/1/4293732/4293732907.pdf(дата обращения: 17.01.2024)

[18] Biryukov A., Perrin L., Udovenko A. Reverse-Engineering the S-Box of Streebog, Kuznyechik and STRIBOBr1 (Full Version) //Cryptology ePrint Archive. – 2016.

[19] AlTawy R., Youssef A. M. A meet in the middle attack on reduced round Kuznyechik //IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences. – 2015. – Т. 98. – №. 10. – С. 2194-2198.

[20] URL: https://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm(дата обращения: 17.01.2024)

[21] E. Biham, O. Dunkelman, N. Keller. A New Attack on 6-Round IDEA // Лекционные записи по теории вычислительных систем = Lecture Notes In Computer Science. — Berlin / Heidelberg: Springer-Verlag, 18 августа 2007 г. — P. 211—224.
[22] URL: https://web.archive.org/web/20070928014200/http://www.mediacrypt.com/_pdf/NXT_Technical_Description_-0406.pdf (дата обращения: 17.01.2024)

10. Приложение

[23] Wu Z. et al. Impossible differential cryptanalysis of FOX //Cryptology ePrint Archive. – 2009.

Блок E шифра DES

Expansion permutation (E)

E

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

Блок P шифра DES

Permutation (P)

P

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

Блок PC1 шифра DES

Permuted choice 1 (PC-1)

PC-1

Left						
57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
Right						
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Блок PC2 шифра DES

Permuted choice 2 (PC-2)

PC-2

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

Нелинейное преобразование шифра "Кузнечик"

Int_8 - это представление 8-битовой последовательности как числа, Vec_8 - делает обратную операцию к Int_8.

4.1.1 Нелинейное биективное преобразование

В качестве нелинейного биективного преобразования выступает подстановка $\pi = Vec_8 \pi' Int_8 : V_8 \to V_8$, где $\pi' : \mathbb{Z}_{2^8} \to \mathbb{Z}_{2^8}$. Значения подстановки π' записаны ниже в виде массива $\pi' = (\pi'(0), \pi'(1), \ldots, \pi'(255))$:

π' = (252, 238, 221, 17, 207, 110, 49, 22, 251, 196, 250, 218, 35, 197, 4, 77, 233, 119, 240, 219, 147, 46, 153, 186, 23, 54, 241, 187, 20, 205, 95, 193, 249, 24, 101, 90, 226, 92, 239, 33, 129, 28, 60, 66, 139, 1, 142, 79, 5, 132, 2, 174, 227, 106, 143, 160, 6, 11, 237, 152, 127, 212, 211, 31, 235, 52, 44, 81, 234, 200, 72, 171, 242, 42, 104, 162, 253, 58, 206, 204, 181, 112, 14, 86, 8, 12, 118, 18, 191, 114, 19, 71, 156, 183, 93, 135, 21, 161, 150, 41, 16, 123, 154, 199, 243, 145, 120, 111, 157, 158, 178, 177, 50, 117, 25, 61, 255, 53, 138, 126, 109, 84, 198, 128, 195, 189, 13, 87, 223, 245, 36, 169, 62, 168, 67, 201, 215, 121, 214, 246, 124, 34, 185, 3, 224, 15, 236, 222, 122, 148, 176, 188, 220, 232, 40, 80, 78, 51, 10, 74, 167, 151, 96, 115, 30, 0, 98, 68, 26, 184, 56, 130, 100, 159, 38, 65, 173, 69, 70, 146, 39, 94, 85, 47, 140, 163, 165, 125, 105, 213, 149, 59, 7, 88, 179, 64, 134, 172, 29, 247, 48, 55, 107, 228, 136, 217, 231, 137, 225, 27, 131, 73, 76, 63, 248, 254, 141, 83, 170, 144, 202, 216, 133, 97, 32, 113, 103, 164, 45, 43, 9, 91, 203, 155, 37, 208, 190, 229, 108, 82, 89, 166, 116, 210, 230, 244, 180, 192, 209, 102, 175, 194, 57, 75, 99, 182).

Рис. 21. Нелинейное преобразование шифра "Кузнечик"