

Министерство образования Российской Федерации
Ярославский государственный университет им. П.Г. Демидова
Кафедра компьютерных сетей

**Математические методы защиты
информации**

Методические указания

Ярославль 2011

ББК В 311
У 68
УДК 517.2

Составители **М.В. Краснов**

Математические методы защиты информации (ч. 2.): Метод. указания / Сост. М.В. Краснов; Яросл. гос. ун-т. Ярославль, 2011. 25 с.

Основное использование вычислительной техники связано с хранением информации. Естественно, возникает задача защиты информации от несанкционированного использования. В работе сформулированы основные идеи создания блочных симметричных алгоритмов. Наиболее известные из них подробно описаны. Рассмотрена проблема управления ключами, которая возникает при работе с симметричными шрифтами.

Указания предназначены для помощи студентам при изучении курса «Математические методы защиты информации» и могут быть использованы как справочный материал при выполнении домашних заданий, курсовых работ и при подготовке к экзаменам.

Рецензент: кафедра компьютерных сетей Ярославского государственного университета им. П.Г. Демидова.

© Ярославский государственный университет им. П.Г. Демидова, 2011

© Краснов М.В. 2011

Математические методы защиты информации (ч. 2.)

Составители: Краснов Михаил Владимирович

Корректор А.А. Антонова

Лицензия ЛР № 020319 от 30.12.96.

Подписано в печать 22.05.2000. Формат 60×84/16. Бумага тип.
Усл. печ. л. 1,6. Уч.-изд. л.1,73. Тираж 100 экз. Заказ

Оригинал-макет подготовлен в редакционно-издательском отделе
Ярославского государственного университета.

Отпечатано на ризографе

Ярославский государственный университет.
150000, Ярославль, ул. Советская, 14.

Введение

В настоящее время использование электронной вычислительной техники в различных областях человеческой деятельности все более и более возрастает. Однако чаще всего вычислительная техника используется для хранения и передачи информации. Естественно, возникает задача защиты информации от несанкционированного использования. Среди способов защиты информации одним из наиболее распространенных методов является криптографический метод. Он предусматривает такое преобразование информации, при котором она становится доступной для прочтения лишь обладателю некоторого секретного параметра (ключа).

Опишем задачу защиты информации с помощью криптографического метода. Отправитель хочет послать получателю по каналу, который не является безопасным, текст T . Взломщик хочет перехватить передаваемую информацию. Отправителю нужно так преобразовать сообщение, чтобы взломщик не смог прочитать исходный текст T из перехваченного сообщения, а получатель мог бы за приемлемое время восстановить исходный текст из полученного сообщения.

Чтобы решить поставленную задачу, отправитель шифрует исходный текст T с помощью некоторого преобразования E_k , где k - ключ шифрования. Шифртекст $C = E_k(T)$ передается по каналу связи.

Получатель должен уметь расшифровать шифртекст - восстановить исходный текст T с помощью некоторого преобразования $D_{\tilde{k}}$, где \tilde{k} - ключ расшифрования:

$$T = D_{\tilde{k}}(C).$$

Если отправитель знает ключ k , то он может зашифровывать информацию; если получатель знает ключ \tilde{k} , то он может расшифровывать сообщение.

Перед взломщиком стоит более сложная задача: он должен найти ключ \tilde{k} или свой способ дешифровки.

Алгоритмы, используемые в современных криптосистемах, можно разделить на два типа:

- симметричные, в которых ключ расшифрования легко находится по ключу шифрования;
- с открытым ключом, в которых ключ расшифрования трудно найти даже при известном ключе шифрования.

С другой стороны симметричные шифры можно разделить на два типа шифров: блочное и потоковое шифрование.

Блочное шифрование - в этом случае исходное сообщение разбивается на блоки фиксированной размерности (например 64 или 128 бит), которые потом и шифруются.

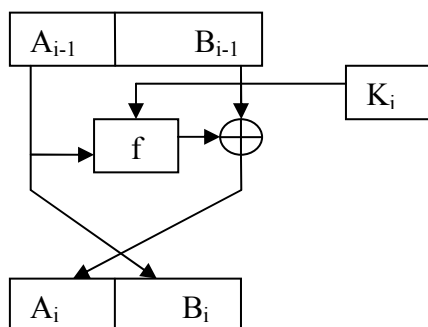
Потоковое шифрование используется, когда нельзя разбить исходное сообщение на блоки. Например, каждый символ исходного сообщения должен быть зашифрован, не дожидаясь остальных данных.

В представленных методических указаниях основное внимание будет уделено симметричным блочным шифрам.

Можно сказать, что блочные шифры реализуются путем многократного применения к блокам открытого текста некоторых базовых преобразований. Обычно используются базовые преобразования двух типов — это локальные преобразования над отдельными частями шифруемых блоков и простые преобразования, переставляющие между собой части шифруемых блоков. Первое преобразование усложняет восстановление взаимосвязи статистических и аналитических свойств открытого и зашифрованного текстов, а второе преобразование распространяет влияние одного знака открытого текста на большое число знаков шифра.

Сформулируем основные конструкции, которые часто используются в процессе создания симметричного блочного шифра:

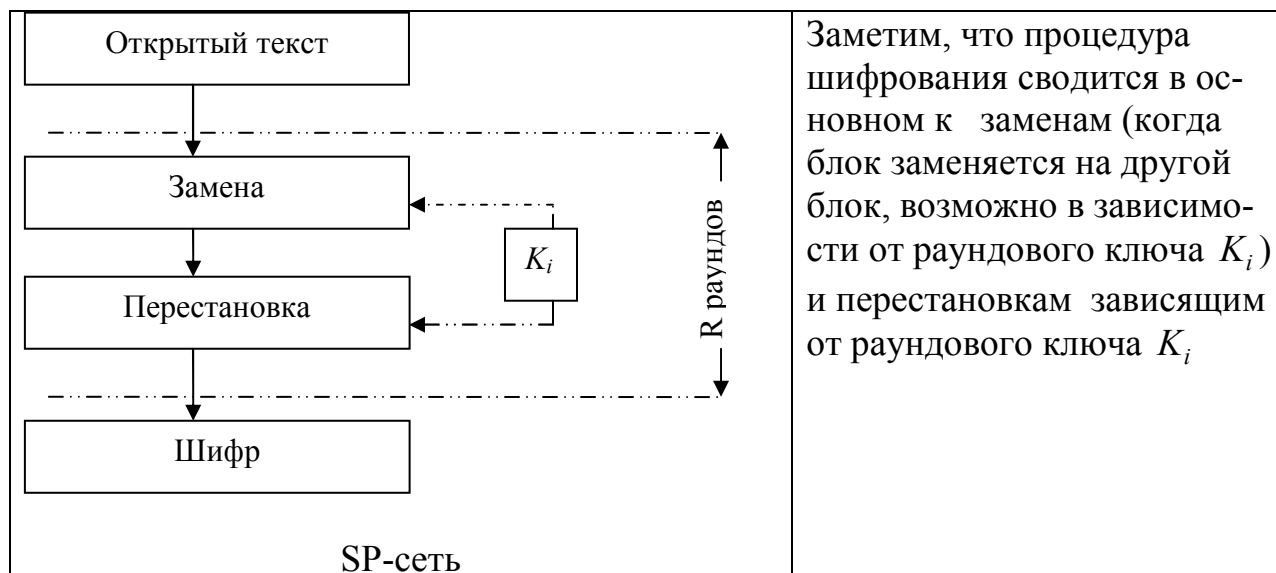
- сеть Фейстеля предполагает разбиение рассматриваемого вектора на несколько подблоков (например на два), один из блоков обрабатывается некоторой функцией f , а результат складывается по модулю два с одним или несколькими из оставшихся подблоков.



Легко заметить, что в качестве дополнительного параметра функции f является раундовый ключ K_i . Раундовый ключ получается из исходного ключа обычно путем развертывания.

сеть Фейстеля

- SP-сети. Обработка данных сводится, в основном, к заменам (например, с помощью таблицы замен) и перестановкам, которые зависят от ключа.



Самая простая сеть состоит из слоёв двух типов, используемых многократно по очереди. Первый тип слоя — Р-слой, состоящий из Р-блока большой разрядности, за ним идёт второй тип слоя — S-слой, представляющий собой большое количество S-блоков малой разрядности.

Также популярны алгоритмы, построенные на основе SP-сети со структурой «квадрат». В этом случае обрабатываемый в процессе работы алгоритма блок данных представляется в виде двумерного байтового массива. Криптографические преобразования могут выполняться как над отдельными байтами массива, так и над его строками или столбцами.

Режимы работы блочных шифров.

Обычно при реальном применении используется не только сам алгоритм шифрования-дешифрования, но и специальные режимы работы блочных симметричных шифров. Наиболее популярны четыре режима: электронная шифрованная книга, сцепление шифрованных блоков, обратная связь по шифру, обратная связь по выходу¹. Этих режимов хватает, чтобы использовать шифр практически в любой области, для которой этот алгоритм подходит. Алгоритм шифрования или дешифрования является базовым блоком защиты передачи данных в этих режимах. Опишем эти режимы на примере процесса шифрования открытого текста разбитого на n – битовые блоки.

название режима	описание режима	применение режима
Электронная	Каждый n – битовый блок открытого текста шифруется независимо от других с одним и тем	Защищенная передача от-

¹ Используются и другие режимы, например, режим сцепления блоков вида:

$$y_t = E_r(x_t \oplus y_{t-1} \oplus y_{t-2} \oplus \dots \oplus y_1 \oplus y_0), \quad t = 1, 2, \dots, \text{где}$$

y_t - t -ый блок шифртекста;

y_0 - вектор инициализации;

E_r - алгоритм шифрования с ключом r ;

x_t - t -ый блок открытого текста

или режим генерации кода целостности сообщения и т.д.

шифрованная книга ЕСВ	<p>же ключом</p> $y_t = E_r(x_t), \quad t = 1, 2, \dots, \text{ где}$ <p>y_t - t-ый блок шифртекста; E_r - алгоритм шифрования с ключом r; x_t - t-ый блок открытого текста.</p>	дельных значений (например, ключа шифрования)
Сцепление шифрованных блоков СВС	<p>Рассматриваемый режим можно описать следующей формулой.</p> $y_t = E_r(x_t \oplus y_{t-1}), \quad t = 1, 2, \dots, \text{ где}$ <p>y_t - t-ый блок шифртекста; E_r - алгоритм шифрования с ключом r; y_0 - вектор инициализации; x_t - t-ый блок открытого текста.</p>	Поблочная передача данных общего назначения Аутентификация ²
Обратная связь по шифру СFB	<p>Шифрование выполняется блоками по k бит ($k = 1 \dots n$) по формуле $C_i = M_i \oplus P_i$, где</p> <p>y_t - блок крипто текста; x_t - блок открытого текста; P_i - блок псевдослучайной последовательности.</p> <p>Идея использования этого режима заключается в том, чтобы получить псевдослучайную последовательность. Это достигается следующим образом:</p> <ul style="list-style-type: none"> • вначале входной блок алгоритма содержит n – битовый вектор инициализации и k старших бит, полученного блока шифртекста используются в качестве блока псевдослучайной последовательности; • полученный на предыдущем шаге шифрованный текст используется как часть входных данных для блока шифрования и т.д. 	Потоковая передача данных общего назначения. Аутентификация

² Аутентификация – процедура установления соответствия параметров, характеризующих пользователя, процесс или данные, заданным критериям.

	<p>сдвиг k бит</p> <pre> graph TD KeyStream[kлюч r бит] --> XOR((⊕)) Plaintext[исходный текст k бит] --> XOR XOR --> Ciphertext[крипто текст k бит] Ciphertext -- "сдвиг k бит" --> Shift[n-k k] Shift --> EncBlock[Блок шифрования] EncBlock --> KeyStream </pre>	
Обратная связь по выходу OFB	<p>Подобен CFB, но в качестве входных данных для алгоритма шифрования используются ранее полученные выходные данные блока шифрования</p> <pre> graph TD KeyStream[kлюч r бит] --> XOR((⊕)) Plaintext[исходный текст k бит] --> XOR XOR --> Ciphertext[крипто текст k бит] Ciphertext -- "сдвиг k бит" --> Shift[n-k k] Shift --> EncBlock[Блок шифрования] EncBlock --> KeyStream </pre>	Потоковая передача данных по каналам с помехами.

Несколько блочных симметричных шифров.

Алгоритм DES

Общие положения

Стандарт шифрования данных DES был опубликован в 1977 году в США и использовался для защиты от несанкционированного доступа к важной, но не секретной информации в различных организациях. Официально Des считался стандартом до 31 декабря 1998 г.

В криптосистеме DES используется блочный принцип шифрования двоичного текста. Размер блока шифрования 64 бита. Размер ключа также составляет 64 бита. При этом каждый восьмой бит ключа является служебным (используется в качестве бита проверки на четность для семи предыдущих бит) и в шифровании не участвует. Таким образом, полезный размер ключа составляет 58 бит.

Процесс шифрования

Алгоритм шифрования DES состоит из трех основных этапов:

Первый этап – преобразование исходного текста. Биты исходного сообщения z подвергаются начальной перестановке P в соответствии с таблицей

P	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

Затем полученный вектор $z_0 = P(z)$ представляется в виде $z_0 = L_0 R_0$, где L_0 левая половина из 32 бит, а R_0 правая половина из 32 бит.

Второй этап – непосредственно само шифрование. Полученное сообщение $z_0 = L_0 R_0$ подвергаются преобразованию по следующей схеме:

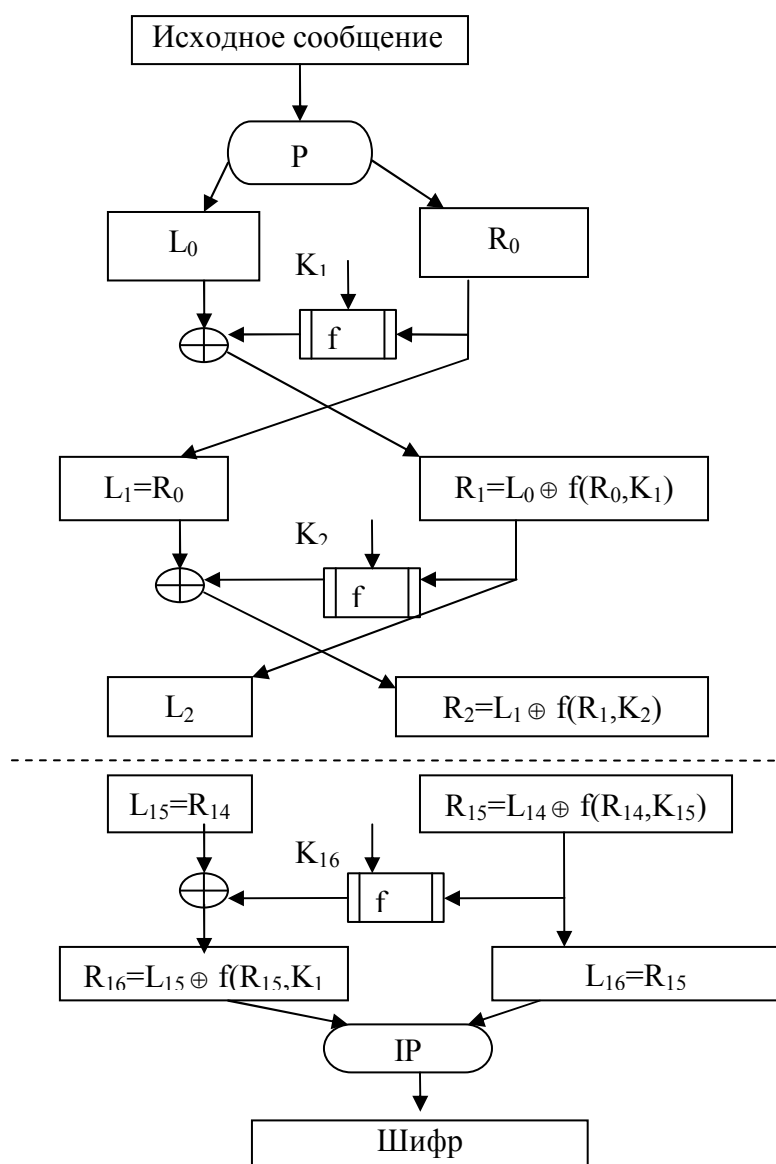
$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, K_i), \end{cases} \quad i = 1, \dots, 16$$

Функция f и процесс создания ключей K_i будет описан ниже.

Третий этап – завершающее преобразование. Сообщение $L_{16} R_{16}$ перемешивается подстановкой IP результат ее и есть зашифрованное сообщение, другими словами $y = IP(R_{16} L_{16})$.

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

Общая схема алгоритма Des может быть представлена следующей схемой:



Функция f .

Рассмотрим функцию f более подробно. Функция имеет два аргумента. Первый из них R_{i-1} - это вектор в 32 бита, второй K_i вектор в 48 бит. Выходом функции служит вектор в 32 бита. Работу же f можно проиллюстрировать следующей схемой.

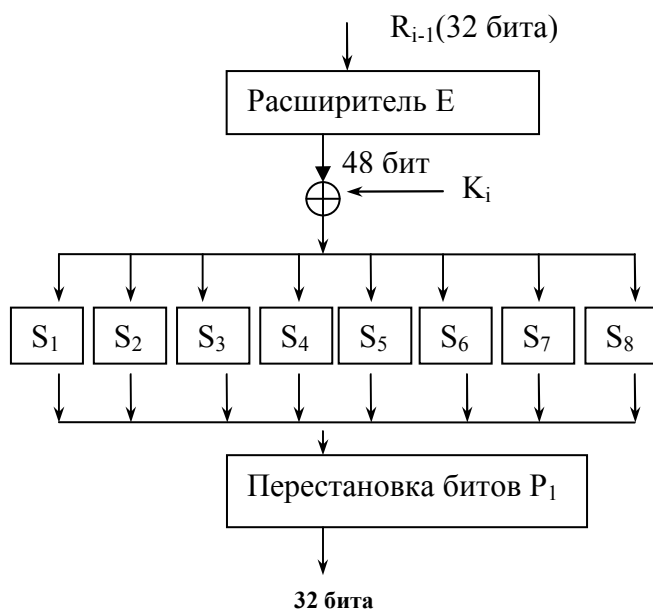


Схема функции $f(R_{i-1}, K_i)$

Работа функции f состоит в выполнении 4 операций:

Шаг 1. Первый аргумент с помощью функции расширения $E(R_{i-1})$ преобразуется в 48 битовый вектор. Эта процедура одна и та же для всех раундов(итераций) и задается с помощью следующей таблицы

$E(R_{i-1})$	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

Шаг 2. Вычисляется сумма $E(R_{i-1}) \oplus K_i$ и записывается в виде конкатенации восьми 6 битовых слов $E(R_{i-1}) \oplus K_i = B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8$.

Шаг 3. Каждое слово B_i поступает на соответствующий S-блок. Блок S_i преобразует 6 битовый вход в 4 битовый выход C_i . S-блок есть матрица 4×16 с целыми элементами от 0 до 15. Выбор элемента в матрице S_i осуществляется следующим образом:

Пусть на вход матрицы S_i поступает 6-битовый блок $B_i = b_1 b_2 b_3 b_4 b_5 b_6$, тогда число $b_1 b_6$ указывает номер строки, а $b_2 b_3 b_4 b_5$ номер столбца. Тем самым найден некоторый элемент матрицы S_i . Выходом C_i является двоичное представление этого элемента.

номер столбца		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	S_1
номер строки	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	6	13	

номер столбца		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
номер строки	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S_2
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	

номер столбца		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
номер строки	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	S_3
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	

номер столбца		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
номер строки	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	S_4
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	

номер столбца		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
номер строки	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	S_5
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	

номер столбца		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
номер строки	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	S_6
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	

номер столбца		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
номер строки	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	S_7
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	

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

номер столбца		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
номер строки	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	S_8
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

В результате получаем вектор $C = C_1C_2C_3C_4C_5C_6C_7C_8$, где $C_i = S_i(B_i)$.

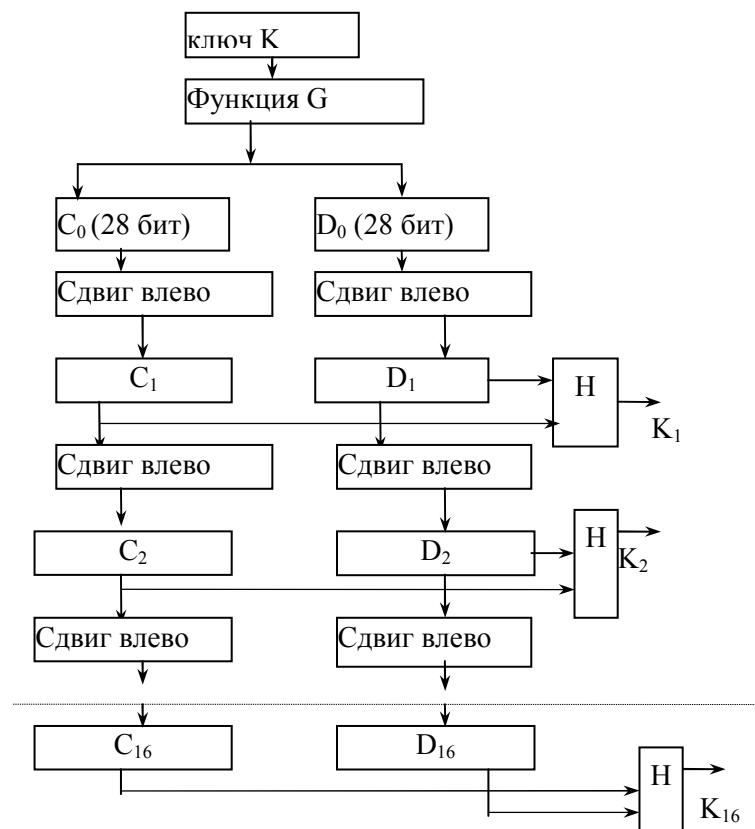
Шаг 4. Выход $C = C_1 \dots C_8$ перемешивается фиксированной подстановкой P_1 .

P_1	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

Описание функции f окончено.

Формирование ключа.

Легко заметить, что в каждом раунде используется новое значение ключа K_i (длиной 48 бит). Новое значение ключа K_i вычисляется из начального ключа K . Процесс формирования ключа можно представить с помощью следующей схемы:



Как уже отмечалось, в 64-битовом ключе K удаляется каждый восьмой бит. После этого оставшиеся биты подвергаются перестановки с помощью функции G .

функция G	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
	53	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

Результат этой перестановки делится на две половины C_0 и D_0 (по 28 битов в каждой). Очередные значения C_i D_i вычисляются по схеме:

$$\begin{cases} C_i = M_i(C_{i-1}) \\ D_i = M_i(D_{i-1}) \end{cases}, \text{ где } M_i - \text{циклический сдвиг влево на одну позицию, если } i = 1, 2, 9, 16.$$

При других значениях i , M_i - циклический сдвиг влево на две позиции.

Наконец, две части C_i и D_i соединяются вместе и подаются на вход перестановки H , выходом которой и будет 48-битовый подключ i – го раунда.

перестановка H	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

Заметим, что существуют 64-битовые последовательности, которые не рекомендуется использовать в качестве ключей, например вектор $K = \{0000\}^3$.

Формирование ключа окончено.

Процесс дешифрования.

Процесс дешифрования является инверсным по отношению к процессу шифрования. Это означает, что дешифрование осуществляется тем же алгоритмом и ключом, но все действия выполняются в обратном порядке. Другими словами сначала расшифрованные данные переставляются в соответствии с матрицей IP , а затем над последовательностью битов $R_{16}L_{16}$ выполняются действия, которые можно описать схемой

$$\begin{cases} R_{i-1} = L_i \\ L_{i-1} = R_i \oplus f(L_i, K_i) \end{cases} \quad i = 1, \dots, 16$$

³ $\{x\}$ число в шестнадцатеричной системе счисления.

Алгоритм AES

Общие положения

AES представляет собой алгоритм шифрования 128-битных блоков данных ключами по 128, 192 и 256 бит. AES является упрощенной версией алгоритма Rijndael (этот алгоритм был разработан Винсентом Райманом и Йоан Дамен). Алгоритм Rijndael победил в конкурсе о выборе стандарта шифрования США и был видоизменен для большей стандартизации и назван AES. Алгоритм AES в 2002г. был объявлен стандартом шифрования.

Алгоритм оперирует байтами, которые рассматриваются как элементы конечного поля $GF(2^8)$. Напомним, что элементами поля $GF(2^8)$ можно описать как множество многочленов степень которых меньше 8, так например элементу-вектору (10001011) будет соответствовать многочлен $x^7 + x^3 + x + 1$. Поскольку мы работаем в поле, то зададим операции сложения и умножения:

- сложение \oplus - суть операция поразрядной XOR.

Пример $(x^7 + x + 1) \oplus (x^6 + x + 1) = x^7 + x^6$ ■

- умножение $*$ - это операция умножения многочленов со взятием результата по модулю некоторого неприводимого многочлена и использованием операции XOR при приведении подобных членов. Авторы алгоритма рассматривают в качестве неприводимого многочлена

$$\varphi(x) = x^8 + x^4 + x^3 + x + 1.$$

Пример

$$((x^7 + x + 1) * (x^6 + x^4 + x^2 + x + 1)) \bmod (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1. \blacksquare$$

Раундовые преобразования работают с четырехбайтовыми словами. Этому слову можно поставить в соответствие многочлен $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$, где $a_i \in GF(2^8)$. Рассмотрим как будет происходить сложение и умножение четырехбайтовых слов $a(x)$ и $b(x)$, где

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0, \quad b(x) = b_3x^3 + b_2x^2 + b_1x + b_0:$$

- сложение $a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$
- умножение $c(x) = a(x) \otimes b(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$,

где

$$\begin{aligned} c_6 &= a_3 * b_3 & c_0 &= a_0 * b_0 & c_5 &= a_2 * b_3 \oplus b_2 * a_3 \\ c_3 &= a_0 * b_3 \oplus a_1 * b_2 \oplus a_2 * b_1 \oplus a_3 * b_0 & c_2 &= a_0 * b_2 \oplus a_1 * b_1 \oplus a_2 * b_0 \\ c_4 &= a_3 * b_1 \oplus a_2 * b_2 \oplus a_1 * b_3 & c_1 &= a_0 * b_1 \oplus a_1 * b_0 \end{aligned}$$

Для того, чтобы результат умножения был снова представлен в виде четырехбайтового слова, его надо взять по модулю многочлена $x^4 + 1$. Следовательно, в результате получим вектор $d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$, где

$$\begin{aligned} d_0 &= a_3 * b_1 \oplus a_2 * b_2 \oplus a_1 * b_3 \oplus a_0 * b_0 & d_1 &= a_0 * b_1 \oplus a_1 * b_0 \oplus a_2 * b_3 \oplus b_2 * a_3 \\ d_2 &= a_0 * b_2 \oplus a_1 * b_1 \oplus a_2 * b_0 \oplus a_3 * b_3 & d_3 &= a_0 * b_3 \oplus a_1 * b_2 \oplus a_2 * b_1 \oplus a_3 * b_0 \end{aligned}$$

Предварительная обработка данных.

Промежуточные результаты преобразований, выполняемые в рамках алгоритма, называются состояниями(State). Состояние обычно представляют в виде прямоугольного массива байтов (а именно 16 байтов 4 строки и 4 столбца). Входные данные для шифра обозначаются как байты состояния в порядке $s_{00}, s_{10}, s_{20}, s_{30}, s_{01}, s_{11}, \dots$

После завершения процесса шифрования выходные данные получают из байтов состояния в том же порядке.

Ключ шифрования аналогичным образом представляется в виде прямоугольного байтового массива с 4 строками. Количество столбцов (Nk) равно длине ключа деленному на 32 бита. Поскольку ключ также как и входной блок подаются в виде одномерного массива, то и заполнение байтовых массивов происходит одинаково: заполнение происходит вначале по столбцам, а затем по строкам.

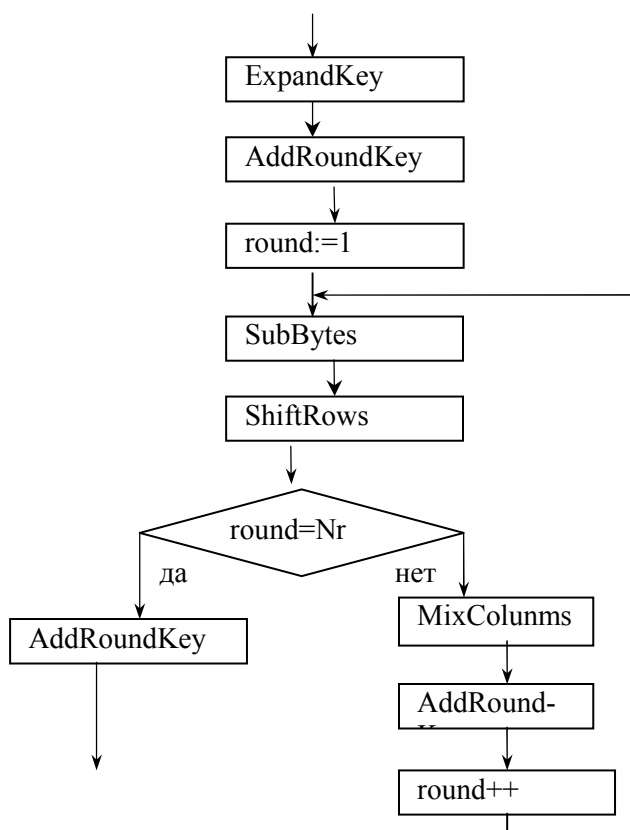
Представление 128 битовой исходной строки $s_{00}, s_{10}, s_{20}, s_{30}, s_{01}, s_{11}, \dots$ в виде массива. s_{ij} - байт			
s_{00}	s_{01}	s_{02}	s_{03}
s_{10}	s_{11}	s_{12}	s_{13}
s_{20}	s_{21}	s_{22}	s_{23}
s_{30}	s_{31}	s_{32}	s_{33}

Представление 128 битового ключа $k_{00}, k_{10}, k_{20}, k_{30}, k_{01}, k_{11}, \dots$ в виде массива. k_{ij} - байт			
k_{00}	k_{01}	k_{02}	k_{03}
k_{10}	k_{11}	k_{12}	k_{13}
k_{20}	k_{21}	k_{22}	k_{23}
k_{30}	k_{31}	k_{32}	k_{33}

В зависимости от длины ключа количество раундов(итераций) будет различным (размер ключа 128 бит - 10 раундов; 196 бит - 12 раундов; 256 бит

Процесс шифрования

Общая схема шифрования в алгоритма AES может быть представлена следующим образом:



Обозначения используемые в схеме шифрования:

round - текущий раунд

Nr - количество раундов

ExpandKey - вычисление ключей для всех раундов

AddRoundKey - сложение раундового ключа с состояниями (State)

SubBytes - замены байтов с помощью побайтовой подстановки

ShiftRows - побайтовый циклический сдвиг строк массива State на различное количество байт

MixColumns - перемешивание столбцов из массива State

Опишем каждую процедуру более подробно

Процедура *SubBytes*

Рассматриваемое преобразование заключается в замене каждого байта $\{xy\}$, которая выполняются с помощью S-блоков. В алгоритме два типа S-блоков. Один тип применяется для шифрования, а другой для дешифрования. Таблица замены S-блока состоит из двух преобразований входного байта:

1. Вычисляется мультипликативно обратный элемент в поле $GF(2^8)$ и записывается как новый байт $x = (x_7, \dots, x_0)$. По соглашению элемент $\{00\}$ переходит сам в себя.
2. Над полем $GF(2)$ применяется преобразование вида

$$\begin{bmatrix} \tilde{x}_0 \\ \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \\ \tilde{x}_4 \\ \tilde{x}_5 \\ \tilde{x}_6 \\ \tilde{x}_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Пример. Найдем замену для байтов {01}, {02}

Элемент	{01}		{02}	
результат	в виде многочлена	в виде числа	в виде многочлена	в виде числа
1 преобразование	1	{01}	$x^7 + x^3 + x^2 + 1$	10001101
2 преобразование	$x^6 + x^5 + x^4 + x^3 + x^2$	{7c}	$x^6 + x^5 + x^4 + x^2 + x + 1$	{77}

Однако чаще пользуются уже готовой таблицей.

Таблица подстановок S процедуры *SubBytes*

		{y}															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
{x}	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	fl	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Например, байт {xy} = {43} заменится на байт {1a}

Описание процедуры *SubBytes* завершено

Процедура *ShiftRows*

Рассматриваемое преобразование заключается в циклическом сдвиге строк состояния (State). Каждая из ее строк сдвигается на свое число позиций. Первая строка остается неизменной. Во второй производится сдвиг на 1 байт, то есть первый байт переносится в конец. В третьей — сдвиг на 2 байта, в четвертой — на 3.

Пример

До процедуры <i>ShiftRows</i>					После процедуры <i>ShiftRows</i>			
{d4}	{e0}	{b8}	{1e}		{d4}	{e0}	{b8}	{1e}
{27}	{bf}	{b4}	{41}		{bf}	{b4}	{41}	{27}
{11}	{98}	{5d}	{52}		{5d}	{52}	{11}	{98}
{ae}	{f1}	{e5}	{30}		{30}	{ae}	{f1}	{e5}

Описание процедуры *ShiftRows* завершено.

Процедура *MixColumns*

В этом преобразовании столбцы состояния (State) рассматриваются как многочлены над $GF(2^8)$ и умножаются по модулю $x^4 + 1$ на многочлен

$g(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$. Это можно представить в матричном виде

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}, \quad 0 \leq c \leq 3,$$

где c — номер столбца из State

Пример Применим процедуру *MixColumns* к вектору $(e0, b4, 52, ae)$, другими словами мы должны вычислить $d(x) = a(x) \otimes g(x)$, где

$$a(x) = \{ae\}x^3 + \{52\}x^2 + \{b4\}x + \{e0\}.$$

элемент из поля $GF(2^8)$	двоичный вектор	многочлен
{03}	(00000011)	$x + 1$
{02}	(00000010)	x
{ae}	(10101110)	$x^7 + x^5 + x^3 + x^2 + x$
{52}	(01010010)	$x^6 + x^4 + x$
{b4}	(10110100)	$x^7 + x^5 + x^4 + x^2$
{e0}	(11100000)	$x^7 + x^6 + x^5$

$$d_0 = a_3 * g_1 \oplus a_2 * g_2 \oplus a_1 * g_3 \oplus a_0 * g_0 = \{ae\} \oplus \{52\} \oplus \{b4\} * \{03\} \oplus \{e0\} * \{02\} =$$

$$(x^7 + x^5 + x^3 + x^2 + x) \oplus (x^6 + x^4 + x) \oplus ((x+1) * (x^7 + x^5 + x^4 + x^2)) \oplus (x * (x^7 + x^6 + x^5)) =$$

$$(x^7 + x^5 + x^3 + x^2 + x) \oplus (x^6 + x^4 + x) \oplus (x^7 + x^6 + x^2 + x + 1) \oplus (x^7 + x^6 + x^4 + x^3 + x + 1) =$$

$$x^7 + x^6 + x^5 = (11100000) = \{e0\}.$$

Напомним, что произведение многочленов $r(x) * h(x)$ берется по модулю $\varphi(x) = x^8 + x^4 + x^3 + x + 1$.

$$d_1 = a_0 * g_1 \oplus a_1 * g_0 \oplus a_2 * g_3 \oplus g_2 * a_3 = \{e0\} \oplus (\{b4\} * \{02\}) \oplus (\{52\} * \{03\}) \oplus \{ae\} = \{cb\}$$

$$d_2 = a_0 * g_2 \oplus a_1 * g_1 \oplus a_2 * g_0 \oplus a_3 * g_3 = \{e0\} \oplus \{b4\} \oplus (\{02\} * \{52\}) \oplus (\{ae\} * \{03\}) = \{19\}$$

$$d_3 = a_0 * g_3 \oplus a_1 * g_2 \oplus a_2 * g_1 \oplus a_3 * g_0 = (\{e0\} * \{03\}) \oplus \{b4\} \oplus \{52\} \oplus (\{ae\} * \{02\}) = \{9a\}.$$

В результате получили вектор $(e0, cb, 19, 9a)$ ■

Описание процедуры *MixColumns* завершено.

Процедура *AddRoundKey*

Данная процедура добавляет раундовый ключ к столбцам матрицы State посредством побитовой операции XOR:

$$[s'_{0c}, s'_{1c}, s'_{2c}, s'_{3c}] = [s_{0c}, s_{1c}, s_{2c}, s_{3c}] \oplus [w_{4*round+c}], \text{ где } 0 \leq c \leq 3 \text{ и } w_i - \text{столбцы ключа.}$$

Раундовый ключ вырабатывается из ключа шифрования посредством алгоритма выработки ключей (процедура *ExpandKey*).

Пример

таблица состояния State		раундовый ключ		После процедуры AddRoundKey
{04} {e0} {48} {28}		{a0} {88} {23} {2a}		{a4} {68} {6b} {02}
{66} {cb} {f8} {06}	⊕	{fa} {54} {a3} {6c}	=	{9c} {9f} {5b} {6a}
{81} {19} {d3} {26}		{fe} {2c} {39} {76}		{7f} {35} {ea} {50}
{e5} {9a} {7a} {4c}		{17} {b1} {39} {05}		{f2} {2b} {43} {49}

Описание процедуры *AddRoundKey* завершено.

Процедура *ExpandKey*

Процедура состоит из двух операций:

- ключ шифрования преобразуется в расширенный ключ
- из расширенного ключа выбираются раундовые ключи.

Рассмотрим операции более подробно:

- Расширенный ключ представляет собой массив $w[i]$ из $4(N_r + 1)$ 4-байтовых слов. Формирование этого массива можно задать следующим псевдокодом

```

KeyExpansion(byte key[4*Nk], word w[4*(Nr+1)],Nk)
begin
word temp
i = 0
while ( i < Nk )
    w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
    i = i + 1
end while
i = Nk
while ( i < 4*(Nr+1))
    temp = w[i-1]
    if (i mod Nk = 0)
        temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
    else if ( Nk > 6 and i mod Nk = 4)
        temp = SubWord(temp)
    end if
    w[i] = w[i-Nk] xor temp
    i = i + 1
end while
end

```

Поясним функции, которые встретились в псевдокоде:

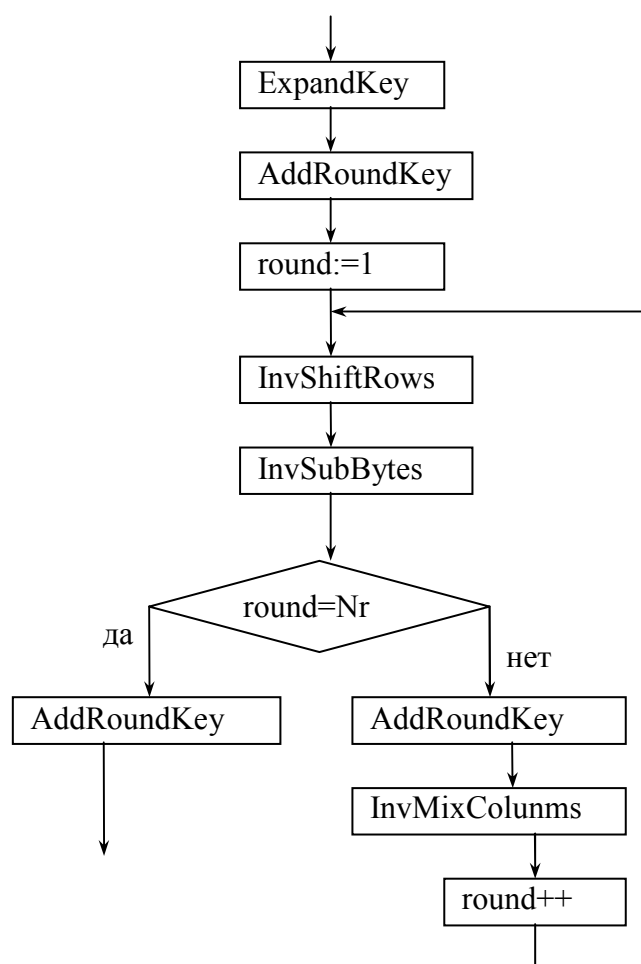
- функция *RotWord()*-осуществляет побайтовый сдвиг 32-разрядного слова по формуле $(a_0a_1a_2a_3) \rightarrow (a_1a_2a_3a_0)$;
- функция *SubWord()*-осуществляет побайтовую замену, используя подстановки из функции *SubBytes()*;
- функция *Rcon[j]* возвращает 4-байтовое слово старший байт в котором равен 2^{j-1} по модулю $\varphi(x)$, другими словами получим вектор $(2^{j-1}000000)$.

- Выбор раундового ключа. Раундовый ключ i получается из слов массива раундового ключа от $w[4i]$ и до $w[4(i+1)]$.

Описание процедуры *ExpandKey* завершено.

Процесс дешифрования

При дешифровании все преобразования производятся в обратном порядке. Используются следующие обратные преобразования вместо соответствующих шифрующих:



Обозначения используемые в схеме дешифрования:

round - текущий раунд;

Nr - количество раундов;

ExpandKey - вычисление ключей для всех раундов;

AddRoundKey - сложение раундового ключа с состояниями (State);

InvSubBytes - подстановка байтов с помощью обратной таблицы подстановок;

InvShiftRows - циклический сдвиг строк в форме (State) на различные величины;

InvMixColumns - смешивание данных внутри каждого столбца формы State.

Процедуры *ExpandKey* и *AddRoundKey* остаются неизменными. Ключи раунда используются в обратном порядке

Процедура *InvSubBytes*

Эта процедура аналогична процедуре *SubBytes*, только таблица подстановок имеет вид

x		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
y	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb

1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Такую табличную замену можно выполнить, применив к входному байту преобразование, обратное второму действию альтернативной операции SubBytes, после чего вычислить мультипликативно обратный элемент в поле $GF(2^8)$

Описание процедуры *InvSubBytes* завершено.

Процедура *InvShiftRows*

Это преобразование обратное преобразованию *ShiftRows*. Первая строка формы (State) остается неизменной. Вторая строка циклически сдвигается вправо на 1 байт. Третья — на 2, четвертая — на 3.

Описание процедуры *InvShiftRows* завершено.

Процедура *InvMixColumns*

В этом преобразовании столбцы состояния (State) рассматриваются как многочлены над $GF(2^8)$ и умножаются по модулю $x^4 + 1$ на многочлен

$z(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$. Это можно представить в матричном виде

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} \{0e\} & \{0b\} & \{0d\} & \{09\} \\ \{09\} & \{0e\} & \{0b\} & \{0d\} \\ \{0d\} & \{09\} & \{0e\} & \{0b\} \\ \{0b\} & \{0d\} & \{09\} & \{0e\} \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}, \quad 0 \leq c \leq 3,$$

где c — номер столбца из State

Легко заметить, что

$$g(x) \otimes z(x) = (\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}) \otimes (\{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}) = \{01\}.$$

Алгоритм RC6

Алгоритм был предложен Рональдом Ривестом в 1988 году, принимал участие в конкурсе AES, где и прошел в финал.

Алгоритм имеет достаточно, гибкую структуру, его параметрами кроме секретного ключа K являются:

- размер слова w 16, 32 или 64 бита, шифрование происходит блоками по 4 слова;
- количество раундов r ;
- размер секретного ключа l в байтах.

Таким образом конкретный тип реализации алгоритма обозначается по схеме $RC6 - w/r/l$ например в конкурсе AES участвовал алгоритм $RC6 - 32/20/16$.

Введем следующие обозначения операций, которые используются в алгоритме:

- $+$, $-$ сложение и вычитание по модулю 2^w ;
- \leftarrow операция присваивания;
- $*$ умножение по модулю 2^w ;
- \oplus побитовое сложение по модулю 2
- \lll , \ggg циклический сдвиг влево или вправо на указанное число бит.

Процесс шифрования

Опишем процесс шифрования с помощью следующего псевдокода
Вход: блок из четырех слов (a, b, c, d) , раундовый ключ W .

Выход: Зашифрованный блок (a, b, c, d) .

$b \leftarrow b + W_0$; $d \leftarrow d + W_1$;

FOR $i = 1, \dots, r$ DO

{
 $t \leftarrow (b * (2b + 1)) \lll \log_2 w$;
 $u \leftarrow (d * (2d + 1)) \lll \log_2 w$;
 $a \leftarrow ((a \oplus t) \lll u) + W_{2i}$;
 $c \leftarrow ((c \oplus u) \lll t) + W_{2i+1}$;
 $(a, b, c, d) \leftarrow (b, c, d, a)$;
}

$a \leftarrow a + W_{2r+2}$; $c \leftarrow c + W_{2r+3}$;

Return (a, b, c, d) .

Процесс шифрования завершен.

Процесс дешифрования.

Процесс дешифрования аналогичен процессу шифрования, следовательно его также будет удобно представить в виде псевдокода.

Вход: блок из четырех слов (a, b, c, d) , раундовый ключ W .

Выход: Дешифрованный блок (a, b, c, d) .

```

 $c \leftarrow c - W_{2r+3}; \quad a \leftarrow a - W_{2r+2};$ 
FOR  $i = r, \dots, 1$  DO
{
     $(a, b, c, d) \leftarrow (d, a, b, c);$ 
     $t \leftarrow (b * (2b + 1)) \lll \log_2 w;$ 
     $u \leftarrow (d * (2d + 1)) \lll \log_2 w;$ 
     $a \leftarrow ((a - W_{2i}) \ggg u) \oplus t;$ 
     $c \leftarrow ((c - W_{2i+1}) \ggg t) \oplus u;$ 
}
 $d \leftarrow d - W_1; \quad b \leftarrow b - W_0;$ 
Return  $(a, b, c, d)$ .

```

Процесс дешифрования завершен.

Легко заметить, что процесс шифрования и дешифрования выполняется с помощью одного и того же раундового ключа длиной $2r + 4$ слова. Рассмотрим процедуру формирования раундового ключа более подробно.

Формирование раундового ключа.

Формирование ключа выполняется в два этапа:

- Инициализация массива ключей W_0, \dots, W_{2r+3} производится следующим образом:

$$W_0 \leftarrow P_w;$$

$$W_{i+1} \leftarrow W_i + Q_w,$$

где P_w и Q_w - псевдослучайные константы, а именно первые w бит двоичного разложения чисел w и $\phi - 1$ соответственно (где e - число Эйлера, а $\phi = \frac{\sqrt{5} - 1}{2}$).

Можно построить таблицу зависимости псевдослучайных констант P_w и Q_w от размера слова w .

w	16	32	64
P_w	{b7e1}	{b7e15163}	{b7e151628aed2a6b}
Q_w	{9e37}	{9e3779b9}	{9e3779b97f4a7c15}

- Циклически выполняются следующие действия:

$$W_i \leftarrow (W_i + a + b) \lll 3;$$

$$a \leftarrow W_i;$$

$$K_j \leftarrow (K_j + a + b) \lll (a + b);$$

$$b \leftarrow K_j;$$

$$i \leftarrow i+1 \bmod 2r+4;$$

$$j \leftarrow j+1 \bmod c,$$

где i, j, a, b - временные переменные, их начальные значения равны нулю.

Процесс формирования раундового ключа завершен.

Управление ключами

Для успешного использования симметричных криптосистем пользователям необходимо как-то договориться о секретном ключе, т.е. найти путь управления ключами. При описании ключей можно говорить, что существует два типа ключей: статический и сеансовый.

Статичный ключ. Так называют ключ, который используется в течение большого периода времени. Раскрытие статичного ключа обычно считается главной проблемой с потенциально катастрофическими последствиями.

Сеансовый (кратковременный) ключ применяется лишь малое время, от нескольких секунд до одного дня. Его обычно берут на вооружение для обеспечения конфиденциальности в одном сеансе связи. Раскрытие сеансового ключа может повлечь за собой лишь нарушение секретности сеанса и ни коим образом не должно влиять на криптостойкость всей системы.

Распределение ключей между пользователями также может происходить различными путями: физическое распространение, распространение основанное на симметричных криптосистемах и распространение основанное на ассиметричных криптосистемах.

Управление ключами - это процесс, состоящий из следующих функций:

- ☐ генерация ключей;
- ☐ хранение ключей;
- ☐ распределение ключей;
- ☐ удаление ключей.

Можно сделать следующие утверждение: эффективность криптографической защиты определяются стойкостью используемых алгоритмов и надежностью протоколов⁴ управления ключами.

Протоколы генерации ключей.

Рассмотрим один из методов генерации сеансового ключа использующий алгоритм DES.

Обозначения:

$E_k(X)$ – результат шифрования алгоритмом DES значения X ;

K – ключ, зарезервированный для генерации секретных ключей (статический ключ);

V_0 – секретное 64-битовое начальное число;

T – временная метка

Случайный сеансовый ключ R_i генерируют, вычисляя значение

$$R_i = E_k(E_k(T) \oplus V_i).$$

⁴ Протокол — это последовательность шагов, которые предпринимают две или большее количество сторон для совместного решения некоторой задачи. Следует обратить внимание на то, что все шаги предпринимаются в порядке строгой очередности и ни один из них не может быть сделан прежде, чем закончится предыдущий

Следующее значение V_{i+1} вычисляется так:

$$V_{i+1} = E_k(E_k(T) \oplus R_i)$$

Если необходим 128-битовый ключ, генерируют пару ключей R_i, R_{i+1} и объединяют их вместе.

Случайный ключ R_i следует регулярно менять, не выполнение этого требования может привести к его раскрытию и утечки информации.

Распределение ключей

Отметим, что можно говорить о существовании двух способов распределения ключей между пользователями:

- протоколы в которых стороны выполняют передачу ключей при непосредственном взаимодействии, то есть двусторонние протоколы (протоколы типа «точка-точка»);
- протоколы с централизованным распределением ключей (протоколы с доверенным центром).

Протокол типа «точка-точка» основанный на симметричной криптосистеме

Предположим, что пользователи A и B обладают общей секретной информацией (секретным ключом k_{AB}). Тогда для передачи сеансового ключа можно выполнить одностороннюю передачу заданную следующей символьной записью:

$$A \rightarrow B: \quad E_{k_{AB}}(k, T, b),$$

где $E_{k_{AB}}$ - алгоритм шифрования с ключом k_{AB} , k - сеансовый ключ, T – временная метка⁵, b -идентификатор пользователя B . Зная секретный ключ k_{AB} пользователь B легко может найти ключ k .

Передача временной метки и идентификатора пользователя выполняются по следующим причинам:

- передача временной метки позволяет надеяться, что злоумышленник не сможет осуществить повторную передачу того же сообщения пользователю A ;
- передача же идентификатора получателя позволяет надеяться, что злоумышленник не сможет вернуть отправителю перехваченное сообщение.

Если дополнительно требуется аутентификация сеанса, то можно использовать протокол, состоящий из следующих действий:

Символьная запись	Пояснения
1. $B \rightarrow A: \quad r_B$	r_B - случайное число, сгенерированное пользо-

⁵ При использовании временной метки предполагается, что участники пытаются соблюдать синхронизацию часов.

	вателем B и отправленное пользователю A
2. $A \rightarrow B: E_{k_{AB}}(k, r_B, T, b)$	

Приведем теперь протокол Шамира, который позволяет пользователям A и B безопасно обмениваться информацией P без использования какой-либо общей секретной информации. Он предполагает использование коммутативного симметричного шифра, для которого:

$$E_{k_A}(E_{k_B}(P)) = E_{k_B}(E_{k_A}(P)),$$

где E -шифрующее преобразование, k_A - секретный ключ пользователя A , а k_B - секретный ключ пользователя B . Тогда трехшаговый протокол Шамира для передачи ключа k от A к B может быть реализован следующим образом:

Символьная запись	Пояснения
1. $A \rightarrow B: E_{k_A}(k)$	Пользователь A шифрует ключ k и передает результат пользователю B
2. $B \rightarrow A: E_{k_B}(E_{k_A}(k))$	Пользователь B шифрует полученное сообщение и передает результат пользователю A
3. $A \rightarrow B: D_{k_A}(E_{k_B}(E_{k_A}(k)))$, где D -дешифрующее преобразование	Пользователь A дешифрует полученное сообщение и передает результат пользователю B . В результате у пользователя B остается сообщение $E_{k_B}(k)$, которое он легко может дешифровать.

Заметим, что в этом протоколе можно использовать не каждое коммутативное преобразование E . Например, легко заметить, что для преобразования $E_{k_P}(P) = P \oplus k_P$ протокол оказывается заведомо нестойким. В этом случае протокол для передачи секретного ключа k от A к B будет выглядеть:

$$A \rightarrow B: k \oplus k_A,$$

$$B \rightarrow A: k \oplus k_A \oplus k_B,$$

$$A \rightarrow B: k \oplus k_A \oplus k_B \oplus k_A = k \oplus k_B,$$

и перехватив все три сообщения, злоумышленник сможет восстановить секретный ключ k , действительно, $(k \oplus k_A) \oplus (k \oplus k_A \oplus k_B) \oplus (k \oplus k_B) = k$.

Протоколы с централизованным распределением ключей

В рассматриваемом разделе будем предполагать, что в обмене закрытой информацией участвуют двое: пользователи A и B . Кроме того, предполагаем, что они прибегают к услугам доверенного лица S . Пользователи A и S обладают общей секретной информацией (секретным ключом k_{AS}), соответственно пользователи B и S обладают секретным ключом k_{BS} .

Протокол Нидхейма-Шредера. Для выработки сеансового ключа k_{AB} нужно выполнить следующие действия:

Символьная запись	Пояснения
-------------------	-----------

1. $A \rightarrow S: n_a$	Пользователь A создает уникальную числовую вставку n_a и передает ее S
2. $S \rightarrow A: E_{k_{AS}}(n_a, b, k_{AB}, E_{k_{BS}}(k_{AB}, a))$ где E -симметричное шифрующее преобразование, b - идентификатор пользователя B , a - идентификатор пользователя A .	Доверенное лицо S генерирует ключ k_{AB} и посылает его пользователю A зашифрованным письмом. В письмо включается числовая вставка n_a , по которой пользователь A узнает, что полученное сообщение было послано в ответ на его запрос.
3. $A \rightarrow B: E_{k_{BS}}(k_{AB}, a);$	Сеансового ключа k_{AB} пересылаются пользователю B
4. $B \rightarrow A: E_{k_{AB}}(n_b);$	Пользователь B создает уникальную числовую вставку n_b и передает ее A зашифрованным письмом. Пользователь B хочет удостовериться в пользователе A .
5. $A \rightarrow B: E_{k_{AB}}(n_b - 1).$	Пользователь A убеждает в своей дееспособности.

Основным недостатком протокола Нидхейма-Шредера является отсутствие временных меток. Следовательно, злоумышленник, найдя сообщения и ключи предыдущих сеансов, может попытаться использовать их вместо последних трех действий протокола Нидхейма-Шредера. В результате сможет обмануть пользователя B выдав себя за A .

Приведем теперь протокол Цербера, в котором устранены недостатки присущие протоколу Нидхейма-Шредера. Для выработки сеансового ключа k_{AB} нужно выполнить следующие действия:

Символьная запись	Пояснения
1. $A \rightarrow S: A, B$	Пользователь A сообщает S , что хотел бы связаться с B
2. $S \rightarrow A: E_{k_{AS}}(t_s, l, k_{AB}, b, E_{k_{BS}}(t_s, l, k_{AB}, a))$, где b - идентификатор пользователя B , a - идентификатор пользователя A , t_s - временная метка, l - время жизни ключа	Доверенное лицо S создает сообщение $E_{k_{BS}}(t_s, l, k_{AB}, a)$ и передает его A для передачи B . Пользователь A получает копию ключа k_{AB} в форме, которую он может прочесть.
3. $A \rightarrow B: E_{k_{BS}}(t_s, l, k_{AB}, a), E_{k_{AB}}(a, t_a)$	Пользователь B , получив $E_{k_{BS}}(t_s, l, k_{AB}, a)$, легко может найти ключ k_{AB} . Клиент A , желая проверить возможность общения с B , посылает зашифрованную временную

	метку t_a
$B \rightarrow A: E_{k_{AB}}(t_a + 1)$	Проверив, что временная метка t_a является свежей, пользователь B отправляет временную метку $t_a + 1$, показывая, что готов к сеансу.

Открытое распределение ключей

Открытое распределение ключей позволяет двум пользователям вырабатывать общий секретный ключ путем динамического взаимодействия на основе обмена открытыми сообщениями без какой-либо общей секретной информации, распределенной заранее.

протокол DIFFIE-HELLMAN

Предположим, что есть два пользователя A и B для того, чтобы сгенерировать ключ надо выполнить следующие 5 действий:

1. Пользователи A и B должны выбрать большое простое число n и g , где g должно быть образующим элементом мультипликативной группы $Z_n^* = \{1, 2, \dots, n-1\}$. Пользователи A и B могут открыто распространять информацию о n и g ;
2. Пользователь A выбирает случайное большое целое число x и отправляет пользователю B величину $X = g^x \bmod n$;
3. Пользователь B выбирает случайное большое целое число y и отправляет пользователю A величину $Y = g^y \bmod n$;
4. Пользователь A вычисляет величину $k = Y^x \bmod n$;
5. Пользователь B вычисляет величину $\tilde{k} = X^y \bmod n$.

В результате у пользователей A и B появился один общий ключ $\tilde{k} = X^y \bmod n = g^{xy} \bmod n = Y^x \bmod n = g^{xy} \bmod n = k$

Однако следует отметить, что указанный протокол является уязвимым для атаки, называемой «человек в середине». Злоумышленник C может перехватить открытое значение, посылаемое от A к B , и послать вместо него своё открытое значение. Затем он может перехватить открытое значение, посылаемое от B к A , и также послать вместо него своё открытое значение. Тем самым пользователь C получит общие секретные ключи и с A и B .

протокол МТИ

Предположим, что есть два пользователя A и B для того, чтобы сгенерировать ключ надо выполнить следующие 6 действий:

1. Пользователи A и B должны выбрать большое простое число n и g , где g должно быть образующим элементом мультипликативной группы

$Z_n^* = \{1, 2, \dots, n-1\}$. Пользователи A и B могут открыто распространять информацию о n и g ;

2. Пользователи A и B должны сгенерировать секретные ключи a , $1 \leq a \leq n-2$, и b , $1 \leq b \leq n-2$, соответственно, и публикуют свои открытые ключи $z_A = g^a \bmod n$ и $z_B = g^b \bmod n$;

3. Пользователь A выбирает случайное целое число x , $1 \leq x \leq n-2$ и отправляет пользователю B величину $X = g^x \bmod n$;

4. Пользователь B выбирает случайное большое целое число y , $1 \leq y \leq n-2$ и отправляет пользователю A величину $Y = g^y \bmod n$;

5. Пользователь A вычисляет величину $k = Y^a z_B^x \bmod n$;

6. Пользователь B вычисляет величину $\tilde{k} = X^b z_A^y \bmod n$.

В результате у пользователей A и B появился один общий ключ

$$\tilde{k} = (g^x)^b (g^a)^y = (g^y)^a (g^b)^x = k = g^{xb+ya} \bmod n.$$

Пример реализуем протокол МТИ.

1. Пусть $n = 13$, $g = 2$. Легко убедиться, что любой элемент группы

$$Z_{13}^* = \{1, 2, \dots, 12\}$$
 является степенью числа $g = 2$;

2. Пользователь A генерирует число $a = 5$ и публикует $z_A = 2^5 \bmod 13 = 6$, соответственно пользователь B генерирует число $b = 3$ и публикует $z_B = 2^3 \bmod 13 = 8$;

3. Пользователь A генерирует число $x = 2$ и отправляет пользователю B величину $X = 2^2 \bmod 13 = 4$;

4. Пользователь B генерирует число $y = 4$ и отправляет пользователю A величину $Y = 2^4 \bmod 13 = 3$;

5. Пользователь A на настоящий момент знает величины: $n, g, a, z_A, z_B, x, X, Y$. Пользователь A вычисляет величину $k = (Y^a z_B^x) \bmod n = (3^5 8^2) \bmod 13 = (9 * 12) \bmod 13 = 4$

6. Пользователь B на настоящий момент знает величины: $n, g, b, z_A, z_B, y, X, Y$. Пользователь B вычисляет величину $\tilde{k} = (X^b z_A^y) \bmod n = (4^3 6^4) \bmod 13 = 12 * 9 \bmod 13 = 4$ ■

Схемы разделения секрета.

Будем говорить, что t участников A_i $i = 1, \dots, n$ (законных пользователей) t -хранят секрет s , $1 \leq t \leq n$, если выполняются следующие три условия:

- каждый A_i знает некоторую информацию (частичный секрет) a_i , неизвестное любому другому участнику;
- секрет s может быть легко вычислен на основе любых t частичных секретов a_{i_1}, \dots, a_{i_t} ;

3. знание любых $t - 1$ частичных секретов a_i не дает такой возможности.

Схема разделения секрета включает два протокола:

- протокол формирования частичных секретов и распределения их между пользователями;
- протокол восстановления секрета группой пользователей.

В качестве примера рассмотрим пороговую схему Шамира. Для построения пороговой схемы (n, t) Шамир воспользовался многочленами вида $f(x) = b_{t-1}x^{t-1} + b_{t-2}x^{t-2} + \dots + b_1x + b_0$ в конечном поле. Секретным считается свободный член b_0 . В качестве частных секретов выступают значение многочлена $f(x)$ в некоторых точках. Заметим, что любые t участников, воспользовавшись интерполяционной формулой Лагранжа, могут найти многочлен $f(x)$.

Пример. Построим пороговую схему $(3, 5)$. В качестве конечного поля возьмем Z_{13} , а в качестве многочлена $f(x) = (7x^2 + 8x + 11) \bmod 13$.

- протокол формирования частичных секретов состоит в вычислении $f(x)$

$$a_1 = f(1) = 7 + 8 + 11 = 0 \pmod{13};$$

$$a_2 = f(2) = 28 + 16 + 11 = 3 \pmod{13};$$

$$a_3 = f(3) = 63 + 24 + 11 = 7 \pmod{13};$$

$$a_4 = f(4) = 112 + 32 + 11 = 12 \pmod{13};$$

$$a_5 = f(5) = 175 + 40 + 11 = 5 \pmod{13}.$$

- протокол восстановления секрета группой пользователей.

Чтобы восстановить $f(x)$ из трех частных секретов a_2, a_3, a_5 решается система линейных уравнений:

$$\begin{cases} f(2) = 4b_2 + 2b_1 + b_0 = 3 \pmod{13} \\ f(3) = 9b_2 + 3b_1 + b_0 = 7 \pmod{13} \\ f(5) = 25b_2 + 5b_1 + b_0 = 5 \pmod{13} \end{cases}$$

Решением будут $b_2 = 7, b_1 = 8, b_0 = 11$. ■

Криптоанализ

Криптоанализ занимаются задачами, обратными по отношению к задачам криптографии. Он ставит своей задачей в разных условиях получить дополнительные сведения о ключе шифрования, чтобы значительно уменьшить диапазон вероятных ключей. Взлом шифра совсем не обязательно подразумевает обнаружение способа, примененного на практике для восстановления открытого текста по перехваченному зашифрованному сообщению. Шифр считается взломанным, если в системе обнаружено слабое место, которое может быть использовано для более эффективного взлома, чем метод полного перебора ключей.

Перед тем как продолжить разговор о криптоанализе, естественно принять следующее соглашение: «Злоумышленник знает используемую криптосистему».

Сформулируем основные типы симметричного блочного криптоанализа.

Типы криптоанализа.	Данные, известные криптоаналитику
Анализ только шифрованного текста	<ul style="list-style-type: none">• Алгоритм шифрования.• Подлежащий расшифровке шифрованный текст.
Анализ с известным открытым текстом	<ul style="list-style-type: none">• Алгоритм шифрования.• Подлежащий расшифровке шифрованный текст.• Одну или несколько пар $(pt, E_k(pt))$, где pt -открытый текст, $E_k(pt)$ -шифрованный текст, естественно для всех пар ключ шифрования одинаков.
Анализ с избранным открытым шифром	<ul style="list-style-type: none">• Алгоритм шифрования.• Подлежащий расшифровке шифрованный текст.• Выбранный криптоаналитиком открытый текст и соответствующий ему шифротекст.

Легко заметить, что следует различать методы криптоанализа и типы криптоанализа. Например, к типу анализа только шифрованного текста можно отнести метод полного перебора и метод статистического криптоанализа. Лишь относительно слабые алгоритмы могут быть взломаны при анализе только шифрованного текста. Отметим, что задача криптоанализа является достаточно сложной.

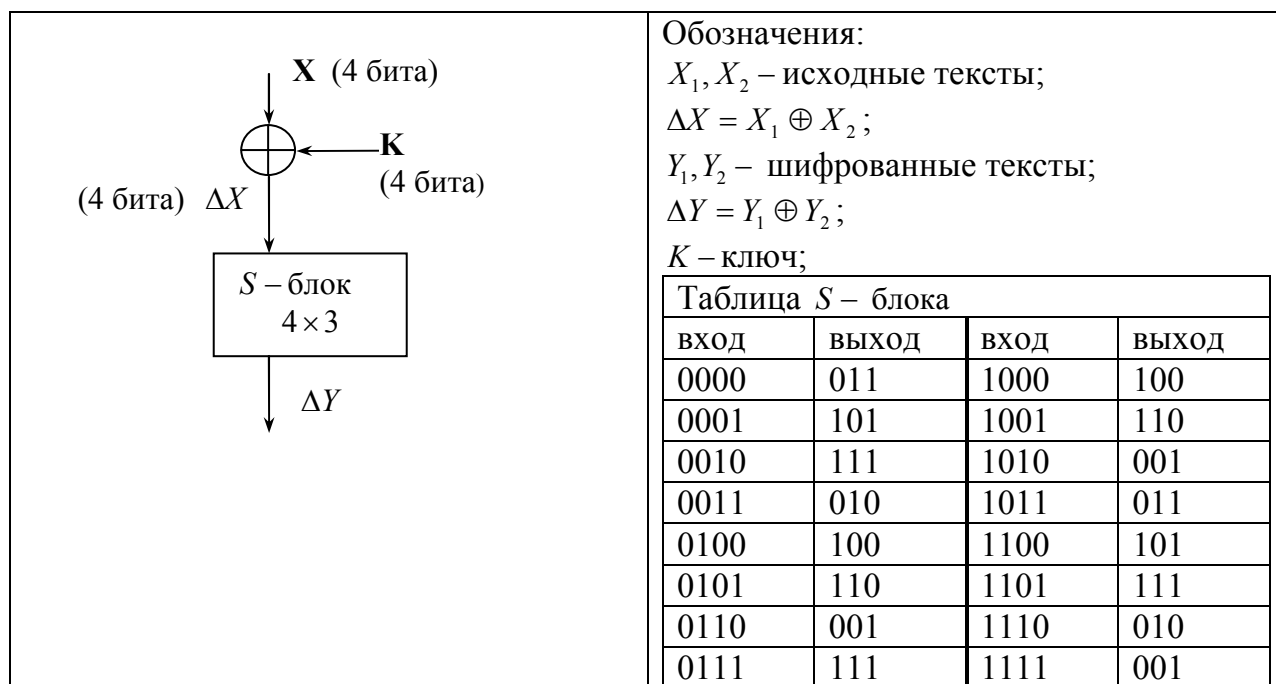
Дифференциальный криптоанализ.

Дифференциальный криптоанализ – метод криптоанализа симметричных блочных шифров, был предложен в 1990 году Эли Бихамом и Ади Шамиром. Дифференциальный криптоанализ работает с парами шифротекстов, открытые тексты которых содержат определенные отличия. Идея заключается в анализе процесса изменения несходства⁶ для пары открытых тестов, в процессе прохождения через

⁶ Для DES например, термин «несходства» определяется с помощью операции **исключающего или**. Для других алгоритмов этот термин может определяться другим образом.

циклы шифрования с одним и тем же ключом. Проиллюстрируем идею на нескольких простых примерах:

1. Предположим, что шифр состоит из двух операций (исключающее или и подстановки S), как показано на рисунке.



Доказано, что для любого заданного ΔX не все значения ΔY равновероятны, следовательно возможно установить вероятностные отношения (построить таблицу зависимости ΔY от ΔX).

Строится таблица следующим образом. Диапазон изменения ΔX лежит в пределах 0001-1111. Каждое из значений ΔX может быть получено шестнадцатью возможными комбинациями векторов X_1 и X_2 . Например,

$\Delta X = 0001$ может быть получено

$\Delta X = X_1 \oplus X_2$	$\Delta X = X_1 \oplus X_2$	$\Delta X = X_1 \oplus X_2$	$\Delta X = X_1 \oplus X_2$
$0000 \oplus 0001$	$0010 \oplus 0011$	$0100 \oplus 0101$	$0110 \oplus 0111$
$0001 \oplus 0000$	$0011 \oplus 0010$	$0101 \oplus 0100$	$0111 \oplus 0110$
$1000 \oplus 1001$	$1010 \oplus 1011$	$1100 \oplus 1101$	$1110 \oplus 1111$
$1001 \oplus 1000$	$1011 \oplus 1010$	$1101 \oplus 1100$	$1111 \oplus 1110$

При этом при прохождении разных пар X_1, X_2 через S – блок могут получиться различные значения ΔY .

Зависимость ΔY от ΔX .								
ΔY	000	001	010	011	100	101	110	111
ΔX								
0001	0	0	8	2	0	2	4	0
0010	0	2	0	0	2	6	2	4
0011	0	2	2	2	2	2	0	6
0100	0	4	2	4	0	2	2	2
0101	4	4	0	6	0	2	0	0
0110	0	0	4	2	6	0	2	2
0111	0	2	0	0	6	2	6	0
1000	0	6	0	4	0	0	4	2
1001	2	2	0	6	2	4	0	0

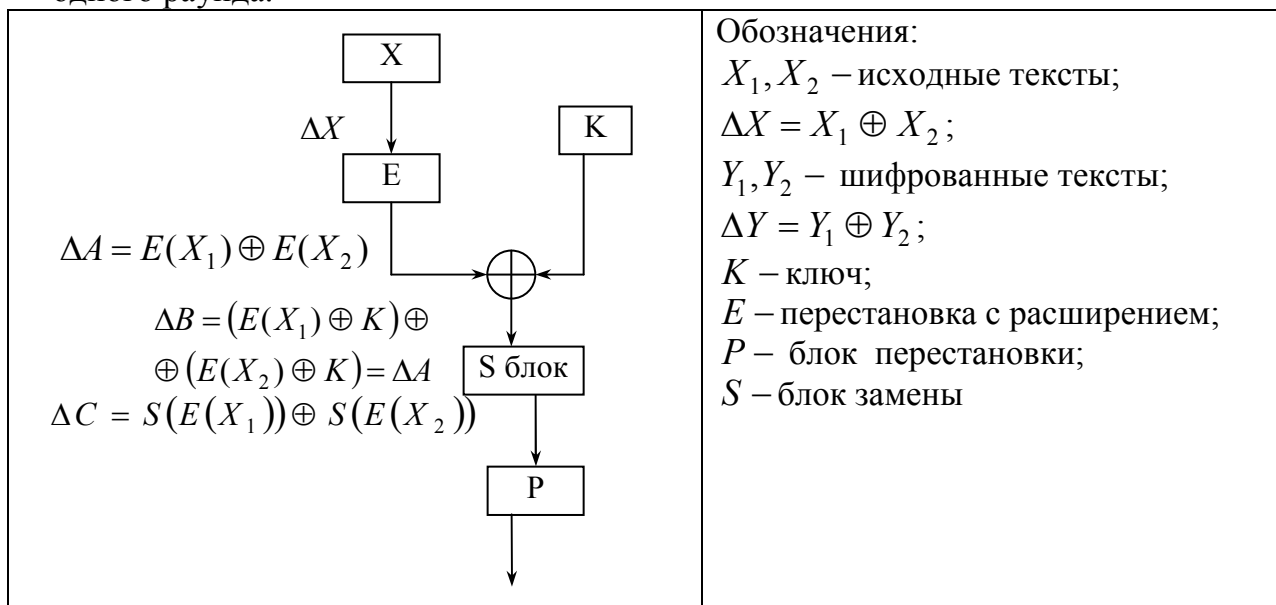
1010	2	0	2	2	4	0	4	2
1011	2	2	2	0	4	2	4	0
1100	6	0	2	2	2	2	2	0
1101	4	0	6	0	2	0	4	0
1110	0	4	2	0	2	8	0	0
1111	2	0	2	2	0	0	0	10

Из таблицы легко заметить, что оптимальной парой $(\Delta X, \Delta Y)$ является пара (1111,111). Теперь попытаемся найти ключ K . Для этого возьмем несколько пар открытых текстов (X_1, X_2) , таких, что $\Delta X = 1111$. Пусть будут пары (0000,1111) и (0100,1011).

- Пара $(X_1, X_2) = (0000, 1111)$. Предположим, что мы получили пару $(Y_1, Y_2) = (001, 110)$. Перейдем к анализу:
 - $Y_1 = 001 \Rightarrow X_1 \oplus K = 0110$ или $X_1 \oplus K = 1010$, или $X_1 \oplus K = 1111$. Следовательно, $K = 0110$ или $K = 1010$, или $K = 1111$.
 - $Y_2 = 110 \Rightarrow X_2 \oplus K = 0101$ или $X_2 \oplus K = 1001$. Следовательно, $K = 1010$ или $K = 0110$.
- Пара $(X_1, X_2) = (0100, 1011)$. Предположим, что мы получили пару $(Y_1, Y_2) = (010, 101)$. Перейдем к анализу:
 - $Y_1 = 010 \Rightarrow X_1 \oplus K = 0011$ или $X_1 \oplus K = 1110$. Следовательно, $K = 0111$ или $K = 1010$.
 - $Y_2 = 101 \Rightarrow X_2 \oplus K = 0001$ или $X_2 \oplus K = 1100$. Следовательно, $K = 1010$ или $K = 0111$.

Вывод: Один ключ, а именно $K = 1010$, встречается чаще остальных. Таким образом, можно предположить, что он и есть ключ шифрования. ■

2. Предположим, что рассматриваемый симметричный блочный шифр состоит из одного раунда.

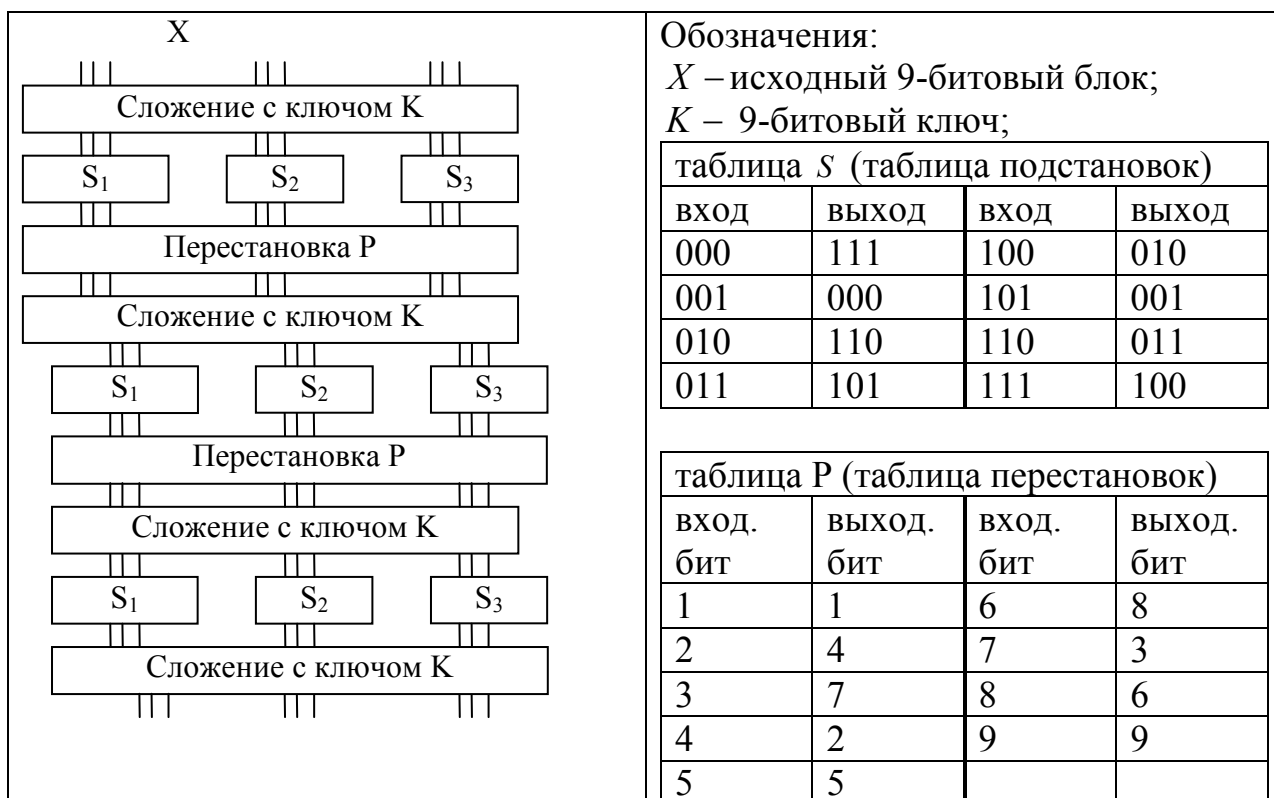


Криптоаналитик задает пару (X_1, X_2) с несходством ΔX . Выходы (Y_1, Y_2) ему известны, следовательно, известна ΔY . Взломщик знает перестановку с расширением E и P – блок, а следовательно и ΔA и ΔC . Значения на выходе сумматора по модулю 2 неизвестны, однако известно ΔB . Заметим, что для любого

заданного ΔA не все значения ΔC равновероятны. Комбинация ΔC и ΔA позволяет предположить значение битов для $E(X_1) \oplus K$ и $E(X_2) \oplus K$. Напомним, что $E(X_1)$ и $E(X_2)$ известны, следовательно можем найти информацию о K . ■

Отметим, что способ вскрытия циклового ключа по набору выходов и разности входов для каждого шифра индивидуален, а дифференциальный криптоанализ по существу является методом, который может быть адаптирован к конкретному шифру.

Пример. Предположим, что последовательность шифрующих операций такова: два раза выполняется цикл (сложение, подстановка, перестановка), затем сложение, подстановка, сложение; как показано на рисунке.



Проанализируем S – блок замены и построим таблицу зависимости выходной разности блок замены ΔC от входной разности того же блока ΔA . В результате получим таблицу

таблица M (зависимость ΔC от ΔA).								
ΔC	000	001	010	011	100	101	110	111
ΔA								
000	8	0	0	0	0	0	0	0
001	0	0	0	4	0	0	0	4
010	0	4	0	0	0	4	0	0
011	0	0	4	0	0	0	4	0
100	0	4	0	0	0	4	0	0
101	0	0	4	0	0	0	4	0
110	0	0	0	0	8	0	0	0
111	0	0	0	4	0	0	0	4

Для одного блока перестановки в одном цикле легко можно найти оптимальную пары $(\Delta A, \Delta C) = (110, 100)$ и $(\Delta A, \Delta C) = (000, 000)$. Возьмем $\Delta A_1 = 110000000$, $\Delta A_2 = 000000001$, $\Delta A_3 = 111001000$.⁷

Используя таблицу М, рассмотрим, что будет происходить с предложенными разностями при прохождении их по раундам алгоритма

рассматриваемая разность	$\Delta A_1 = 110000000$	$\Delta A_2 = 000000001$		$\Delta A_3 = 111001000$
вход в блоки замены 1 раунда	110000000	000000001		111001000
возможные выходы из блоков замены 1 раунда	100000000	000000011	000000111	011011000, 011111000, 111011000, 111111000
возможные входы в блоки замены 2 раунда	100000000	000001001	001001001	zz0110110
возможные выходы из блоков замены 2 раунда	001000000, 101000000	000011011, 000011111, 000111011, 000111111	011011011, 011011111, 011111011, 011111111, 111011011, 111011111, 111111011, 111111111	000100100 : zz=00 001100100 : zz=01 101100100 : zz=01 001100100 : zz=10 101100100 : zz=01 100100100 : zz=11
возможные входы в блоки замены 3 раунда	000000100, 100000100 На вход блока замены третьего раунда поступит разность z00000100, где z неизвестный символ.	000011011, 001011011, 010011011, 011011011 На вход блока замены третьего раунда поступит разность zzzz11z11, где z неизвестный символ.	000111111, 001111111, 010111111, 011111111, 100111111, 101111111, 110111111, 111111111 На вход блока замены третьего раунда поступит разность z11000z00, где z неизвестный символ.	011000000 011000000 111000100 011000000 111000100 111000000 На вход блока замены третьего раунда поступит разность z11000z00, где z неизвестный символ.

Предположим, что нам известны некоторые результаты шифрования

$\Delta A_1 = 110000000$			$\Delta A_2 = 000000001$			$\Delta A_3 = 110110110$		
N_0	X_1	Y_1	N_0	X_1	Y_1	N_0	X_1	Y_1
1	001000000	101010101	1	000000000	000010010	1	000000000	000010010
2	010000000	101010100	2	111111111	000000000	2	010000000	101010100
N_0	X_2	Y_2	N_0	X_2	Y_2	N_0	X_2	Y_2

⁷ Можно было взять $\Delta A_3 = 111001000$, но тогда на вход блока замены третьего раунда поступит разность z00100100, где z неизвестный символ.

1	111000000	000010000	1	000000001	011101101	1	111001000	010010011
2	100000000	000010001	2	111111110	011011011	2	101001000	010010100

Рассмотрим предложенные значения входной разности

➤ $\Delta A_1 = 110000000$

– Проанализируем пару текстов $(X_1, X_2) = (001000000, 111000000)$ и соответствующую пару шифров $(Y_1, Y_2) = (101010101, 000010000)$. Легко заметить, что $\Delta C = Y_1 \oplus Y_2 = 101000101$. Ненулевое значение разностей будет только на выходе блоков S_{31} и S_{33} . Входная разность на блоках S_{31} и S_{33} равна 100 (если бы входная разность на блоках была 000, то на выходе из блоков тоже бы получили нулевые значения).

Разность равную 100 можно получить следующими способами

Возможные входы для разности 100				Соответствующие выходы			
1	$000 \oplus 100$	5	$100 \oplus 000$	1	$111 \oplus 010 = 101$	5	$010 \oplus 111 = 101$
2	$001 \oplus 101$	6	$101 \oplus 001$	2	$000 \oplus 001 = 001$	6	$001 \oplus 000 = 001$
3	$010 \oplus 110$	7	$110 \oplus 010$	3	$110 \oplus 011 = 101$	7	$011 \oplus 110 = 101$
4	$011 \oplus 111$	8	$111 \oplus 011$	4	$101 \oplus 100 = 001$	8	$100 \oplus 101 = 001$

На выходе блоков находится значение разности 101, оно могло быть получено, если использовались пары под номерами 1,3,5,7. Легко заметить, что для выходов блоков S_{31} , S_{33} с учетом шифров могут выполняться следующие равенства:

$111 \oplus K_i = 000$	$111 \oplus K_i = 101$	$110 \oplus K_i = 000$	$110 \oplus K_i = 101$
$010 \oplus K_i = 101$	$010 \oplus K_i = 000$	$011 \oplus K_i = 101$	$011 \oplus K_i = 000$

Подключ K_1 и K_3 могут принимать одно из следующих значений: 111, 110, 011, 010.

– Проанализируем пару текстов $(X_1, X_2) = (010000000, 100000000)$ и соответствующую пару шифров $(Y_1, Y_2) = (101010100, 000010001)$,

$\Delta C = 101000101$. Ненулевое значение разностей будет на выходах блоков S_{31} и S_{33} . На вход этих блоков подается входная разность, равная 100. Аналогично предыдущим рассуждениям можно утверждать, что подключ K_1 может принимать одно из следующих значений: 111, 110, 011, 010. Напомним, что выход блока S_{33} складывается с подключом K_3 , в результате получается известный шифртекст и следующие равенства:

$111 \oplus K_3 = 100$	$111 \oplus K_3 = 001$	$110 \oplus K_3 = 100$	$110 \oplus K_3 = 001$
$010 \oplus K_3 = 001$	$010 \oplus K_3 = 100$	$011 \oplus K_3 = 001$	$011 \oplus K_3 = 100$

Подключ K_3 может принимать одно из следующих значений: 011, 110, 111, 010.

➤ $\Delta A_2 = 000000001$

– Проанализируем пару текстов $(X_1, X_2) = (000000000, 000000001)$ и соответствующую пару шифров $(Y_1, Y_2) = (000010010, 011101101)$.

$\Delta C = 01111111$. Рассмотрим блоки S_{32} и S_{33} (на выходе и входе блоков S_{32} и S_{33} находится разность 111).

Разность равную 111 можно получить следующими способами:

Возможные входы для разности 111				Соответствующие выходы			
1	$000 \oplus 111$	5	$111 \oplus 000$	1	$111 \oplus 100 = 011$	5	$100 \oplus 111 = 011$
2	$010 \oplus 101$	6	$101 \oplus 010$	2	$110 \oplus 001 = 111$	6	$001 \oplus 110 = 111$
3	$001 \oplus 110$	7	$110 \oplus 001$	3	$000 \oplus 011 = 011$	7	$011 \oplus 000 = 011$
4	$011 \oplus 100$	8	$100 \oplus 011$	4	$101 \oplus 010 = 111$	8	$010 \oplus 101 = 111$

На выходе блоков находится значение разности 111, оно могло быть получено, если использовались пары под номерами 2,4,6,8. Легко заметить, что для выходов блока S_{32} и S_{33} с учетом шифров могут выполняться следующие равенства:

$110 \oplus K_i = 010$	$110 \oplus K_i = 101$	$101 \oplus K_i = 010$	$101 \oplus K_i = 101$
$001 \oplus K_i = 101$	$001 \oplus K_i = 010$	$010 \oplus K_i = 101$	$010 \oplus K_i = 010$

Подключи K_2 и K_3 могут принимать одно из следующих значений: 100, 011, 111, 010

Ранее нами были определены еще четыре возможных варианта подключения K_3 : 011, 110, 111, 010. Как видно, совпадают только три возможных подключения, а именно: 011, 111, 010, а значит один из них и является истинным.

– Проанализируем пару текстов $(X_1, X_2) = (11111111, 11111110)$ и соответствующую пару шифров $(Y_1, Y_2) = (00000000, 01101101)$. Легко заметить, что $\Delta C = 01101101$. Рассмотрим блоки S_{32} и S_{33} (на выходе блоков S_{32} и S_{33} находится разность 011, а на входе разность 111).

Легко заметить, что для выходов блока S_{32} и S_{33} с учетом шифров могут выполняться следующие равенства:

$111 \oplus K_i = 011$	$111 \oplus K_i = 000$	$000 \oplus K_i = 011$	$000 \oplus K_i = 000$
$100 \oplus K_i = 000$	$100 \oplus K_i = 011$	$011 \oplus K_i = 000$	$011 \oplus K_i = 011$

Подключи K_2 и K_3 могут принимать одно из следующих значений: 100, 111, 011, 000.

Ранее нами были определены еще четыре возможных варианта подключения K_2 : 100, 011, 111, 010. Как видно, совпадают только три возможных подключения, а именно: 100, 111, 011, а значит один из них и является истинным. Истинным подключом K_3 является 011 или 111.

➤ $\Delta A_3 = 111001000$

– Проанализируем пару текстов $(X_1, X_2) = (00000000, 111001000)$ и соответствующую пару шифров $(Y_1, Y_2) = (000010010, 010010011)$

$\Delta C = 010000001$. Рассмотрим блок S_{31} (на выходе блока находится разность 010, на входе блока находится разность 011).

Возможные входы для разности 011				Соответствующие выходы			
1	$000 \oplus 011$	5	$011 \oplus 000$	1	$111 \oplus 101 = 010$	5	$101 \oplus 111 = 010$
2	$010 \oplus 001$	6	$001 \oplus 010$	2	$110 \oplus 000 = 110$	6	$000 \oplus 110 = 110$
3	$100 \oplus 111$	7	$111 \oplus 100$	3	$010 \oplus 100 = 110$	7	$100 \oplus 010 = 110$
4	$101 \oplus 110$	8	$110 \oplus 101$	4	$001 \oplus 011 = 010$	8	$011 \oplus 001 = 010$

Легко заметить, что для выходов блока S_{31} с учетом шифров могут выполняться следующие равенства:

$101 \oplus K_1 = 010$	$101 \oplus K_1 = 000$	$011 \oplus K_1 = 000$	$011 \oplus K_1 = 010$
$111 \oplus K_1 = 000$	$111 \oplus K_1 = 010$	$001 \oplus K_1 = 010$	$001 \oplus K_1 = 000$

Подключ K_1 может принимать одно из следующих значений: 111, 101, 011, 001. Ранее нами были определены еще четыре возможных варианта подключения K_1 : 011, 110, 111, 010. Как видно, совпадают только два возможных подключения, а именно: 011, 111, а значит один из них и является истинным

Рассмотрим блоки S_{33} (на выходе блока находится разность 001, на входе блока находится разность 100).

Возможные входы для разности 100				Соответствующие выходы			
1	$000 \oplus 100$	5	$100 \oplus 000$	1	$111 \oplus 010 = 101$	5	$010 \oplus 111 = 101$
2	$001 \oplus 101$	6	$101 \oplus 001$	2	$000 \oplus 001 = 001$	6	$001 \oplus 000 = 001$
3	$010 \oplus 110$	7	$110 \oplus 010$	3	$110 \oplus 011 = 101$	7	$011 \oplus 110 = 101$
4	$011 \oplus 111$	8	$111 \oplus 011$	4	$101 \oplus 100 = 001$	8	$100 \oplus 101 = 001$

Легко заметить, что для выходов блока S_{33} с учетом шифров могут выполняться следующие равенства:

$000 \oplus K_3 = 010$	$000 \oplus K_3 = 011$	$100 \oplus K_3 = 010$	$100 \oplus K_3 = 011$
$001 \oplus K_3 = 011$	$001 \oplus K_3 = 010$	$101 \oplus K_3 = 011$	$101 \oplus K_3 = 010$

Подключ K_3 может принимать одно из следующих значений: 010, 011, 110, 111.

– Проанализируем пару текстов $(X_1, X_2) = (010000000, 101001000)$ и соответствующую пару шифров $(Y_1, Y_2) = (101010100, 010010100)$

$\Delta C = 111000000$. Рассмотрим блок S_{31} (на выходе блока S_{31} находится разность 111, а на входе блока находится разность 111).

Возможные входы для разности 111				Соответствующие выходы			
1	$000 \oplus 111$	5	$111 \oplus 000$	1	$111 \oplus 100 = 011$	5	$100 \oplus 111 = 011$
2	$010 \oplus 101$	6	$101 \oplus 010$	2	$110 \oplus 001 = 111$	6	$001 \oplus 110 = 111$
3	$001 \oplus 110$	7	$110 \oplus 001$	3	$000 \oplus 011 = 011$	7	$011 \oplus 000 = 011$
4	$011 \oplus 100$	8	$100 \oplus 011$	4	$101 \oplus 010 = 111$	8	$010 \oplus 101 = 111$

Легко заметить, что для K_1 с учетом шифров могут выполняться следующие равенства:

$110 \oplus K_1 = 010$	$110 \oplus K_1 = 101$	$101 \oplus K_1 = 010$	$101 \oplus K_1 = 101$
$001 \oplus K_1 = 101$	$001 \oplus K_1 = 010$	$010 \oplus K_1 = 101$	$010 \oplus K_1 = 010$

Подключ K_1 может принимать одно из следующих значений: 100, 011, 111, 000.

Таким образом, у нас есть 12 возможных значений ключа из всех 512 возможных комбинаций. При последующей их проверки можно убедиться, что ключ $K = 111111111$. ■

Литература.

1. Зензин О.С., Иванов М.А. Стандарт криптографической защиты – AES/Кудиц-Образ.2003
2. С. Панасенко Алгоритмы шифрования. Специальный справочник./БХВ-Петербург, 2009
3. Н. Сمارт Криптография. / Техносфера. Москва. 2005
4. А.В. Бабаш Криптография/ Соломон-Пресс. Москва. 2007

Введение.....	3
Режимы работы блочных шифров.....	5
Несколько блочных симметричных шифров.....	8
Алгоритм DES.....	8
Алгоритм AES.....	14
Алгоритм RC6.....	23
Управление ключами	26
Протоколы генерации ключей.....	26
Распределение ключей	27
Открытое распределение ключей	30
Схемы разделения секрета.....	31
Криптоанализ	33
Дифференциальный криптоанализ.....	33
Литература.....	41