3.4. Алгоритм DES

Самым распространенным и наиболее известным алгоритмом симметричного шифрования является *DES* (Data Encryption Standard). Алгоритм был разработан в 1977 году, в 1980 году был принят NIST (National Institute of Standards and Technology США) в качестве стандарта (FIPS PUB 46).

DES является классической *сетью Фейстеля* с двумя ветвями. Данные шифруются 64-битными блоками, используя 56-битный ключ. Алгоритм преобразует за несколько *раундов* 64-битный вход в 64-битный выход. Длина ключа равна 56 битам. Процесс шифрования состоит из четырех этапов. На первом из них выполняется начальная перестановка (*IP*) 64-битного исходного текста (забеливание), во время которой биты переупорядочиваются в соответствии со стандартной таблицей. Следующий этап состоит из 16 *раундов* одной и той же функции, которая использует операции сдвига и подстановки. На третьем этапе левая и правая половины выхода последней (16-й) итерации меняются местами. Наконец,на четвертом этапе выполняется перестановка IP^{-1} результата, полученного на третьем этапе. Перестановка IP^{-1} инверсна начальной перестановке.

Рисунок 3.3 – Общая схема DES

Шифрование

Начальная перестановка

Начальная перестановка и ее инверсия определяются стандартной таблицей. Если M- это произвольные 64 бита, то X = IP (M)-переставленные 64 бита. Если применить обратную функцию перестановки $Y = IP^{-1}(X) = IP^{-1}(IP(M))$, то получится первоначальная последовательность бит.

Последовательность преобразований отдельного раунда

Теперь рассмотрим последовательность преобразований, используемую в каждом *раунде*.

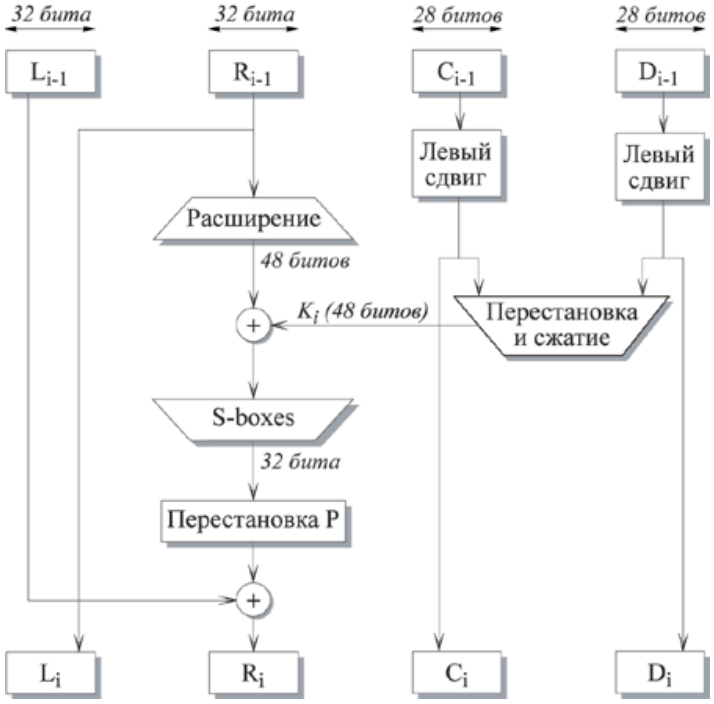

Рисунок 3.4 –I-ый раунд DES

64-битный входной блок проходит через 16 *раундов*, при этом на каждой итерации получается промежуточное 64-битное значение. Левая и правая части каждого промежуточного значения трактуются как отдельные 32-битные значения, обозначенные *L* и *R*. Каждую итерацию можно описать следующим образом:

$L_i = R_{i-1}$

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

Где $\oplus$ обозначает операцию XOR.

Таким образом, выход левой половины L_i равен входу правой половины R_{i-1}. Выход правой половины R_i является результатом применения операции XOR к L_{i-1} и функции F, зависящей от R_{i-1} и K_i.

Рассмотрим функцию F более подробно.

R_i, которое подается на вход функции F, имеет длину 32 бита. Вначале R_iрасширяется до 48 бит, используя таблицу, которая определяет перестановку плюс расширение на 16 бит. Расширение происходит следующим образом. 32 бита разбиваются на группы по 4 бита и затем расширяются до 6 бит, присоединяя крайние биты из двух соседних групп. Например, если часть входного сообщения

. . . efgh ijkl mnop . . .

то в результате расширения получается сообщение

. . . defghi hijklm lmnopq . . .

После этого для полученного 48-битного значения выполняется операция XOR с 48-битным *подключом* K_i. Затем полученное 48-битное значение подается на вход функции подстановки, результатом которой является 32-битное значение.

Подстановка состоит из восьми *S-boxes*,каждый из которых на входе получает 6 бит, а на выходе создает 4 бита. Эти преобразования определяются специальными таблицами. Первый и последний биты входного значения *S-box* определяют номер строки в таблице, средние 4 бита определяют номер столбца. Пересечение строки и столбца определяет 4-битный выход. Например, если входом является 011011, то номер строки равен 01 (строка 1) и номер столбца равен 1101 (столбец 13). Значение в строке 1 и столбце 13 равно 5, т.е. выходом является 0101.

Далее полученное 32-битное значение обрабатывается с помощью перестановки *P*, целью которой является максимальное переупорядочивание бит, чтобы в следующем *раунде* шифрования с большой вероятностью каждый бит обрабатывался другим *S-box*.

Создание подключей

Ключ для отдельного *раунда* K_i состоит из 48 бит. Ключи K_i получаются по следующему алгоритму. Для 56-битного ключа,используемого на входе алгоритма, вначале выполняется перестановка в соответствии с таблицей Permuted Choice 1 (PC-1). Полученный 56-битный ключ разделяется на две 28-битные части, обозначаемые как C_0 и D_0 соответственно. На каждом *раунде* C_i и D_i независимо циклически сдвигаются влево на 1 или 2 бита, в зависимости от номера *раунда*. Полученные значения являются входом следующего *раунда*. Они также представляют собой вход в Permuted Choice 2 (PC-2), который создает 48-битное выходное значение, являющееся входом функции $F(R_{i-1}, K_i)$.

Дешифрование

Процесс дешифрования аналогичен процессу шифрования. На входе алгоритма используется зашифрованный текст, но ключи K_i используются в обратной последовательности. K_{16} используется на первом *раунде*, K_1 используется на последнем *раунде*.

Проблемой DES является малая длина ключа. Также без ответа пока остается вопрос, возможен ли криптоанализ с использованием существующих характеристик алгоритма *DES*. Основой алгоритма являются восемь таблиц подстановки, или *S-boxes*, которые применяются в каждой итерации. Существует опасность, что эти *S-boxes* конструировались таким образом, что криптоанализ возможен для взломщика, который знает слабые места *S-boxes*. В течение многих лет обсуждалось как стандартное, так и неожиданное поведение *S-boxes*, но все-таки никому не удалось обнаружить их фатально слабые места.

Двойной DES

Простейший способ увеличить длину ключа состоит в повторном применении *DES* с двумя разными ключами. Используя незашифрованное

сообщение P и два ключа K_1 и K_2, зашифрованное сообщение C можно получить следующим образом:

$C = E_{k2} [E_{k1} [P]]$

При дешифрования два ключа применяются в обратном порядке:

$P = D_{k1} [D_{k2} [C]]$

В этом случае длина ключа равна 56 * 2 = 112 бит.

Атака "встреча посередине"

Для приведенного выше алгоритма двойного *DES* существует так называемая атака "встреча посередине". Она основана на следующем свойстве алгоритма. Мы имеем

$C = E_{k2} [E_{k1} [P]]$

Тогда

$X = E_{k1} [P] = D_{k2} [C]$.

Атака состоит в следующем. Требуется, чтобы атакующий знал хотя бы одну пару незашифрованный текст и соответствующий ему зашифрованный текст: (P, C). В этом случае, во-первых, шифруется P для всех возможных 2^{56} значений K_1. Этот результат запоминается в таблице, и затем таблица упорядочивается по значению X. Следующий шаг состоит в дешифровании C, с применением всех возможных 2^{56} значений K_2. Для каждого выполненного дешифрования ищется равное ему значение в первой таблице. Если соответствующее значение найдено, то считается, что эти ключи могут быть правильными, и они проверяются для следующей известной пары незашифрованный текст, зашифрованный текст.

Тройной DES с двумя ключами

Очевидное противодействие атаке "встреча посередине" состоит в использовании третьей стадии шифрования с тремя различными ключами. Это поднимает стоимость лобовой атаки до 2^{168}, которая на сегодняшний день считается выше практических возможностей. Но при этом длина ключа равна 56 * 3 = 168 бит, что иногда бывает громоздко.

В качестве альтернативы предлагается метод тройного шифрования,использующий только два ключа. В этом случае выполняется последовательность зашифрование-расшифрование-зашифрование (EDE).

$C = E_{K1} [D_{K2} [E_{K1} [P]]]$

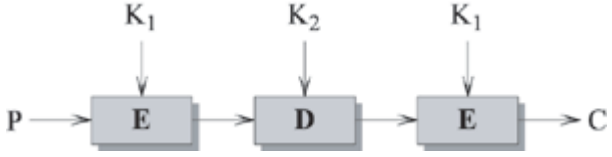

Рисунок 3.5 – Шифрование тройным DES

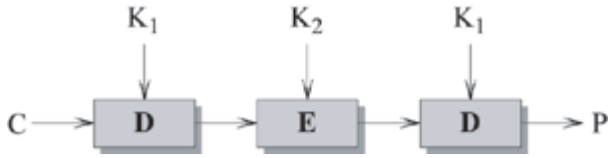

Рисунок 3.6 – Дешифрование тройным DES

Не имеет большого значения, что используется на второй стадии: шифрование или дешифрование. В случае использования дешифрования существует только то преимущество, что можно *тройной DES* свести к обычному одиночному *DES*, используя $K_1 = K_2$:

$$C = E_{K1}\,[D_{K1}\,[E_{K1}\,[P]]] = E_{K1}\,[P]$$

Известных криптографических атак на *тройной DES* не существует. Цена подбора ключа в *тройном DES* равна 2^{112}.