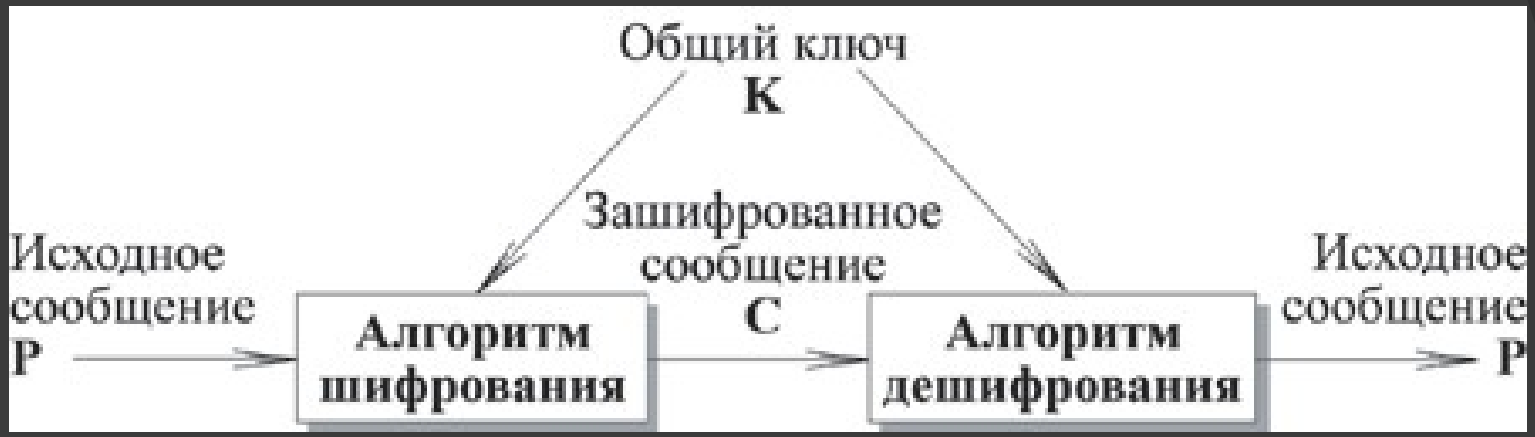
Лекция

КРИПТОГРАФИЯ

Симметричное шифрование



Алгоритм шифрования:
Вход: исходное незашифрованное сообщение (plaintext) и ключ.
Выход: зашифрованное сообщение (ciphertext).

Алгоритм расшифрования:
Вход: исходное зашифрованное сообщение (ciphertext) и ключ.
Выход: незашифрованное сообщение (plaintext).

В обоих алгоритмах используется один и тот же ключ.

Безопасность, обеспечиваемая симметричной или традиционной криптографией, зависит от нескольких факторов.

Криптографический алгоритм должен быть достаточно сильным, чтобы передаваемое зашифрованное сообщение невозможно было расшифровать без ключа, используя только различные статистические закономерности зашифрованного сообщения или какие-либо другие способы его анализа.

Безопасность передаваемого сообщения должна зависеть от секретности ключа, но не от секретности алгоритма.

Алгоритм должен быть таким, чтобы нельзя было узнать ключ, даже зная достаточно много пар (зашифрованное сообщение, незашифрованное сообщение), полученных при шифровании с использованием данного ключа.

Определения

Диффузия - это рассеяние статистических особенностей незашифрованного текста в широком диапазоне статистических особенностей зашифрованного текста. Это достигается тем, что значение каждого элемента незашифрованного текста влияет на значения многих элементов зашифрованного текста или, что то же самое, любой элемент зашифрованного текста зависит от многих элементов незашифрованного текста.

Конфузия - это уничтожение статистической взаимосвязи между зашифрованным текстом и ключом.

Если **X** - это исходное сообщение и **K** - криптографический ключ, то зашифрованный передаваемый текст можно записать в виде $Y = E_K[X]$. Обратное преобразование в этом случае будет выглядеть так: $X = D_K[Y]$.

Алгоритмы симметричного шифрования различаются способом, которым обрабатывается исходный текст. Возможно шифрование блоками или шифрование потоком.

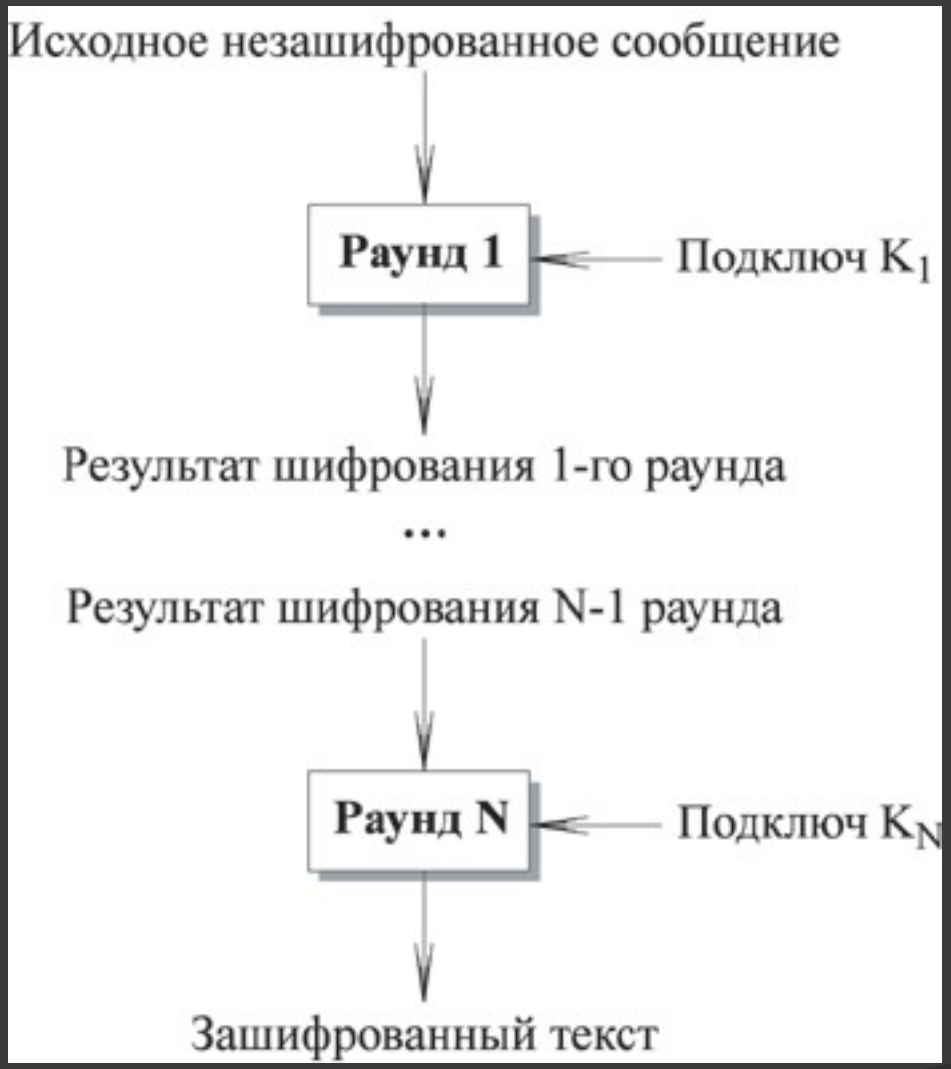
Блок текста рассматривается как неотрицательное целое число, либо как несколько независимых неотрицательных целых чисел. Длина блока всегда выбирается равной степени двойки.

В большинстве блочных алгоритмов симметричного шифрования используются следующие типы операций:

- Табличная подстановка, при которой группа бит отображается в другую группу бит. Это так называемые **S-box**.
- Перемещение, с помощью которого биты сообщения переупорядочиваются.
- Операция сложения по модулю **2**, обозначаемая **XOR**.
- Операция сложения по модулю 2^{32} или по модулю 2^{16}.
- Циклический сдвиг на некоторое число бит.

Эти операции циклически повторяются в алгоритме, образуя так называемые **раунды**.

Каждый алгоритм шифрования может быть представлен следующим образом:



Входом каждого раунда является выход предыдущего раунда и ключ, который получен по определенному алгоритму из ключа шифрования K.

Ключ раунда называется подключом.

Блочный алгоритм

Блочный алгоритм преобразовывает n-битный блок незашифрованного текста в n-битный блок зашифрованного текста.

Число блоков длины n равно 2n. Для того чтобы преобразование было обратимым, каждый из таких блоков должен преобразовываться в свой уникальный блок зашифрованного текста.

При маленькой длине блока такая подстановка плохо скрывает статистические особенности незашифрованного текста. Если блок имеет длину 64 бита, то он уже хорошо скрывает статистические особенности исходного текста.

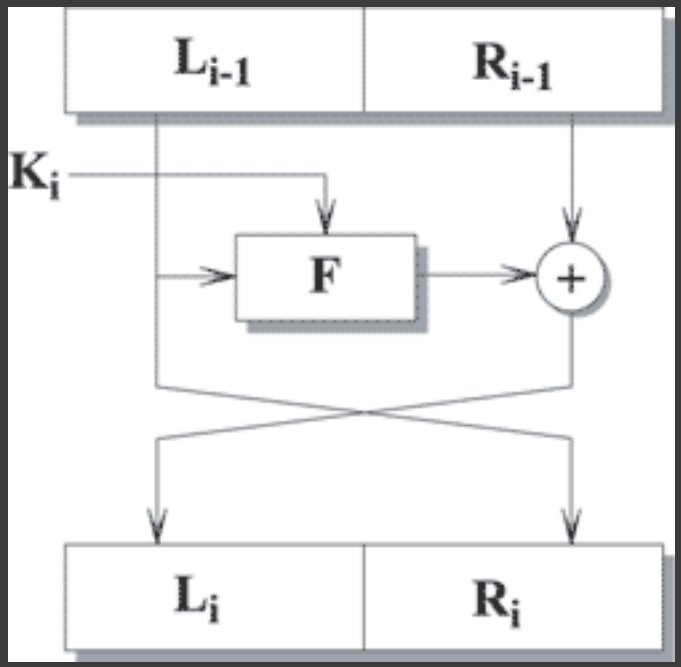
Наиболее широкое распространение получили сети Фейстеля, так как, с одной стороны, они удовлетворяют всем требованиям к алгоритмам симметричного шифрования, а с другой стороны, достаточно просты и компактны. **Сеть Фейстеля** имеет следующую структуру.

Входной блок делится на несколько равной длины подблоков, называемых ветвями.

Каждая ветвь обрабатывается независимо от другой, после чего осуществляется циклический сдвиг всех ветвей влево.

Такое преобразование выполняется несколько циклов или раундов.

Функция F называется образующей.



Каждый раунд состоит из вычисления функции F для одной ветви и побитового выполнения операции XOR результата F с другой ветвью. После этого ветви меняются местами.

Считается, что оптимальное число раундов - от 8 до 32.

Важно то, что увеличение количества раундов значительно увеличивает криптостойкость алгоритма.

Сеть Фейстеля является обратимой даже в том случае, если функция F не является таковой.

Для расшифрования не требуется вычислять F^{-1}. Для расшифрования используется тот же алгоритм, но на вход подается зашифрованный текст, и ключи используются в обратном порядке.

В настоящее время все чаще используются различные разновидности сети Фейстеля для 128-битного блока с четырьмя ветвями. Увеличение количества ветвей, а не размерности каждой ветви связано с тем, что наиболее популярными до сих пор остаются 32-разрядные процессоры.

Основной характеристикой алгоритма, построенного на основе сети Фейстеля, является функция F. Различные варианты касаются также начального и конечного преобразований. Подобные преобразования, называемые забеливанием (whitening), осуществляются для того, чтобы выполнить начальную рандомизацию входного текста.

Алгоритм DES

DES является классической сетью Фейстеля с двумя ветвями.

Данные шифруются 64-битными блоками, используя 56-битный ключ.

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.

Алгоритм 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.

Шифрование

Начальная перестановка и ее инверсия определяются стандартной таблицей.

Если M - это произвольные 64 бита, то X = IP(M) - переставленные 64 бита.

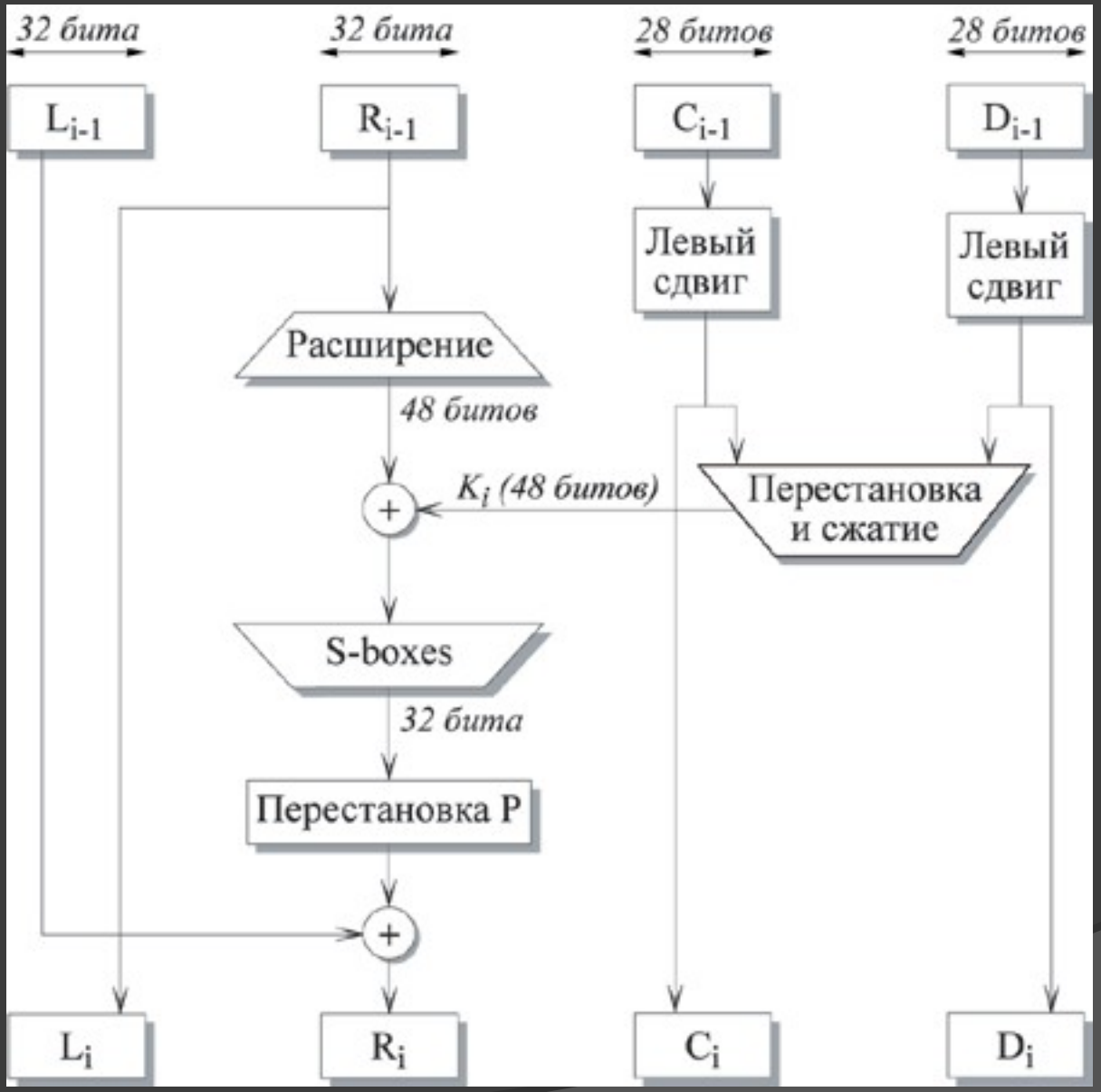
Если применить обратную функцию перестановки $Y = IP^{-1}(X) = IP^{-1}(IP(M))$, то получится первоначальная последовательность бит.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Первоначально ключ подается на вход функции перестановки.

Затем для каждого из 16 раундов подключ K_i является комбинацией левого циклического сдвига и перестановки.

Функция перестановки одна и та же для каждого раунда, но подключи K_i для каждого раунда получаются разные вследствие повторяющегося сдвига битов ключа.


32 бита
32 бита
28 битов
28 битов
L_{i-1}
R_{i-1}
C_{i-1}
D_{i-1}
Левый сдвиг
Левый сдвиг
Расширение
48 битов
K_i (48 битов)
Перестановка и сжатие
+
S-boxes
32 бита
Перестановка P
+
L_i
R_i
C_i
D_i

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 (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 табличную перестановку (PC-2), которая и создает 48-битное выходное значение, являющееся подключем раунда.

Расшифрование

Процесс расшифрования 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.

Пусть выходом i-ого раунда шифрования будет $L_i||R_i$. Тогда соответствующий вход (16-i)-ого раунда расшифрования будет $R_i||L_i$.

После последнего раунда процесса расшифрования две половины выхода меняются местами так, чтобы вход заключительной перестановки IP^{-1} был $R_{16}||L_{16}$.

Выходом этой стадии является незашифрованный текст.

Проверим корректность процесса расшифрования.

Возьмем зашифрованный текст и ключ и используем их в качестве входа в алгоритм.

Известно, что IP и IP^{-1} взаимно обратны.

$$L_{16} = R_{15}$$
$$R_{16} = L_{15} \oplus F(R_{15}, K_{16})$$

Следовательно, $L^d_0||R^d_0 = IP$ (ciphertext)

Ciphertext = $IP^{-1}(R_{16}||L_{16})$ $L^d_0||R^d_0 = IP(IP^{-1}(R_{16}||L_{16})) = R_{16}||L_{16}$

Таким образом, вход первого раунда процесса расшифрования эквивалентен выходу 16-ого раунда процесса шифрования, у которого левая и правая части записаны в обратном порядке.

$$L^d_1 = R^d_0 = L_{16} = R_{15}$$
$$R^d_1 = L^d_0 \oplus F(R^d_0, K_{16}) =$$
$$= R_{16} \oplus F(R^d_0, K_{16} =$$
$$= (L_{15} \oplus F(R_{15}, K_{16})) \oplus F(R_{15}, K_{16})$$

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$
$$D \oplus D = 0$$
$$E \oplus 0 = E$$

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

$IP^{-1}(R_0||L_0) = IP^{-1}$(IP (plaintext)) = plaintext, что и демонстрирует возможность расшифрования DES.

Проблемы DES

Так как длина ключа равна 56 битам, существует 2^{56} возможных ключей.

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:

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

Для расшифрования требуется, чтобы два ключа применялись 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 $X = E_{k1} [P] = D_{k2} [C]$.

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 K_1, и упорядочивается по значению X.

Следующий шаг состоит в расшифровании C, с применением всех K_2.

Для каждого выполненного расшифрования 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.

Атака “встреча посередине”

Если известна только одна пара значений незашифрованный текст, зашифрованный текст, то может быть получено достаточно большое число неверных значений ключей. Но если противник имеет возможность перехватить хотя бы две пары значений, то сложность взлома двойного DES фактически становится равной сложности взлома обычного DES.

Очевидное противодействие атаке "встреча посередине" состоит в использовании третьей стадии шифрования с тремя различными ключами. Это поднимает стоимость лобовой атаки до 2^{168}. 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.

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

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

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