СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

А.Б. Левина

Криптография и криптографические протоколы

Симметричное шифрование

СПбГЭТУ «ЛЭТИ», 2022 г.

1 ВВЕДЕНИЕ

На прошлой лекции мы познакомились с некоторыми историческими шифрами, ввели основные обозначения и определения. Сейчас мы начинаем изучать симметричные шифры. Симметричному шифрованию мы посвятим несколько лекций и за это время узнаем:

1. Основные принципы современных симметричных алгоритмов;
2. Принцип Керкгоффса и математические понятия, используемые в симметричном шифровании;
3. Сеть Фейстеля;
4. Алгоритм DES, режимы шифрования;
5. ГОСТ;
6. Алгоритм Rijndael (AES);
7. Область поточных шифров и регистров сдвига с линейной обратной связью.

2 СИММЕТРИЧНЫЕ АЛГОРИТМЫ ШИФРОВАНИЯ

Давайте вспомним определение симметричного алгоритма, мы вводили его на первой лекции.

Определение 2.0.1 *Криптосистемы называются* ***симметричными криптосистемами или криптосистемами с секретным ключом****, если для* ***шифрования и дешифрования используется один и тот же ключ****.*

Здесь выделено ***для шифрования и дешифрования используется один ключ***, это нам надо очень хорошо запомнить, так как дальше у нас будет асимметричное шифрование, где для шифрования и дешифрования используются разные ключи.

Исторические шифры, которые мы рассматривали ранее, как раз являются симметричными алгоритмами шифрования.

Симметричные шифры делятся на:

- блочные;
- поточные.

Определение 2.0.2 *Алгоритм симметричного шифрования называется* ***блочным алгоритмом шифрования/блочным шифром***, *если в процессе шифрования данные разбиваются на блоки фиксированной длины, и данные блоки могут обрабатываться независимо друг от друга.*

Схема блочного шифрования представлена на рисунке 1:

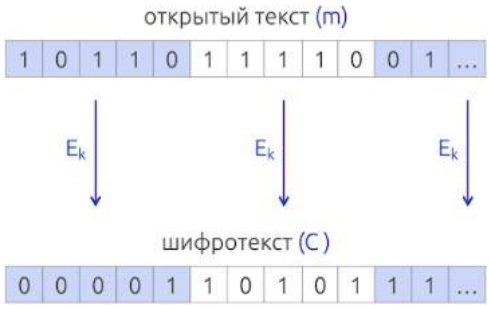

Рис. 1: Схема блочного шифрования.

Например перестановочные шифры и шифр Вижнера, который мы рассматривали на первой лекции, относятся к блочным шифрам.

Определение 2.0.3 *Алгоритм симметричного шифрования называется* ***поточным алгоритмом шифрования или потоковым шифром****, если каждый символ открытого текста преобразуется в символ шифрованного текста в зависимости не только от используемого ключа, но и от его расположения в потоке открытого текста.*

Поточное шифрование это побитовое шифрование (в редких случаях – побайтовое). В поточном шифровании, чаще всего, шифрующей функцией выступает операция XOR, а безопасность поточного шифрования обеспечивается за счет генератора ключевого потока, который создает поток ключей, равный длине исходного открытого текста. Более подробно поточное шифрование мы рассмотрим позже.

Схематично поточное шифрование можно представить следующим образом:

Рис. 2: Схема поточного шифрования.

Симметричные алгоритмы окружают нас повсюду; блочные алгоритмы находятся и в наших смартфонах, и в проходках (RFID-карты), в банковских картах, в почтовых программах, и т.д. Область применения поточного шифрования значительно уже, она ограничивается обработкой видео и аудио сигналов, один из примеров мы рассмотрим когда будем знакомиться со стандартом GSM.

Если сравнивать поточное и блочное шифрование, то:

- блочное шифрование может трансформироваться в поточное;
- поточный шифр имеет более математизированную структуру;
- поточные шифры не очень удобны с точки зрения программного обеспечения, но высоко эффективны с точки зрения аппаратной реализации;
- блочные шифры удобны и для программных и аппаратных средств, но не допускают быстрой обработки информации;
- аппаратные средства функционируют быстрее, чем программное обеспечение, но это приводит к снижению гибкости, поэтому поточное шифрование работает быстрее блочного.

Если говорить про криптостойкость, то криптостойкость поточного шифрования намного ниже, чем криптостойкость блочного шифрования, так как в шифрующей функции в блочном шифровании может быть намешано много преобразований, а безопасность поточного шифрования основывается только на генераторе ключевого потока. Дальше мы все это увидим в самих алгоритмах.

3 ТЕОРЕТИКО-ИНФОРМАЦИОННАЯ СТОЙКОСТЬ

3.1 ПРИНЦИП КЕРКГОФФСА

Давайте поговорим немного о математике, о некоторых принципах, которые лежат в основе современной криптографии. И начать я хотела бы с принципа Керкгоффса.

Огюст Керкгоффс - 19.01.1835-9.08.1903, нидерландский криптограф, лингвист, историк, математик. Автор фундаментального труда «Военная криптография» (1883 г.), в котором были сформулированы общие требования к криптосистеме, а также утвержден криптоанализ как единственно верный способ испытания шифров, рассмотрим эти

принципы.

1.1 Принцип Керкгоффса:

- система должна быть физически, если не математически, невскрываемой;
- нужно, чтобы не требовалось сохранение системы в тайне; попадание системы в руки врага не должно причинять неудобств («Принцип Керкгоффса»);
- хранение и передача ключа должны быть осуществимы без помощи бумажных записей; корреспонденты должны располагать возможностью менять ключ по своему усмотрению;
- система должна быть пригодной для сообщения через телеграф;
- система должна быть легко переносимой, работа с ней не должна требовать участия нескольких лиц одновременно;
- система должна быть проста в использовании, не требовать значительного «умственного напряжения» или соблюдения большого количества правил.

Как мы видим, все эти принципы актуальны для нас и сейчас, единственное, что изменилось за столь большой промежуток времени это то, что "система должна быть пригодной для сообщения через телеграф". Переформулировав это для нашего времени, мы можем сказать, что система должна быть пригодна и для смартфонов, и для компьютеров и т.д., то есть для любого устройства, где необходимо обеспечить криптографическую защиту.

3.2 ШИФР ВЕРНАМА

Шифр был разработан в 1917 году сотрудником AT&T Гилбертом Верна- мом, этот шифр также называется «шифр-блокнот». Он является единственным доказуемо абсолютно

стойким шифром, но при этом не используется. Почему это так мы поймем, обсудив алгоритм.

При шифровании генерируется случайный ключ k, обладающий следующими свойствами:

- является случайной величиной, имеющей равномерное распределение, то есть реализуется с одинаковой вероятностью: $P_k = 1/2^N$, где k— ключ, а N— количество бинарных символов в ключе;
- совпадает по размеру с заданным открытым текстом;
- применяется только один раз.

Шифрующая функция E=XOR.

Свое название «шифр-блокнот» алгоритм получил за схожесть с записью информации в блокноте. Мы записали открытый текст, записали ключ, произвели побитовую операцию XOR битов открытого текста и ключа и вырвали страничку с ключом, тем самым удалив ключ.

Операция XOR – операция взятия по модулю 2, или, как ее еще часто называют, «исключающее ИЛИ», вычисляется по следующему правилу:

⊕	0	1
0	0	1
1	1	0

Рис. 3: Операция XOR

Шифр Вернама является единственным доказуемо стойким шифром, так как удовлетворяет теореме Шеннона, но на практике не применяется из-за того, что каждый сеанс требуется создавать новый ключ, равный длине текста, что в реальных системах невозможно.

Пример 3.3.1 *Рассмотрим пример работы шифра Вернама. Пусть мы хотим зашифровать два сообщения* $m_1 = 110001$ *и* $m_2 = 100011$, *для этого нам необходимо сгенерировать два ключа, равных длине сообщений, пусть это будет* $k_1 = 000001$ *и* $k_2 = 101111$.

Мы получаем первый шифротекст:

$C_1 = m_1 \oplus k_1 = 110001 \oplus 000001 = 110000$.

Второй шифротекст $C_2 = 100011 \oplus 101111 = 001100$.

Рассмотрим ситуацию, когда мы решили шифровать два сообщения одним ключом $k_1 = 000001$.

Тогда получится $C_1 = m_1 \oplus k_1 = 110001 \oplus 000001 = 110000$ *и*

$C_2 = m_2 \oplus k_1 = 100011 \oplus 000001 = 100010$.

XOR-им их между собой

$C_1 \oplus C_2 = 110000 \oplus 100010 = 010010$.

Посмотрим, что получится если произвести операцию XOR между двумя открытыми текстами m_1, m_2, *мы увидим, что значения* $C_1 \oplus C_2$ *и* $m_1 \oplus m_2$ *совпали. Это как раз произошло из-за того, что мы использовали один ключ.*

Если в этой ситуации один из открытых текстов, например m_1, *был подсунут нам Евой, то она без проблем восстановит второй открытый текст* m_2, *сделав* $C_1 \oplus C_2 \oplus m_1$.

4 БЛОЧНЫЕ ШИФРЫ, ЭТАПЫ РАЗВИТИЯ

Развитие блочного шифрования можно разделить на следующие этапы:

- 1949 – Клод Шеннон опубликовал свою работу и ввел понятие переста- новки + замены;
- 1970 – разработан алгоритм Lucifer (IBM), предложена сеть Фейстеля и введено понятие SP-сеть (Substitution-Permutation network);
- 1973 – National Institute of Standards and Technology (NIST) объявлил конкурс DES (Data Encryption Standard) на создание общего алгоритма блочного шифрования;
- 1974 – 2-ой этап конкурса: выиграл алгоритм компании IBM;
- 1977 – DES признан официальным стандартом в США;
- 1989 – создание советского алгоритма блочного шифрования ГОСТ 28147-89;
- 1990 – публикация ГОСТ 28147-89;
- 1997 – взлом DES на суперкомпьютере за 3 дня;
- 1997 – объявлен конкурс AES (Advanced Encryption Standard);
- 2000 – конкурс AES выигрывает алгоритм Rijndael;
- 2002 – алгоритм Rijndael признан новым официальным стандартом в США.

Несколько известных блочных шифров - RC5, RC6, DES, 3DES, AES, ГОСТ 28147-89.

5 SP-NETWORK И СЕТЬ ФЕЙСТЕЛЯ

Как мы говорили ранее, в 1949 Клод Шеннон в своей работе «Теория связи в секретных системах», предложил идею итеративных блочных шифров на основе SP-network/SP-сеть (перестановки + замены).

Шифр преобразует блоки открытого текста m постоянной длины n в блоки шифротекста C той же длины посредством циклически повторяющихся обратимых функций, известных как раундовые функции.

Запишем это математически:

$C_i = R_i(k_i, C_{i-1})$,

R – раундовая функция,

S - количество раундов,

k_i – подключ – преобразованный ключ, используемый на i-ом раунде,

C_i - значение блока после i-го раунда,

i - номер раунда,

n – длина блока.

Схематически это представлено на рисунке 4:

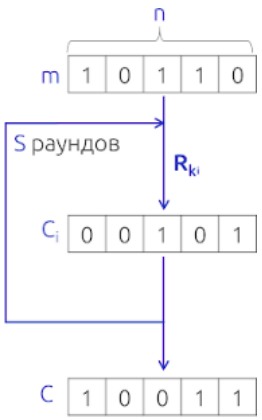

Рис. 4: Работа раундовой функции.

Еще одно важное понятие в блочном шифровании это SP -сеть - substitution-

permutation network (SPN). SP сеть представляет собой чередующиеся подстановки (Substitution) и перестановки (Permutation), которые мы уже рассматривали, говоря об исторических шифрах.

В современных алгоритмах подстановки и перестановки создаются с помощью S-блоков и P-блоков.

- S-блоки (substitution box or S-box) – таблица подстановки;
- P-блоки (permutation box or P-box) – таблица перестановки.

В работе Шеннона были сформулированы основные принципы, которые должны обеспечивать работу блочного алгоритма и которые как раз обеспечиваются с помощью P-блоков и S-блоков:

- Рассеивание обеспечивают таблицы перестановки;
- Перемешивание обеспечивают таблицы подстановки (лавинный эффект).

Рассеивание, часто его еще называют диффузией, обеспечивает избавление от избыточности открытого текста, более конкретно - удаляет взаимосвязь между символами открытого текста.

Рассмотри более подробно, что такое лавинный эффект.

Лавинный эффект - важное криптографическое свойство для шифрования, которое означает, что изменение значения малого количества битов во входном тексте или в ключе ведет к «лавинному» изменению значений выходных битов шифротекста. То есть, это зависимость всех выходных битов от каждого входного бита.

Термин «лавинный эффект» впервые введён Фейстелем в статье «Cryptography and Computer Privacy», опубликованной в мае 1973 года, хотя концептуальное понятие использовалось ещё Шенноном.

Давайте теперь познакомимся с сетью Фейстеля.

«Сеть Фейстеля» предложил в 1971 Хорст Фейстель при патентовании алгоритма

Lucifer. В те времена компьютерные мощности были очень маленькие и на шифрование уходило очень много времени, Фейстель предложил за один раунд шифровать только полблока, а вторую половину просто передвигать.

Схематически это выглядит следующем образом:

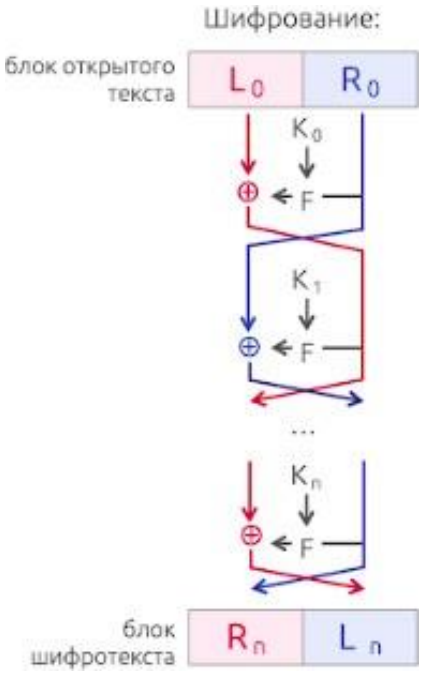

Рис. 5: Сеть Фейстеля

Шифрование происходит по следующей формуле:

$l_i = r_{i-1},\ r_i = l_{i-1}$ $F\ (k_i, r_{i-1})$, где F - шифрующая функция, l_i - левая часть блока, r_i - правая часть блока, k_i -подключ, используемый, на i-ом раунде.

Расшифрование происходит по следующей формуле:

$r_{i-1} = l_i,\ l_{i-1} = r_i \oplus F\ (k_i, l_i)$.

Отметим плюсы сети Фейстеля:

- функция раунда обратима вне зависимости от свойств функции F;
- одну и ту же микросхему можно использовать и для шифрования, и для расшифрования.

Давайте теперь познакомимся с некоторыми алгоритмами, использующими сеть Фейстеля, и начнем мы наше знакомство с алгоритма DES.

6 АЛГОРИТМ DES

6.1 АЛГОРИТМ DES

Сейчас мы познакомимся с первым алгоритмом, который стал стандартом и стал доступен для общего пользования, с алгоритмом DES (Data Encryption Standart).

Как мы помним, алгоритм DES появился в 1977, после того как в 1973 National Institute of Standards and Technology объявил Конкурс на создание блочного алгоритма, который будет стандартизован и будет общедоступен. Алгоритм использует сеть Фейстеля, таблицы перестановки и подстановки, все таблицы известны, секретной информацией является только ключ.

Рассмотрим, какие же параметры использует алгоритм DES:

- число раундов алгоритма равно 16, то есть алгоритм 16 раз проводит одни и те же действия с разными подключами и с данными, которые были получены на предыдущем раунде;
- длина блока n = 64 бита. Поступившие данные разбиваются на блоки длиной 64 бита, если данные не делятся нацело на 64, то в последний блок добавляется информация об изначальной длине данных;
- размер ключа k – 56 бит. Подключи k_1, k_2, . . . по 48 бит (разворачивание из основного

ключа через подстановки, перестановки и циклические сдвиги). Это мы подробно рассмотрим чуть позже.

Структура DES выглядит следующим образом:

- начальная перестановка IP;
- расщепление блока пополам на две части по 32 бита каждая;
- 16 раундов сети Фейстеля;
- соединение 32-ух битных половин блока;
- конечная перестановка IP^{-1} (обратная начальной).

Чуть позже мы рассмотрим все эти перестановки более подробно, а пока давайте разберемся, что происходит в самой важной части алгоритма - шифрующей функции F.

Действие функции F:

- Перестановка с расширением. С помощью таблицы мы «растягиваем» 32 бита до 48-и бит. Зачем необходимо расширение? как раз эта операция создает лавинный эффект, о котором мы говорили ранее;
- Сложение с подключом. На данном шаге 48 получившихся бит мы XOR- им с 48-ми битным подключом;
- Расщепление. На данном шаге, получившиеся ранее 48 бит делят на 8 частей по 6 бит каждая;
- Подстановки через S – блок. Данная подстановка является основной частью шифрующей функции и именно здесь происходит вся криптографическая магия. Получившаяся ранее восемь 6-ти битных кусков мы пропускаем через таблицы

подстановки - S - блоки и получаем на выходе восемь 4-х битных кусков, то есть мы обратно вернулись к 32-м битам;

- Перестановки через P – блок. Это еще одно перемешивание, создаваемое с помощью таблицы перестановки.

Схематически действием функции F на одном раунде можно представить следующей картинкой:

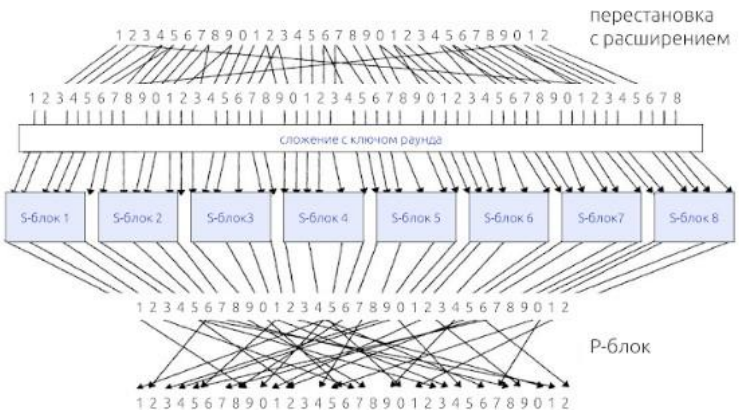

Рис. 6: Схема раунда

Давайте рассмотрим каждый из этих шагов подробно.

IP перестановка.

IP перестановка представляет собой таблицу перестановки, в которой указано, на какую позицию идет какой бит, например 58-ой бит идет на первую позицию, 50-ый на вторую и так далее, в соответствии с представ- ленной таблицей. Плюсом IP перестановки является то, что на шифрование идут уже несвязанные данные.

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

Рис. 7: IP перестановка

Нам необходимо помнить, что все таблицы перестановок и подстановок в алгоритме DES известны, не известен только ключ.

Перестановка с расширением.

Перестановка с расширением работает уже с 32-ух битными подблоками, она создает лавинный эффект за счет повторения определенных бит.

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

Рис. 8: Перестановка с расширением

После расширения мы XOR-им получившиеся данные с подключом и переходим к S - блокам.

S-блоки.

S - блоки являются основной частью алгоритма, рассмотрим их работу. Сами по себе S - блоки являются двумерными массивами размерности 4x16 (4 строки, 16 столбцов). Нумерация строк и столбцов начинается с нуля. В каждой ячейке массива записано число в диапазоне от 0 до 15, числа в строке не повторяются. Числа, записанные в ячейке можно представить с помощью четырех бит.

В алгоритме используется 8 S - блоков. Перед началом работы с S - блоками мы разбиваем 48 бит на 8 кусков по 6 бит каждый. Эти 6 бит задают нам номер столбца и строки в соответствующем S - блоке следующим образом

- первый и последний бит задают номер строки, а четыре бита посередине задают номер столбца. Это выглядит следующим образом:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	5	13

Рис. 9: Пример работы S-блока

Каждые 6 бит работают со своим S - блоком. После прохождения всех S - блоков мы получаем на выходе 8 четырехбитных куска, соответственно мы получаем 32 бита.

P перестановка.

P перестановка представляет собой еще одну таблицу перестановок, ко- торая работает с 32 битами, вышедшими из S - блока.

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

Рис. 10: P перестановка

Мы помним, что алгоритм DES работает с сетью Фейстеля, соответственно в функция F идут правые 32 бита нашего блока и после прохождения всех преобразований на одном раунде, получившиеся биты XORся с 32 бита- ми левой части, и только после этого данные идут на новый раунд.

Создание подключей.

Рассмотрим, как происходит создание подключей для каждого раунда шифрования.

Создание подключа состоит из нескольких этапов:

1 этап - 56-и битный ключ преобразуются в 64 битовую последовательность за счет добавления битов четности. 8, 16, 24, 32, 40, 48, 56, 64 - биты четности, это биты, выбранные таким образом, чтобы каждый байт ключа содержал нечетное число единиц. Это используется для обнаружения ошибок при обмене и хранении ключей. Биты четности используются только при передаче ключа, при дальнейших преобразованиях и получении подключей биты четности не используются.

2 этап - происходит деление 56-и битной последовательности на две половинки по 28 бит каждая.

3 этап - перестановка согласно заданной таблице:

57	49	41	33	25	17	9	1	58	50	42	34	26	18	C_0
10	2	59	51	43	35	27	19	11	3	60	52	44	36	
63	55	47	39	31	23	15	7	62	54	46	38	30	22	D_0
14	6	61	53	45	37	29	21	13	5	28	20	12	4	

Рис. 11: Перестановка, создающая подключ

4 этап - циклический сдвиг 28-и битных половинок в зависимости от номера раунда, где i номер раунда.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
число сдвига	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Рис. 12: Циклический сдвиг

5 этап - перестановка, сжимающая 56 бит в 48 бит.

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

Рис. 13: Сжимающая перестановка

Мы рассмотрели первый стандартизованный алгоритм блочного шифрования. Процесс дешифрования идет по тем же таблицам, но в обратном порядке, то есть на первом раунде дешифрования мы используем подключ k_n и двигаемся от n к номеру 1.

Алгоритм DES является очень красивым и компактным алгоритмом до сих пор используемым в некоторых криптомодулях, но, в настоящее время, в том виде в каком мы его рассмотрели, у него есть существенный минус, а именно длина ключа. Длина ключа в 56 бит, в настоящее время, является не криптостойкой и для решения этой проблемы

была предложена модификация алгоритма DES, алгоритм 3DES (TRIPLE DES).

6.2 АЛГОРИТМ 3DES (TRIPLE DES)

Для решения проблемы длины ключа в алгоритме DES была предложена модификация алгоритма - 3DES, давайте рассмотрим, что она из себя представляет.

Вместо одного 56-битного ключа используется 3 ключа по 56 бит (3*56=168).

Существуют различные модификации 3DES:

- DES-EEE3;
- DES-EDE3;
- DES-EEE2;
- DES-EDE2.

Рассмотрим одну из модификаций, так как остальные работают по схожим схемам.

Мы изучим работу режима DES-EDE3.

Ключ состоит из трех частей k_1, k_2, k_3, каждая часть 56 бит.

Мы также работаем с блоками по 64 бита и также проводим 16 раундов, сам алгоритм DES не меняется.

DES-EDE3 состоит их трех шагов:

1. Мы **шифруем** наши данные алгоритмом DES, используя ключ k_1.
2. Мы **дешифруем** получившиеся данные с помощью ключа k_2.
3. Мы снова **шифруем** получившиеся данные с помощью ключа k_3.

Дешифрование будет получаться следующим образом:

1. Мы **дешифруем** наши данные алгоритмом DES, используя ключ k_3.

2. Мы **шифруем** получившиеся данные с помощью ключа k_2.

3. Мы снова **дешифруем** получившиеся данные с помощью ключа k_1.

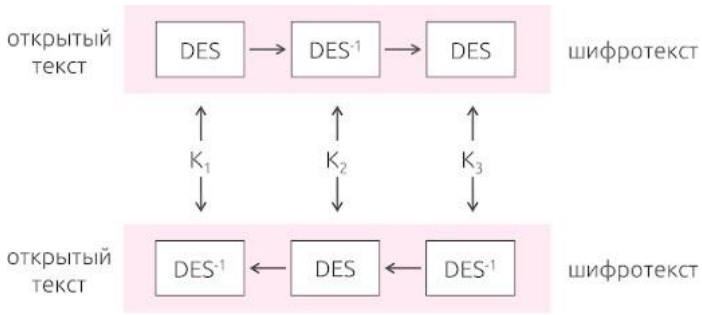

Рис. 14: Алгоритм DES-EDE3

Таким образом мы получили несомненный выигрыш в длине ключа, но при этом получили такой же проигрыш в скорости.

7 РЕЖИМЫ БЛОЧНОГО ШИФРОВАНИЯ

Рассмотрим работу режимов шифрования на примере DES, но в начале проговорим что же такое режимы шифрования.

Определение 7.0.1 *Режим шифрования — метод применения блочного шифра (алгоритма), позволяющий преобразовать последовательность блоков открытых данных в последовательность блоков зашифрованных данных. При этом для шифрования одного блока могут использоваться данные другого блока.*

Режимы шифрования обеспечивают взаимосвязь блоков, обрабатываемых шифрующим алгоритмом, между собой.

Существует четыре основных режима шифрования:

- ECB – Electronic Code Book (электронная кодовая книга);
- CBC – Cipher Block Chaining (сцепление блоков шифротекста);
- OFB – Output FeedBack (обратная связь вывода);
- CFB – Cipher FeedBack (обратная связь шифра); Давайте рассмотрим каждый из них по отдельности.

ECB – Electronic Code Book

Режим ECB работает с блоками открытого текста независимо друг от друга, обрабатывая каждый блок отдельно, что дает этому режиму ряд преимуществ:

1. Прост в обращении и быстр;
2. Обработку блоков можно распараллелить;
3. Нет накопления ошибок. Ошибка, возникшая при обработке, отразится только на битах в которых она возникла.

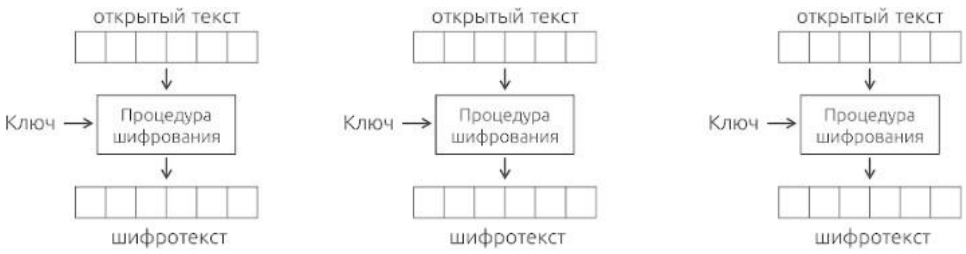

Рис. 15: Режим ECB, шифрование

Но, несмотря на это, у режима ECB есть огромный минус, а именно блоки можно незаметно удалять, так как они не связанны между собой, атакующий может блоки шифротекста менять и отследить это будет невозможно. Кроме того, большой подсказкой

для атакующего будет то, что одинаковые блоки на выходе будут выглядеть одинаково, что опасно для атак на банковские системы, так как у атакующего появляется возможность просто менять блоки с названиями банка и переводить деньги в другой банк.

Резюмируя, ECB – прост в обращении, можно работать с блоками независимо и даже распараллелить вычисления, но данный режим не защищен от атак с удалением и вставками.

CBC – Cipher Block Chaining

Режимом, решающим проблемы режима ECB, является режим CBC – Cipher Block Chaining. Режим CBC – предотвращает потери при атаке со вставкой и удалением. Получается это за счет сцепления блоков между собой. Достигается это за счет того, что на первом шаге блок открытого текста XOR-ся с вектором инициализации (вектор инициализации является случайной последовательностью равной длине блока и в секрете не держится), а на следующих этапах биты шифротекста, полученные на предыдущем шаге, XOR-ся с блоками открытого текста.

Схематически это выглядит так:

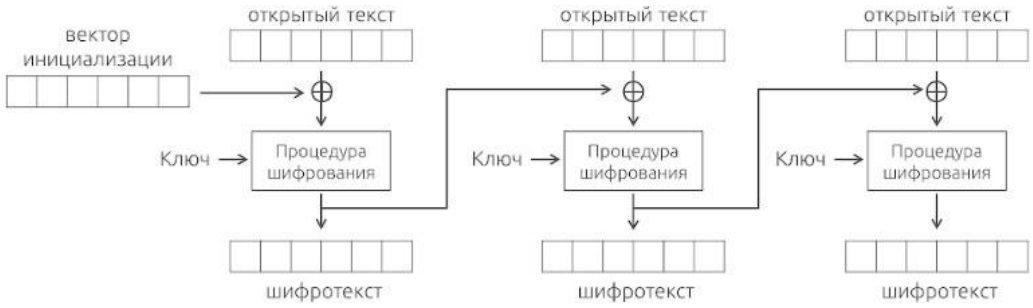

Рис. 16: Режим CBC, шифрование

Математически мы можем представить это следующим образом. Процесс шифрования:

На первом шаге $C_1 = E_k(m_1 \oplus IV)$, где IV инициализирующий вектор, а $C_i = E_k(m_i \oplus C_{i-1})$, где i номер блока начиная со второго, C_i – i- ый блок шифротекста, m_i- i-ый блок открытого текста.

Процесс дешифрования:

На первом шаге $m_1 = D_k(C_1) \oplus IV$, а далее $m_i = D_k(C_i) \oplus C_{i-1}$.

За счет связки блоков данный режим полностью решает проблему вставок и удаления блоков, но возникает другая проблема - проблема распостронения ошибки. Если в каком-то зашифрованном блоке происходит ошибка то она идет в следующий блок и происходит «размножение ошибки». Это видно на схеме.

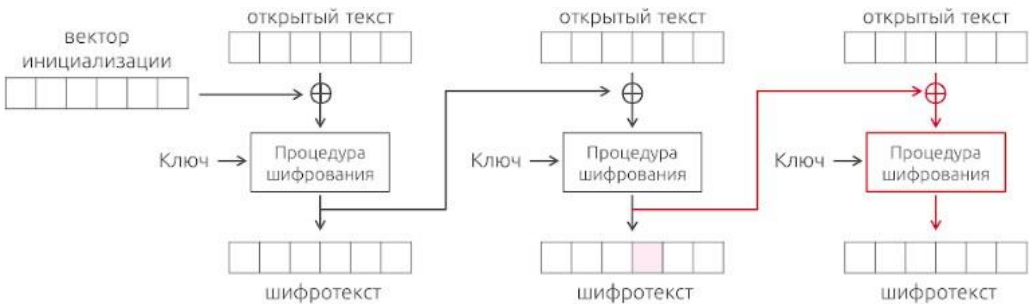

Рис. 17: Режим CBC, процесс «размножение ошибки»

Рассмотрим еще два режима, которые блочное шифрования переводят в поточное.

CFB – Cipher FeedBack

Данный режим переводит блочный шифр в поточный. Отдельная ошибка в криптограмме при этом влияет как на блок, в котором она допущена, так и на следующей блок.

На первом шаге шифруется инициализирующий вектор, а потом уже выбирается

нужное количество бит для дальнейших преобразований.

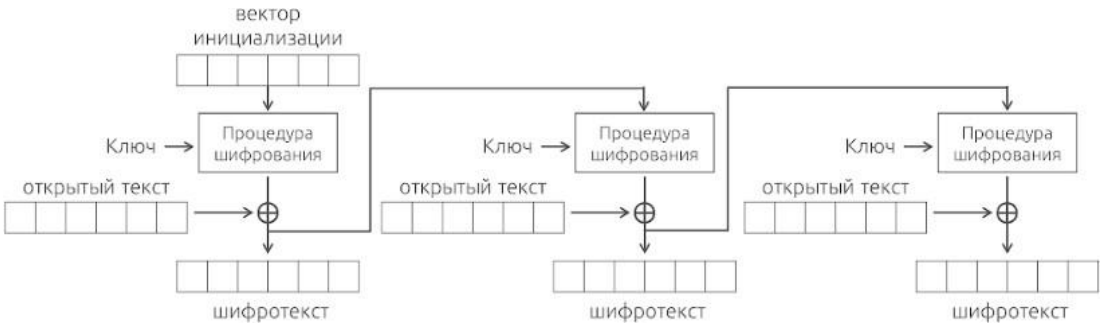

Рис. 18: Режим CFB, шифрование

Запишем процесс шифрования математически, длина блока m равна j, j фиксируется в диапазоне от 0 до n.

$$x_1 = IV$$

$$y_i = E_k(x_i), \qquad i \geq 1$$

$e_i = j$ крайних слева битов блока y_i

$$C_i = m_i \oplus e_i$$

$$x_{i+1} = C_i$$

Давайте рассмотрим последний режим OFB.

OFB - Output FeedBack

Данный режим тоже переводит блочный алгоритм шифрования в поточный. Ошибка в одном бите в шифротексте дает только один ошибочный бит в расшифрованном тексте, то есть нет распространения ошибки, что полезно при передаче аналоговых величин (звука, видео и т.п.).

Данный режим очень похож на режим CFB, единственная разница состоит в том, что открытый текст не участвует в формировании следующего шага.

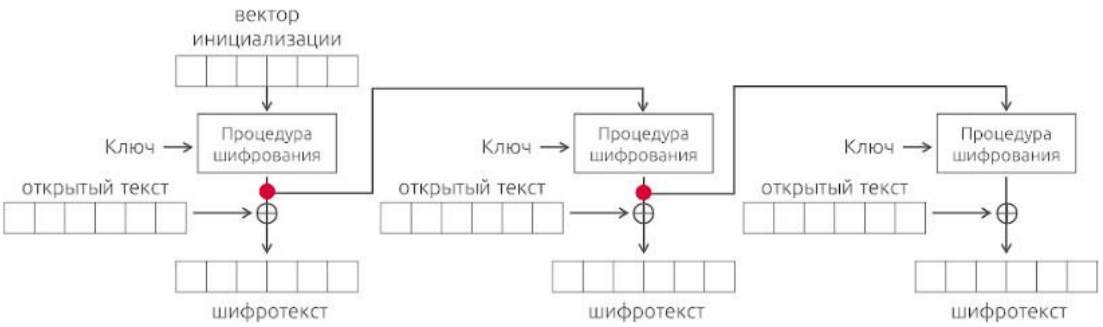

Рис. 19: Режим OFB, шифрование

Математически это выглядит следующим образом:

$$x_1 = IV$$

$$y_i = E_k(x_i), \qquad i \geq 1$$

$e_i = j$ крайних слева битов блока y_i

$$C_i = m_i \oplus e_i$$

$$x_{i+1} = y_i$$

На этом мы заканчиваем наше первое знакомство с блочным шифрованием, далее нас ожидают другие блочные алгоритмы, мы познакомимся с алгоритмом Rijndael и алгоритмом ГОСТ 28147-89.

8 ЗАКЛЮЧЕНИЕ

На данной лекции мы начали наше знакомство с блочным симметричным шифрованием, рассмотрели идею блочного и поточного шифрования. Познакомились с принципом Кергоффса и теоремой Шеннона, ввели понятия абсолютно стойкого шифра, рассмотрели шифр Вернама – шифр-блокнот. Узнали этапы развития блочного шифрования, познакомились с сетью Фей- стеля, понятием SP- network. Рассмотрели подробно работу алгоритма DES, являвшегося стандартом шифрования с 1977 года, разобрали идею работы 3DES и узнали что такое режимы работы блочного шифра.