**DES**

Рассказать сначала:

У нас есть исходный текст, который размером 64 бита, сначала делаем начальную перестановку, она делается с помощью таблицы и значение битов в этой таблицы меняется местами.

Вопросы от Берника.

**Изначально сколько бит ключ, как он генерируется и как он в дальнейшем преобразовывается?**

У нас есть начальный ключ размером 56 бит или 7 байтов. Добавляются биты к ключу таким образом, чтобы каждый байт содержал нечетное число единиц, это делается для того чтобы обнаружить ошибки. Для того чтобы их обнаружить мы можем использовать код с одной проверкой на четность. Мы проверяем так чтобы символ, который мы выбираем, его сумма по модулю 2 со всеми информационными символами равнялась нулю, если она не равна нулю получается он нечетный.

**Берник: так хорошо у нас 56 бит, как мы преобразуем дальше?**

Далее мы расширяем ключ (добавляем ему биты) и для этого расширенного ключа мы делаем перестановку, у нас есть таблица, в которой мы делим на два блока наш ключ, 28 бит каждый, один это старшие биты, а второй это младшие биты. И потом мы делаем циклические сдвиги также с помощью таблицы.

В таблице сдвигов у нас число сдвигов от 1 до 16 и эти сдвиги состоят из 48 бит, выбранных из битов, которые были 56 бит. (То есть мы какие-то биты удаляем).

**Берник: как у нас генерируется ключ имеется ввиду, как определить слабый или полу это слабый ключ?**

Для каждого раунда алгоритма определенные первоначальные ключи они могут является слабыми или полу слабыми. Так как первоначальное значение оно разбивается на две половины каждый из которых сдвигается независимо, если все биты каждой половины равны 0 или 1, то получается для всех раундов алгоритма используется один и тот же ключ. Это может получится если ключ состоит из одних 0-ей или 1-ц или если одна половина ключа состоит из одних единиц, а другая из одних нулей. Также некоторые пары ключей при зашифровании они переводят текст в такой же текст – это называется слабыми ключами.

**Берник: как работает cеть Фейстеля как данные заходят, какой блок данных?**

В сетях Фейстеля информация разбивается на блоки одинаковой длинны, полученные блоки называются входными, так как поступают на вход алгоритма, в случае если длинна входного блока меньше чем размер, который выбрал алгоритм, на который можно разбить, то блок удлиняется, каким-либо способом. Как правило для блока берется размер этой степени к двойке, то есть 64 бита или 128.

В Алгоритме DES функция Фейстеля, для функции аргументами являются 32битный вектор и 48битовый ключ, который мы сгенерировали. Для вычисления этой функции последовательно у нас используется функция расширения, потом сложения по модулю 2 с ключом, потом преобразование, состоящее из 8-ми преобразований из блоков.

Берник: каким образом у нас что расширяется или сжимается. Откройте схему.

**Открываем схему**

У нас поступает исходный текст на вход 64 бита, происходит сначала начальная перестановка, также генерируются ключи, разделяется блок на left и right по 32 бита. f – это функция Фейстеля, к – это ключТо есть сначала у нас функция расширяется.

**Берник: как? С помощью таблицы также функции расширения.**

**Берник: как таблица работает, как расширение происходит?**

Допустим у нас 32битовый вектор расширяется до 48 битного. Путём дублирования некоторых битов это происходит. Далее у нас происходит сложение по модулю 2 с ключом. Потом преобразование, состоящее из 8 преобразований из блоков. Далее конечная перестановка.

**Как 48 бит с помощью s боксов преобразовать?**

Преобразование происходит с помощью s и преобразование происходит с помощью таблицы, эта таблица от 1 до 8.

**Как мы 48битовую последовательность разбиваем на s боксы?**

Разделяем по 6 бит, потом мы берем наши первые и последние разряды и 4 по середине. И она определяют наши строки и столбы s блока, получаем число и делаем перестановку. То есть мы разделяем на 8 блоков по 6 бит результат функции.

Получается 6 битную последовательность преобразовываем в 4 битовую в каждом s боксе.

**Конкретно поясните как это делается!**

Предположим, что у нас есть какое-то число двоичное, нам нужно его преобразовать мы берем первые и последние разряды этого числа и средние 4 разряда. И это будет строки и столбцы нашего s блока, пишем двоичное представление числа.

**То есть у нас есть 6 бит, мы берем крайние биты и средние биты, и что они нам показывают?**

Крайние биты это идёт наше число А и 4 средние разряды – это число B. Эта пара чисел определит число, находящее в пересечении строки А и столбца B. И у нас будет какое-то десятичное число. И это десятичное число мы преобразовываем в 4 битную двоичную последовательность и таких блоков 8, 8\*4=32.

**Что дальше?**

После этого происходит ещё одна перестановка.

**Берник: а что это за перестановка?**

первые четыре бита результирующего вектора после действия функции f

Она перемешивает биты. Начальный и конечный p перестановки, они не влияют на криптостойкость. Криптостойкость повышается операциями внутри раунда (преобразованиями сети фейстеля). Но по алгоритму и спецификации они нужны эти начальные и конечные перестановки.

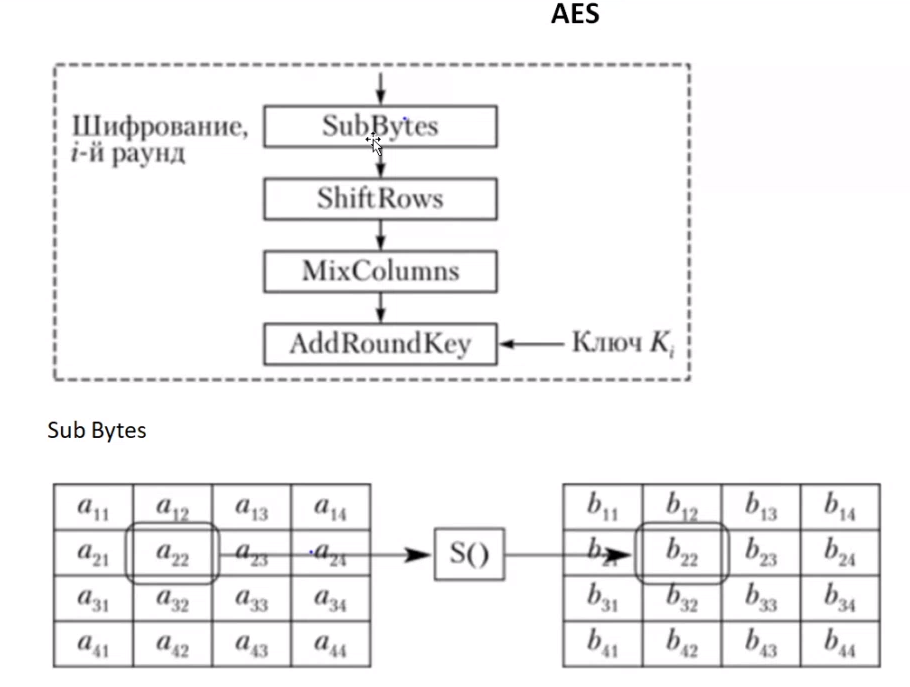
**Берник: Как связаны таблицы начальной и конечно перестановки?**

Если у нас значение первого элемента 40 в обратной матрице, а значение 40-го элемента в начальной – это единица. То есть мы просто будем смотреть по индексу.

**3des**

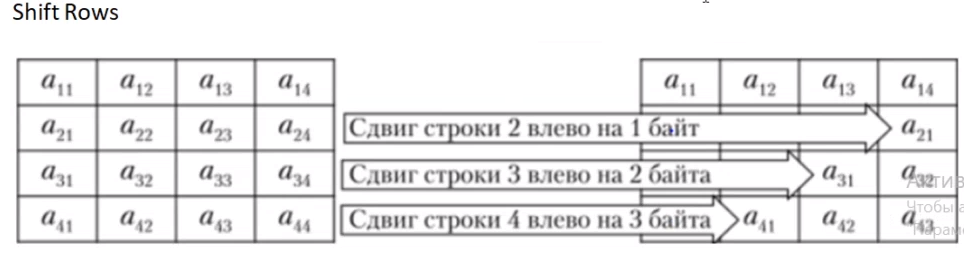
Из-за того, что у des маленькая длинна ключа 56 бит, придумали 3des чтобы сделать его более криптостойким. Мы des используем три раза, то есть результат первого des с первым ключом, потом от этого результата мы к нему добавляем ещё один ключ и так делаем три раза. В итоге улучшили des чтобы длинна ключа стала 112 бит.

Криптостойкость у 3des гораздо выше. Но ниже скорость шифрования по сравнению с des.

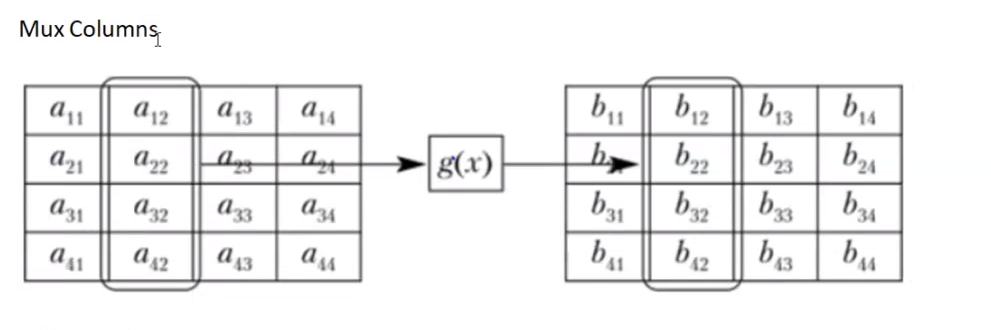


Sub bytes

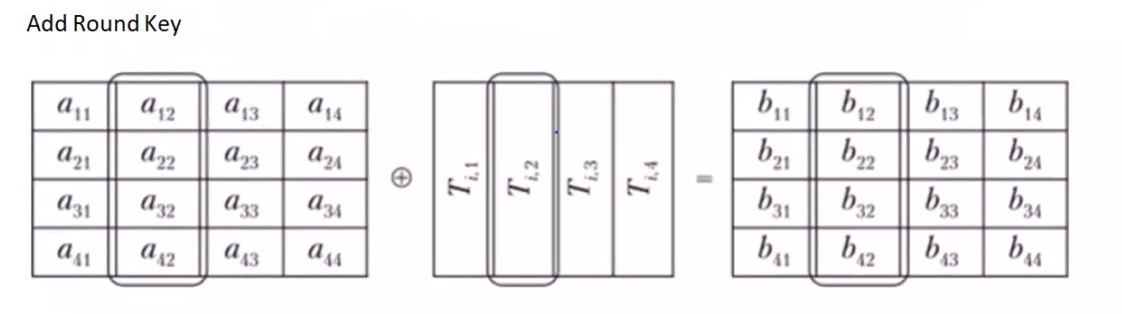
У нас входной массив состоит из квадрата, который представляет собой 16 байт, то есть это двумерный массив 4x4, входные данные 128 бит, то есть 16 байт. Длинна ключа 128 бит 192/256 бит. Когда мы шифруем на iтом раунде у нас первый блок subbytes. Мы просто меняем наш байт на какой-то другой байт. У нас первый 16 ричный разряд, то есть у нас есть 2 разряда, первый разряд отвечает за строку, другой за столбец и когда они соединяются, получается другой байт на выходе, после первого блока.



Далее shift rows, мы просто делаем циклический сдвиг влево 2строки на один байт, в 3-ей на на 2 и в четвертой на 3байта.



У нас идет блок mux columns. Мы наш i-ый столбец рассматриваем как полином, который умножаем на полином константу g(x). Прикол Aes там используются одни и те же данные. g(x) будет для всех раундов одинаковой, ключи разные только будут. То есть мы полином умножаем на g(x) и берем по какому-то модулю и получаем результирующее значение нашей таблицы.

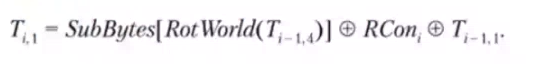


Мы ксорим наш столбец на столбец итого ключа.

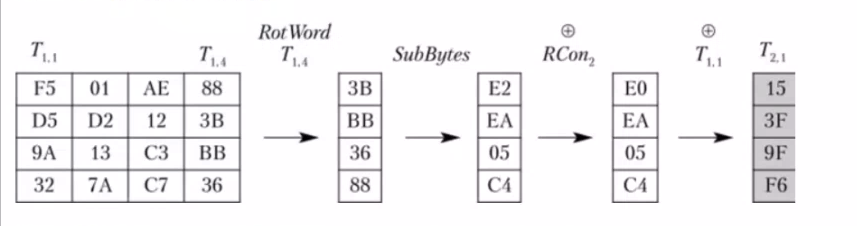
Как генерируются эти ключи, а какая может быть длинна этих ключей?

128,192,256.

В зависимости от длинен ключа определенное количество раундов используется 10,12,14 раундов

.

Для генерации ключа у нас есть формула. rotworld это мы перемешиваем значения наших байт в нашем столбце, вот пример



RCon – это константа, которую мы задаем в начале для генерации наших ключей.

**Serpent**

Так тут 128 бит, идет начальная перестановка, далее xorим с нашим ключом раунда, результат делим на 32 блока по 4 бита, далее каждый из этих 4-х битов идет в блок S0.

Таблица!

Как показано идет табличная замена, допустим к нам пришло число 0, на выходе будет 3-ка, то есть у нас есть 4 бита, это числа от 0 до 15, смотря какое число пришло, если 0 то 3, если 1 то 8, если 2 то 15. Это выходные данные. Использовалось 8 таких замен, то есть 8 вариаций, на каждый раунд использовалась одна и та же табличная замена. То есть в первом у нас s0 использовался для всех 32ух блоков. После этого данные собирались в блок и шли в линейное преобразование.

Таблица!

Здесь у нас таблица, где для каждого блока будет показано что будет на выходе.

Промежуточный ключ, в конце!

Ну и процедура расширения ключа, ключ у нас 128,192,256. Но при этом он всегда будет расширятся автоматически до длины 256. Если у нас будет размер меньше, будут добавлены единицы и то количество нулей, чтобы ключ был 256. В начале мы делим на блок на 8 wi то есть по 32 бита, подключей, благодаря этому генерируем 131 промежуточный ключ по этой формуле. Ну и далее с помощью блока s мы отправляем туда подключи и так генерируются ключи для каждого раунда.

**Twofish**

Картинка

Длинна ключа тоже 128,192,256. Входной текст 128. Пришел входной блок 128, мы его разделили на 4 части, с каждой из частей сделали входное отбеливание, это обычная операция xor с каждым подключём, всего у нас 40 подключей, 8 из которых используется для выходного и входного отбеливания. Далее наши блоки идут в блок g, в котором делятся на 4 блока по 8 бит, которые заходят в s бокс, в котором происходит наша перестановка, на выходе из которого получается вектор, который идет в блок nds, блок nds – это матрица, в которой происходит преобразование с нашими векторами пришедшими, далее оттуда выходят данные, которые мы слаживаем по модулю 32, xorим с другими нашими блоками, производим смещение на какое-то количество битов влево и далее идут другие раунды, всего 16 раундов. Потом идет выходное отбеливание и получаем зашифр текст.

Какой ты бы выбрал алгоритм. AES. Потому что алгоритм должен быть по производительности норм, не долго выполнялся, и чтобы криптостойкость была хорошей. То есть не нужно усложнять алгоритм, там и так длинна ключа 256, 2 в 256 много, и чтобы их перебрать много времени нужно потратить. Поэтому криптостойкость должна быть умеренной.