Тема занятия

СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

Представление эксперта

- Опытный преподаватель и эксперт-практик в области информационной безопасности и комплексной защиты информации;
- Автор учебных программ и курсов в вузах Москвы по дисциплинам “Информационная безопасность”, “Аудит информационной безопасности”, “Защита информации ограниченного доступа”, “ИТ-безопасность”;
- Доступ к ГТ (ф. 2);
- Стаж работы руководителем проектов в разработке систем защиты персональных данных более 16 лет;
- Консультант по вопросам информационной безопасности международной ассоциации «Генералы Мира за Мир».

Абзалов
Олег Накибович

Ведущий эксперт-практик
по информационной
безопасности
и комплексной защите
безопасности

Общие понятия. Обобщенная схема симметричных криптосистем

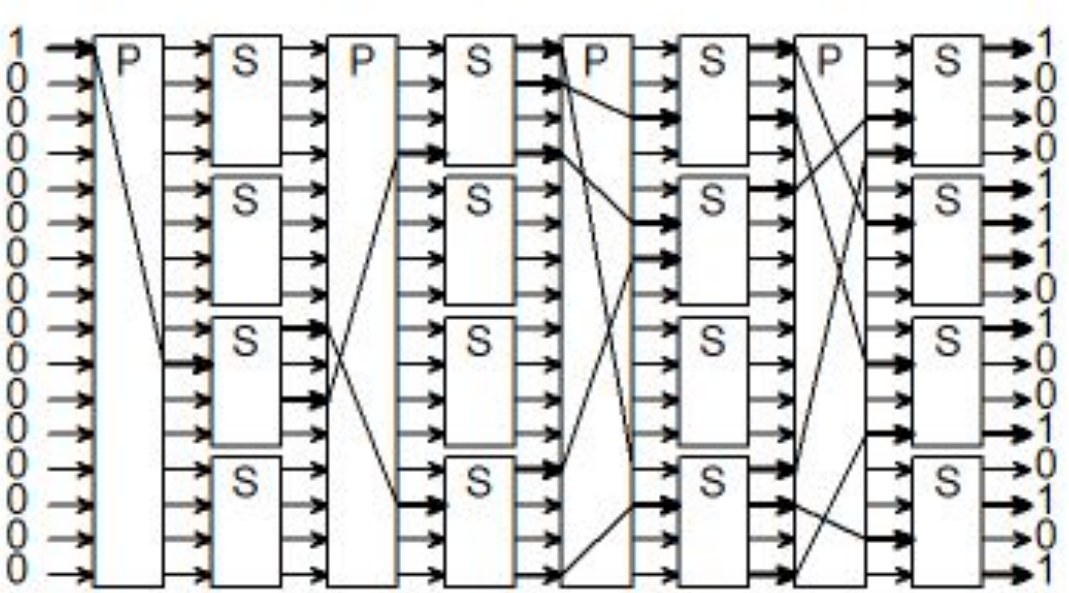
Симметричные криптосистемы – криптосистемы использующие один и тот же ключ для шифрования и расшифровывания.

В процессе шифрования используется определенный алгоритм шифрования, на вход которому подаются исходное незашифрованное сообщение и ключ. Выходом алгоритма является зашифрованное сообщение.
Ключ является значением, не зависящим от шифруемого сообщения. Изменение ключа должно приводить к изменению зашифрованного сообщения.

Зашифрованное сообщение ***C*** передается получателю. Получатель преобразует зашифрованное сообщение в исходное незашифрованное сообщение ***P*** с помощью алгоритма дешифрования и того же самого ключа ***K***.

Свойства криптоалгоритмов

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



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

Основные методы криптографического закрытия информации (шифрования)

Подстановка (замена)

- Одноалфавитная
- Многоалфавитная одноконтурная обыкновенная
- Многоалфавитная одноконтурная монофоническая
- Многоалфавитная многоконтурная

Перестановка

- Простая
- Усложненная по таблице
- Усложненная по маршрутам

Гаммирование

- С конечной короткой гаммой
- С конечной длинной гаммой
- С бесконечной гаммой

Основные методы криптографического закрытия информации (шифрования)

Аналитические преобразования

- Матричные
- По особым зависимостям

Комбинированные

- Подстановка+перестановка
- Подстановка+гаммирование
- Перестановка+гаммирование
- Гаммирование+гаммирование

Все методы основаны на использовании обратимых преобразований данных.

Подстановки

- **Моноалфавитные** (шифр простой замены). Заключаются в **замене** символов исходного сообщения на другие (того же алфавита) по более или менее сложному правилу.

  Пример: шифр Цезаря.
- **Многоалфавитные**. В отличие от моноалфавитных закон изменения символов отличается от символа к символу.

  Пример: шифр Виженера.
- **Вместо замены символа может происходить замена группы символов.**

  Пример: шифр Плейфера.

Перестановки

- Заключаются в перестановке местами символов исходного текста по некоторому правилу.

Пример: переписать символы исходного сообщения сзади на перед. ПРИВЕТ - ТЕВИРП

Гаммирование

Суть метода состоит в том, что символы некоторой специальной последовательности, называемой гаммой, последовательно накладываются по определенному закону на символы шифруемого текста ГОСТ 28147-89]. Иногда такой метод представляют как наложение гаммы на исходный текст, поэтому он получил название "гаммирование". Пример гаммирования: сложение текста с ключом по модулю алфавита.

Тогда для зашифрования: $C_i=T_i+K_i \pmod N$

Для расшифрования: $T_i=C_i-K_i \pmod N$

здесь Ci – i-тый символ шифротекста, Ti – i-й символ открытого текста, Ki -I-й символ ключа.

Пример: Шифр Вернама.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
N=	32																														
с	е	к	р	е	т			17	5	10	16	5	18																		
к	р	и	п	т	о			10	16	8	15	18	14																		
								27	21	18	31	23	32	+																	
ь	ц	у	а	ш	б			27	21	18	31	23	0	Mod 32																	
ь	ц	у	а	ш	б			27	21	18	31	23	0																		
к	р	и	п	т	о			10	16	8	15	18	14																		
								17	5	10	16	5	-14	-																	
с	е	к	р	е	т			17	5	10	16	5	18	Mod 32																	

Наиболее распространенный вариант – представление данных и ключа в двоичном виде, и применение сложения по модулю 2.

Шифрование с помощью аналитических преобразований

Достаточно надежное закрытие информации может обеспечить использование при шифровании некоторых аналитических преобразований. Например, можно использовать методы алгебры матриц - в частности умножение матрицы на вектор.

Пример: В качестве ключа задается квадратная матрица ||a|| размера n*n.

Зашифровывание. Исходный текст разбивается на блоки длиной n символов. Каждый блок рассматривается как n-мерный вектор. А процесс шифрования блока заключается в получении нового n-мерного вектора (зашифрованного блока) как результата умножения матрицы ||a|| на исходный вектор.

Расшифрование текста происходит с помощью такого же преобразования, только с помощью матрицы, обратной ||a||. Очевидно, что ключевая матрица ||a|| должна быть невырожденной.

Методы шифрования на основе матричных вычислений использованы, например, в шифре **AES**.

Блочные шифры

- Представляют семейство обратимых преобразований блоков (частей фиксированной длины) исходного текста.
- Используют комбинированные методы гаммирования, перестановки и подстановки.

N-разрядным блоком называют последовательность из нулей и единиц длины N.

$$X = (x_0 , x_1 , \ldots, x_{N-1})$$

X – можно рассматривать как вектор или как число.

Зашифрование – замена исходного блока X на блок Y в соответствие с заданным алгоритмом и ключом K.

Расшифрование – замена блока Y на блок X в соответствие с заданным алгоритмом и ключом K.

Сеть Фейстеля

В 1971 году Хорст Фейстель (Horst Feistel) запатентовал два устройства, реализовавшие различные алгоритмы шифрования, названные затем общим названием «Люцифер» (Lucifer). Одно из устройств использовало конструкцию, впоследствии названную «сетью Фейстеля» («Feistel cipher», «Feistel network»).

Сеть Фейстеля имеет следующую структуру.

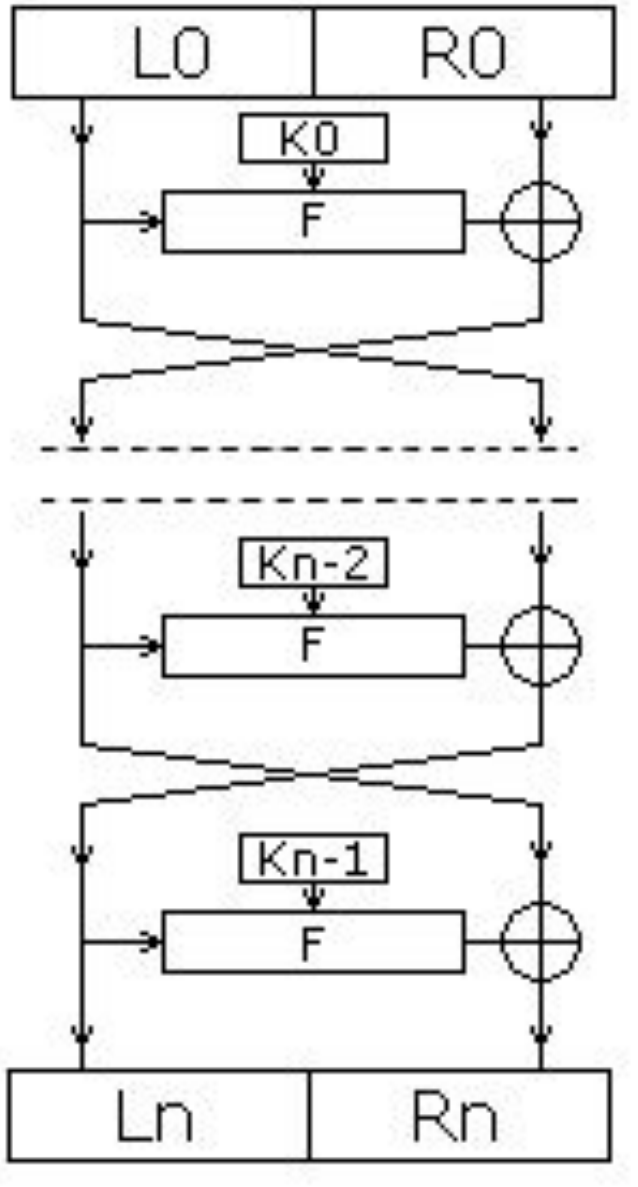
1. Входной блок делится на несколько подблоков равной длины, называемых ветвями (в случае, если блок имеет длину 64 бита, используются две ветви по 32 бита каждая).
2. Каждая ветвь обрабатывается независимо от другой, после чего осуществляется циклический сдвиг всех ветвей влево.
3. Такое преобразование выполняется несколько циклов или раундов.

Зашифровывание

- Выбранный блок делится на два равных подблока — «левый» (*L*0) и «правый» (*R*0).
- «Левый подблок» *L*0 видоизменяется функцией *f*(*L*0,*K*0) в зависимости от раундового ключа *K*0, после чего он складывается по модулю 2 с «правым подблоком» *R*0.
- Результат сложения присваивается новому левому подблоку *L*1, который будет половиной входных данных для следующего раунда, а «левый подблок» *L*0 присваивается без изменений новому правому подблоку *R*1, который будет другой половиной.
- После чего операция повторяется ещё N-1 раз, при этом при переходе от одного этапа к другому меняются раундовые ключи (*K*0 на *K*1 и т. д.) по какому-либо математическому правилу, где N — количество раундов в заданном алгоритме.

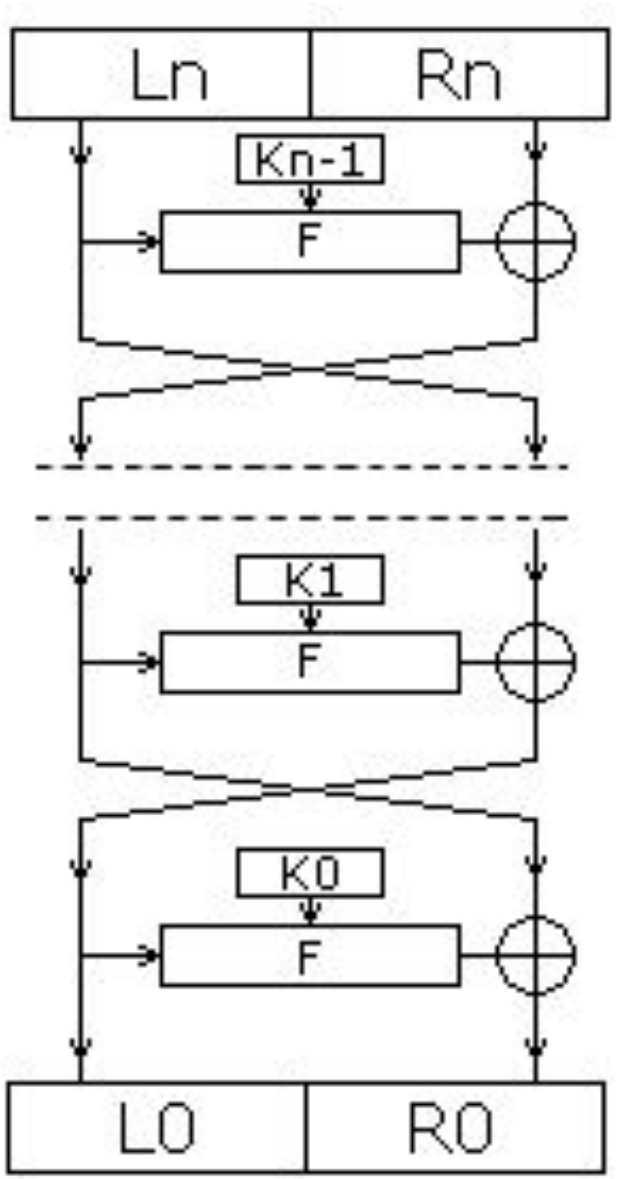
$$R_i = L_{i-1}$$
$$L_i = F(K_{i-1}, L_i) \text{ xor } R_{i-1}$$

Генерация раундовых ключей происходит на базе ключа шифрования и зависит от алгоритма шифрования.



Расшифровывание

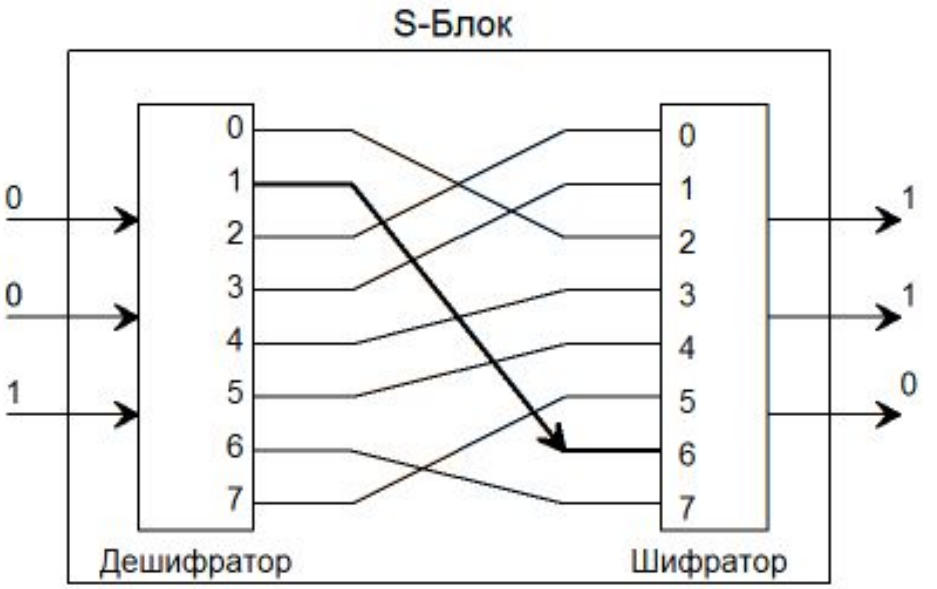
Происходит так же, как и зашифровывание, с тем лишь исключением, что ключи идут в обратном порядке, то есть не от первого к N-ному, а от N-го к первому.



Функция F - S-box (подстановка)

Блок подстановок (S-блок) состоит из:

- **дешифратора,** преобразующего *n*-разрядный двоичный сигнал в одноразрядный сигнал по основанию 2^n,
- **системы коммутаторов** внутренних соединений (всего соединений 2^n)
- **шифратора**, переводящего сигнал из одноразрядного 2^n-ричного в n-разрядный двоичный.



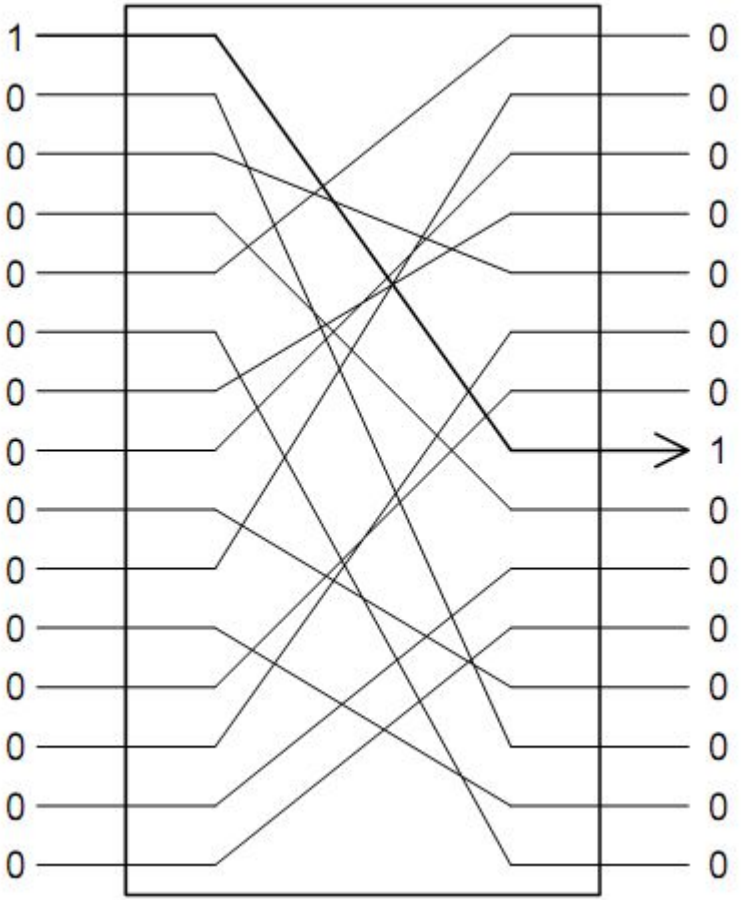
Эквивалентная таблица замен для рассматриваемого трех разрядного S-Box

№ комбинации	0	1	2	3	4	5	6	7
Вход	000	001	010	011	100	101	110	111
Выход	010	110	000	001	011	100	111	101

В электронике можно непосредственно применять приведённую схему, в программировании же генерируют таблицы замены. Оба этих подхода являются эквивалентными, то есть файл, зашифрованный на компьютере, можно расшифровать на электронном устройстве и наоборот.

Функция F - P-box (перестановка)

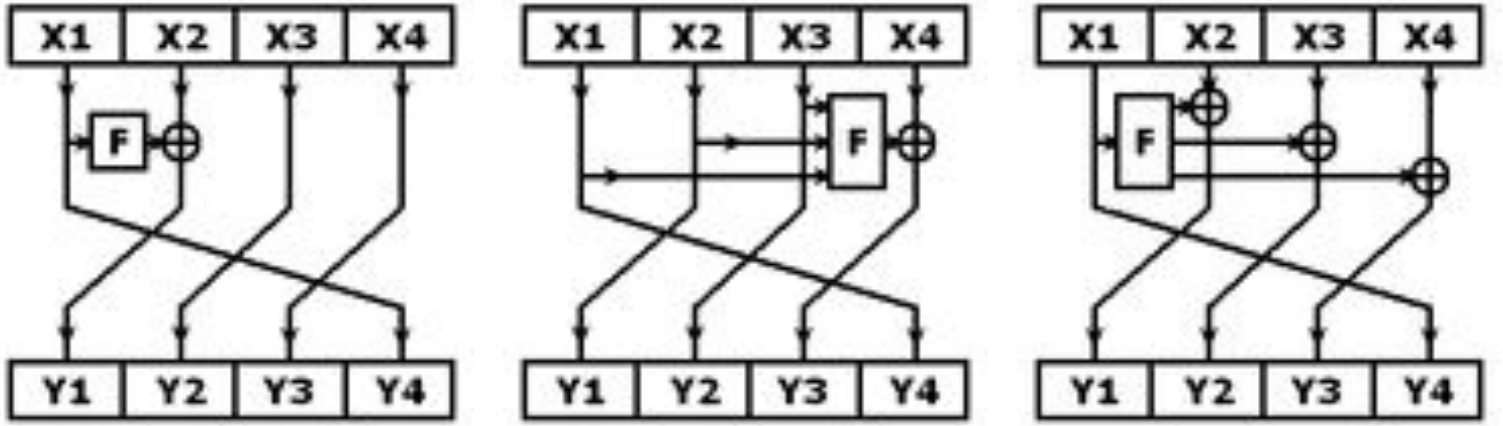
Блок перестановок изменяет положение бит в сообщении и является линейным устройством. Этот блок может иметь очень большое количество входов-выходов, однако в силу линейности систему нельзя считать криптоустойчивой.



Модификации сети Фейстеля

При большом размере блоков шифрования (128 бит и более) реализация такой сети Фейстеля на 32-разрядных архитектурах может вызвать затруднения, поэтому применяются модифицированные варианты этой конструкции.

Обычно используются сети с 4 ветвями. Также существуют схемы, в которых длины ветвей не совпадают. Они называются *несбалансированными.*



Особенности Сети Фейстеля

- Увеличение количества раундов значительно увеличивает криптостойкость алгоритма. Возможно, эта особенность и повлияла на столь активное распространение сети Фейстеля, так как для большей криптостойкости достаточно просто увеличить количество раундов, не изменяя сам алгоритм. В последнее время количество раундов не фиксируется, а лишь указываются допустимые пределы.
- Сеть Фейстеля является обратимой даже в том случае, если функция F не является таковой, так как для дешифрования не требуется вычислять F^{-1} . Для дешифрования используется тот же алгоритм, но на вход подается зашифрованный текст, и ключи используются в обратном порядке.

Особенности Сети Фейстеля

- В настоящее время все чаще используются различные разновидности сети Фейстеля для 128-битного блока с четырьмя ветвями. Увеличение количества ветвей, а не размерности каждой ветви связано с тем, что наиболее популярными до сих пор остаются процессоры с 32-разрядными словами, следовательно, оперировать 32-разрядными словами эффективнее, чем с 64-разрядными.
- Основной характеристикой алгоритма, построенного на основе сети Фейстеля, является функция F. Различные варианты касаются также начального и конечного преобразований. Подобные преобразования, называемые забеливанием (whitening), осуществляются для того, чтобы выполнить начальную рандомизацию входного текста.

Основные критерии используемые при разработке алгоритмов

- Масштабируемый ключ до 256 бит устойчивый к прямому перебору.
- Высокая скорость работы.
- Простота реализации на современной элементной базе и микропроцессорах.
- Отсутствие слабых ключей.

Алгоритмы на базе сети Фейстеля

Алгоритм	Год	Число раундов	Длина ключа	Размер блока	Количество подблоков
Blowfish	1993	16	до 448	64	2
Carman-II	2008	64/36/16	512/384/256	128	4
CAST-128	1996	12/16	40-128	64	2
CAST-256	1998	12×4=48	128/192/256	128	2
DES	1977	16	56	64	2
ГОСТ 28147-89	1989	32/16	256	64	2
IDEA	1991	8+1	128	64	4
Lucifer	1971	16	48/64/128	48/32/128	2
MARS	1998	32	128—448	128	2
RC2	1987	16+2	8-128	64	4
RC5	1994	1-255(12)	0-2040(128)	32/64/128	2
RC6	1998	20	128/192/256	128	4
Serpent	1998	32	128/192/256	128	4
Triple DES	1978	32/48	112/168	64	2
Twofish	1998	16	128/192/256	128	4

Алгоритм DES. Принципы разработки

Одним из первых реализованных и наиболее известным алгоритмом **симметричного шифрования** является **DES** (Data Encryption Standard).

Алгоритм был разработан в 1977 году, в 1980 году был принят **NIST** (National Institute of Standards and Technolody США) в качестве стандарта (FIPS PUB 46).

DES является классической сетью Фейстеля с двумя ветвями.

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

Алгоритм преобразует за несколько раундов 64-битный вход в 64-битный выход.

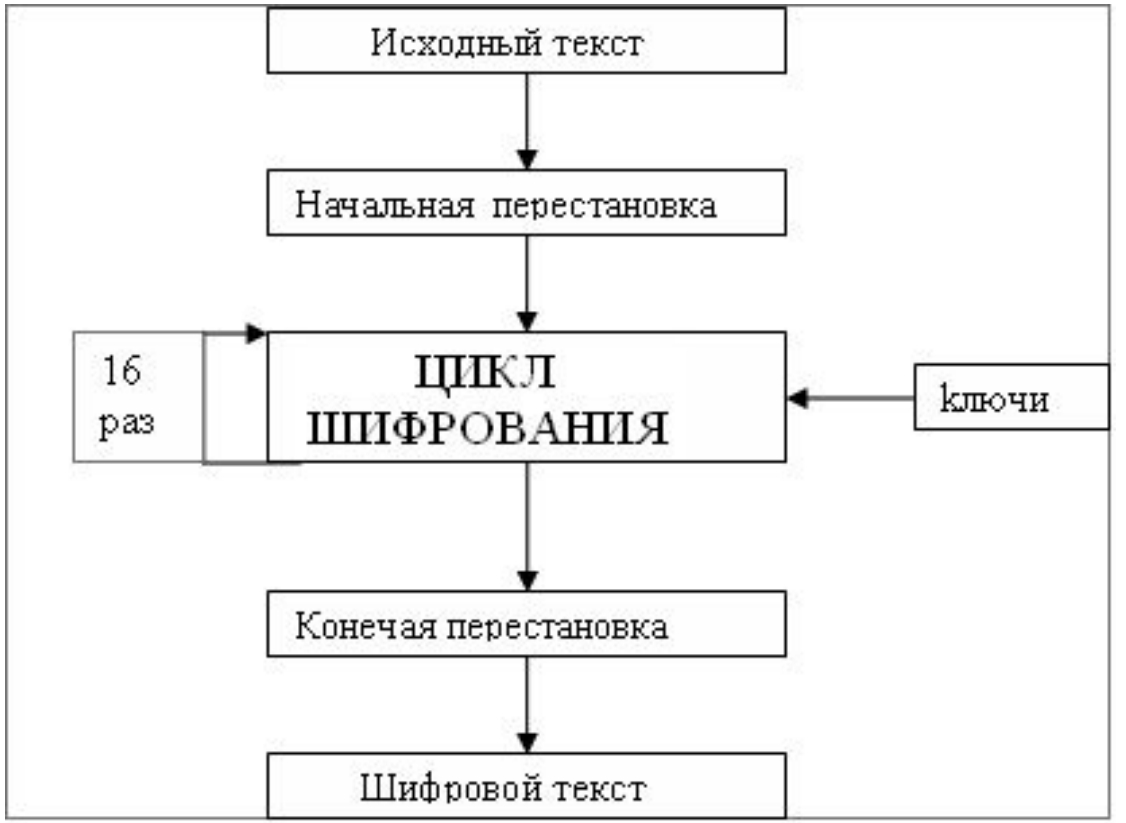
Зашифрование

1.Начальная перестановка (IP) 64-битного исходного текста.

2.Использование сети Фейстеля 16 раундов.

3.Левая и правая половины выхода последней (16-й) итерации меняются местами.

4.Конечная перестановка IP-1 результата, полученного на третьем этапе. Перестановка IP-1 инверсна начальной перестановке.



Расшифрование

Процесс расшифрования аналогичен процессу шифрования.

На входе алгоритма используется зашифрованный текст, но ключи K_i используются в обратной последовательности.

K_{16} используется на первом раунде,

K_1 используется на последнем раунде.

Особенности DES

1. Используется начальное и конечное перемешивание.
2. S-блоки на входе имеют 6 разряда и на выходе 4
3. Используется сложная схема генерации раундовых подключей.

Проблемы DES

В настоящее время основным недостатком DES **считается маленькая длина ключа**, поэтому уже давно начали разрабатываться различные альтернативы этому алгоритму шифрования.

- Один из подходов состоит в том, чтобы разработать новый алгоритм.
- Другой подход предполагает повторное применение шифрования с помощью DES с использованием нескольких ключей.

Недостатки двойного DES

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

Используя незашифрованное сообщение P и два ключа K1 и K2, зашифрованное сообщение C можно получить следующим образом:

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

Для дешифрования требуется, чтобы два ключа применялись в обратном порядке:

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

В этом случае длина ключа равна

$$56 * 2 = 112 \text{ бит.}$$

Атака "встреча посередине"

Для приведенного выше алгоритма двойного DES существует так называемая атака "встреча посередине".

Она основана на следующем свойстве алгоритма. Мы имеем

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

Атака состоит в следующем. Требуется, чтобы атакующий знал хотя бы одну пару незашифрованный текст и зашифрованный текст: (P, C). В этом случае:

1. Шифруется P для всех возможных 2^{56} значений K_1. Этот результат запоминается в таблице, и затем таблица упорядочивается по значению X.
2. Дешифрируется C, с применением всех возможных 2^{56} значений K_2.

Атака "встреча посередине"

3. Для каждого выполненного дешифрования ищется равное ему значение в первой таблице. Если соответствующее значение найдено, то считается, что эти ключи могут быть правильными, и они проверяются для следующей известной пары незашифрованный текст - зашифрованный текст.

Если известна только одна пара значений: незашифрованный текст - зашифрованный текст, то может быть получено достаточно большое число неверных значений ключей.

Но если противник имеет возможность перехватить хотя бы две пары значений (незашифрованный текст - зашифрованный текст), то сложность взлома двойного DES фактически близка сложности взлома обычного DES, $2^{56}+ 2^{56} << 2^{112}$.

Тройной DES (3DES)

Очевидное противодействие атаке "встреча посередине" состоит в использовании третьей **стадии шифрования с тремя различными ключами.**

Это поднимает стоимость лобовой атаки до 2^{168}, которая на сегодняшний день считается выше практических возможностей. При этом длина ключа равна 56 * 3 = 168 бит

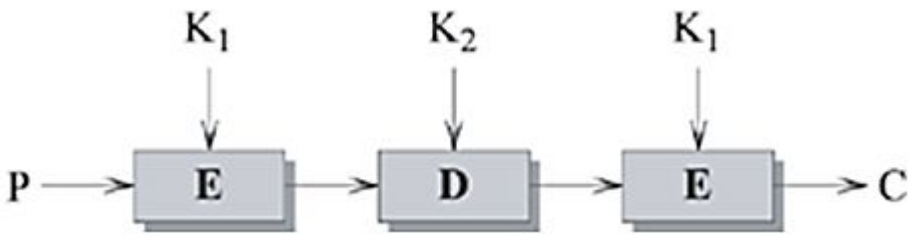
DES(k3,DES(k2,DES(k1,M))) – вариант EEE

В качестве альтернативы предлагается **метод тройного шифрования, использующий только два ключа.**

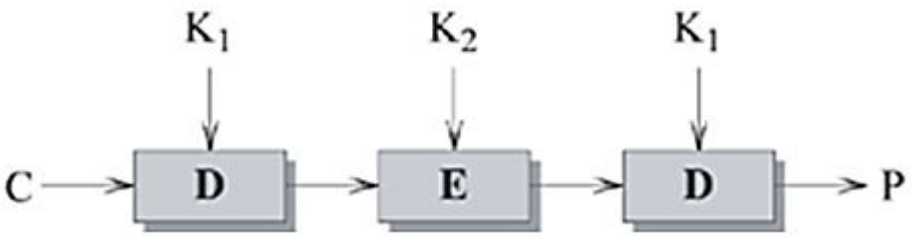
В этом случае выполняется последовательность

зашифрование-расшифрование-зашифрование (EDE).

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



Шифрование тройным DES

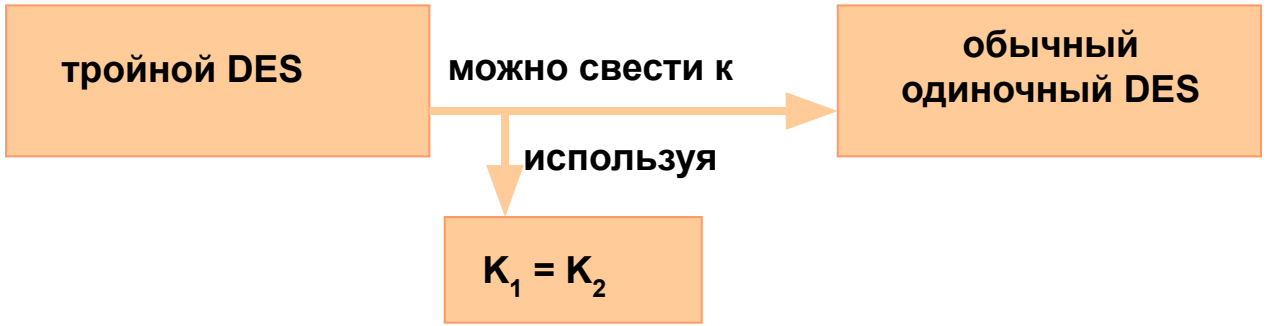


Дешифрование тройным DES

Тройной DES (3DES)

Не имеет большого значения, что используется на второй стадии: **шифрование** или **дешифрование**.

В случае использования **дешифрования** существует только то преимущество, что



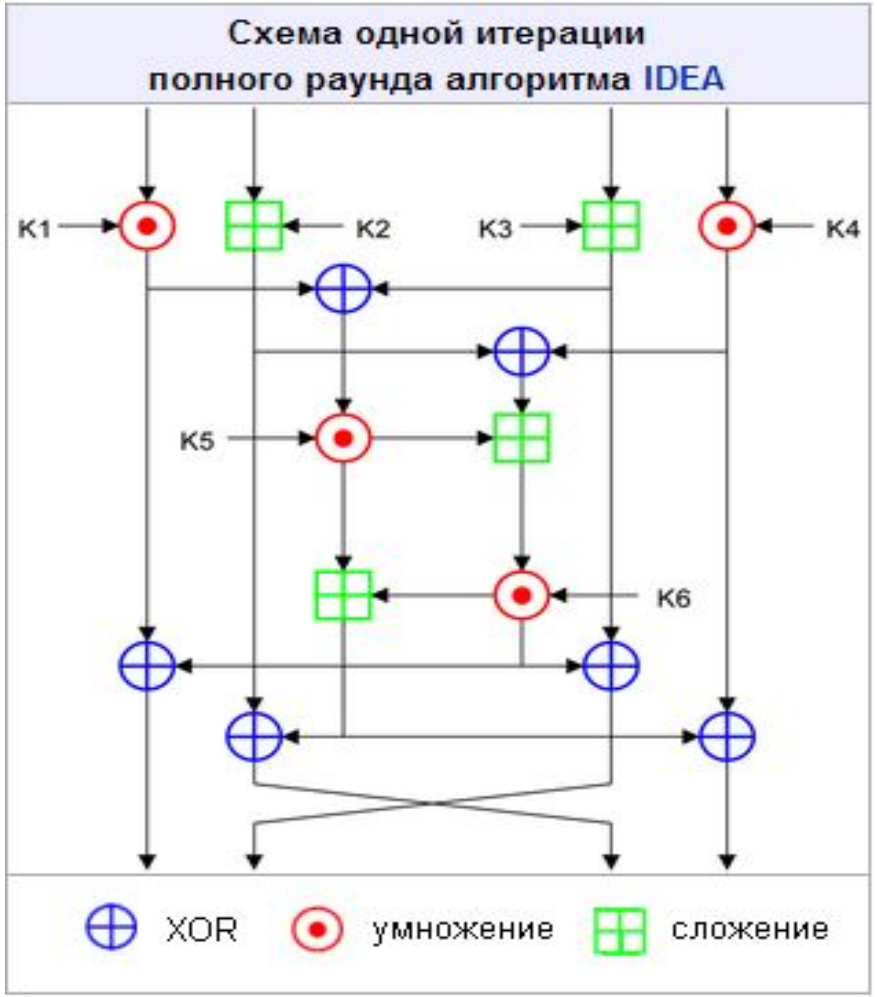
$$C = E_{K1}\,[D_{K1}\,[E_{K1}\,[P]]] = E_{K1}\,[P]$$

Тройной DES является достаточно популярной альтернативой DES и используется при управлении ключами в стандартах

ANSI X9.17 и ISO 8732 и в PEM (Privacy Enhanced Mail).

Существуют некоторые разновидности атак типа встречи по середине на тройной DES, однако они требуют гигантских объемов памяти и с практической точки зрения неосуществимы, хотя быстрее чем время прямого перебора.

Шифр IDEA



IDEA (International Data Encryption Algorithm) – алгоритм использующий глубоко модифицированную сеть Фейстеля.

- Размер блока – 64 бита.
- Размер ключа – 128 бит.
- Количество раундов – 8,5

Алгоритм 64-х битные входные блоки данных (X0) делит на 4 подблока длиной 16 бит X0 = {X(0),X(1),X(2),X(3)}.
На каждом этапе используется 6 16-ти битных подключей.
Всего используется 8 основных этапов и 1 укороченный
Используемые операции:

- сложение по модулю 2^{16}
- умножение по модулю $2^{16} + 1$
- побитовое исключающее ИЛИ (XOR).
- **S-блоки не используются**

Высокая криптостойкость IDEA обеспечивается 128 битовым ключом, а также тем, что шифрование зависит от ключа сложным и запутанным образом Скорость шифрования IDEA больше в 2 раза, чем у DES и выше чем у ГОСТ 28147-89.
Алгоритм имеет пространство слабых ключей.

Симметричный алгоритм блочного шифрования. Advanced Encryption Standard (AES)

Национальный институт стандартов и технологий США NIST 26 ноября 2001 опубликовал спецификацию FIPS-197 Advanced Encryption Standard (AES) в которой AES был объявлен стандартом шифрования.

AES базируется на алгоритме Rijndael (Рэйндэл), который был выбран по результатам открытого пятилетнего конкурса. Среди других претендентов были: CAST-256, CRYPTON, DEAL, DFC, E2, FROG, HPC, LOKI97, MAGENTA, MARS, RC6, SAFER+, Serpent, Twofish

На итоговой конференции провели голосование:

Rijndael:	**86 за,**	**10 против**
Serpent:	**59 за,**	**7 против**
Twofish:	**31 за,**	**21 против**
RC6:	**23 за,**	**37 против**
MARS:	**13 за,**	**83 против**

Алгоритм Rijndael

Создатель: Винсент Рэймен (Vincent Rijmen), Йоан Даймен (Joan Daemen)

Создан: 1998 г.

Опубликован: 2001 г.

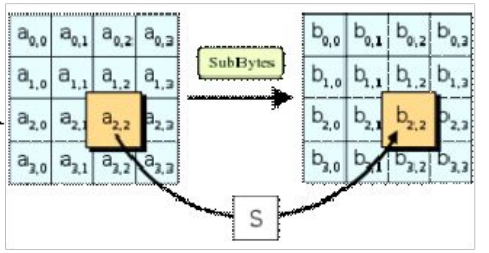
Размер ключа: 128/192/256 бит

Размер блока: 128 бит

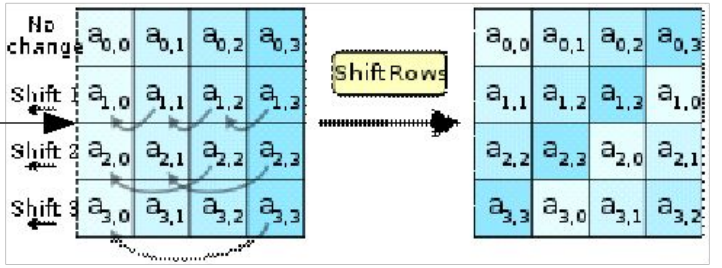
Число раундов: 10/12/14 (зависит от размера ключа)

Блок-схема раунда AES

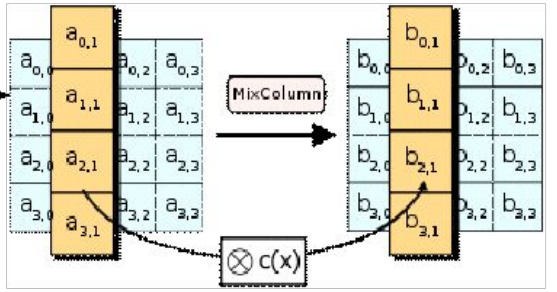
Подстановка над состоянием ***State***
(S-box)



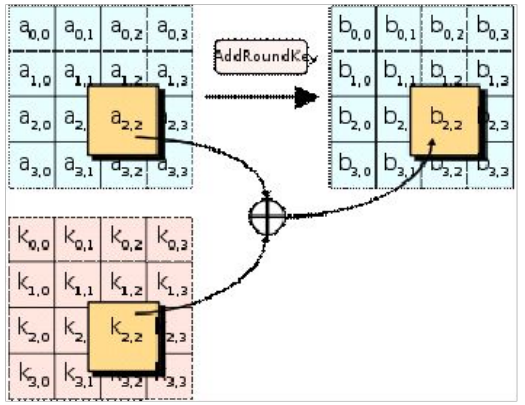
Перестановка над строками ***State***
(P-box)



Матричное преобразование
столбцов



Получение раундового ключа
и
гаммирование им состояния



ГОСТ 28147-89

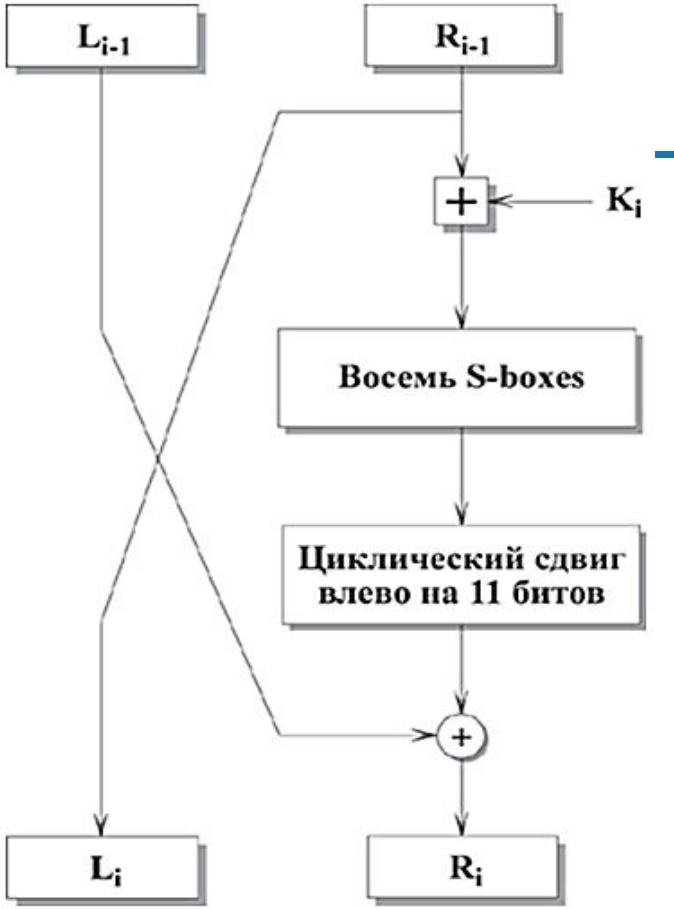
ГОСТ 28147-89 - отечественный стандарт симметричного шифрования.

Полное название: «ГОСТ 28147-89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования».

Это **единственный** разрешенный госрегулятором алгоритм шифрования информации на территории Российской Федерации.

Блочный шифр с 256-битным ключом и 32 циклами преобразования, оперирующий 64-битными блоками.

Основа алгоритма шифра — Сеть Фейстеля. Базовым режимом шифрования по ГОСТ 28147-89 является режим простой замены (определены также более сложные режимы: гаммирование, гаммирование с обратной связью и режим имитовставки).



Создатель:	***КГБ, 8-е управление***
Создан:	***1989 г.***
Опубликован:	***1990 г.***
Размер ключа:	***256 бит***
Размер блока:	***64 бит***
Число раундов:	***32\16***
Тип:	***Сеть Фейстеля***

ГОСТ 28147-89

$$L_i = R_i$$

$$R_i = L_i \oplus f(R_{i-1}, K_i)$$

Функция f проста. Сначала правая половина и i-ый подключ складываются по модулю 2^{32}. Затем результат делится на 8 частей по 4 бита и передается на блок подстановки, состоящий из 8 уникальных узлов замен (S-блоков). Каждый S-блок обрабатывает свою часть входного значения и преобразует 4 входных бита в 4 выходных. S-блок описывает как таблица состоящая из 8 строк и 16 столбцов по 4 бита в каждой ячейке.

Выходы всех S-блоков объединяются в 32-битное слово, которое затем циклически сдвигается на 11 битов влево (P-блок). Наконец, с помощью XOR результат объединяется с левой половиной, в результате чего получается новая правая половина.

ГОСТ 28147-89

1.7) … таблицы блока подстановки являются секретными элементами и поставляются в установленном порядке…

ГОСТ 28147-89

Генерация ключей проста. 256-битный ключ разбивается на восемь 32-битных подключей. Алгоритм имеет 32 раунда, поэтому каждый подключ используется в четырех раундах по следующей схеме:

Раунд	1	2	3	4	5	6	7	8
Подключ	1	2	3	4	5	6	7	8
Раунд	9	10	11	12	13	14	15	16
Подключ	1	2	3	4	5	6	7	8
Раунд	17	18	19	20	21	22	23	24
Подключ	1	2	3	4	5	6	7	8
Раунд	25	26	27	28	29	30	31	32
Подключ	8	7	6	5	4	3	2	1

Считается, что стойкость алгоритма ГОСТ 28147-89 во многом определяется структурой S-блоков. Структура S-блоков в стандарте не определена.

Алгоритм может работать в следующих режимах:

1. Режим простой замены (использование ограничено стандартом) - ECB
2. Режим гаммирования - OFB
3. Режим гаммирования с обратной связь - CFB
4. Режим выработки имитовставки

Критика ГОСТ 28147-89 в открытых источниках

Основные проблемы ГОСТа связаны с неполнотой стандарта в части генерации ключей и таблиц замен.

1. Тривиально доказывается, что у ГОСТа существуют «слабые» ключи и таблицы замен, но в стандарте не описываются критерии выбора и отсева «слабых».
2. Также стандарт не специфицирует алгоритм генерации таблицы замен (S-блоков). С одной стороны, это может являться дополнительной секретной информацией (помимо ключа), а с другой, поднимает ряд проблем:
    a) нельзя определить криптостойкость алгоритма, не зная заранее таблицы замен;
    b) реализации алгоритма от различных производителей могут использовать разные таблицы замен и могут быть несовместимы между собой;
    c) потенциальная возможность (отсутствие запрета в стандарте) использования таблиц замены, в которых узлы не являются перестановками, что может привести к чрезвычайному снижению стойкости шифра.

В рамках «Соглашения о совместимости СКЗИ» следующих компаний производителей СКЗИ:

- ФГУП НТЦ "Атлас",
- ООО "КРИПТО-ПРО",
- ООО "Фактор-ТС",
- ЗАО "МО ПНИЭИ",
- ООО "Инфотекс",
- ЗАО "СПбРЦЗИ",
- ООО "Криптоком",
- ООО "Р-Альфа".

Разработали общие параметры алгоритма, которые в январе 2006 года открытое международное сообщество разработчиков IETF (Internet Engineering Task Force) приняло как стандарт.

RFC 4357 "Additional Cryptographic Algorithms for Use with GOST 28147-89, GOST R 34.10-94, GOST R 34.10-2001, and GOST R 34.11-94 Algorithms".

Помимо определения параметров алгоритма (S-блоки, таблица использования подключей), данный стандарт определяет также режим работы ГОСТ 28147-89 – режим сцепления блоков CBC, а так же ряд алгоритмов ассиметричной криптографии по обмену и выработке ключей.

Основные различия между DES и ГОСТ 28147-89

DES использует гораздо более сложную процедуру создания подключей, чем ГОСТ 28147. В ГОСТ эта процедура очень проста.

В DES применяется 56-битный ключ, а в ГОСТ 28147-89 - 256-битный. При выборе сильных S-блоков ГОСТ 28147-89 считается очень стойким.

У S-блоков DES 6-битовые входы и 4-битовые выходы, а у S-блоков ГОСТ 28147-89 4-битовые входы и выходы. В обоих алгоритмах используется по восемь S-блоков, но размер S-блока ГОСТ 28147-89 существенно меньше размера S-блока DES.

В DES применяются нерегулярные перестановки P, в ГОСТ 28147-89 используется 11-битный циклический сдвиг влево. Перестановка DES увеличивает лавинный эффект. В ГОСТ 28147-89 изменение одного входного бита влияет на один S-блок одного раунда, который затем влияет на два S-блока следующего раунда, три S-блока следующего и т.д. В ГОСТ 28147-89 требуется 8 раундов прежде, чем изменение одного входного бита повлияет на каждый бит результата; DES для этого нужно только 5 раундов.

В DES 16 раундов, в ГОСТ 28147 - 32 раунда, что делает его более стойким к дифференциальному и линейному криптоанализу.

Сравнение ГОСТ 28147-89 и AES

Таблица 1. Сравнительные характеристики алгоритмов ГОСТ28147-89 и Rijndael.

Показатель	ГОСТ28147-89	Rijndael
Размер блока, бит	64	128, 192, 256
Размер ключа, бит	256	128, 192, 256
Архитектура	Однородная сбаланси-рованная сеть Файстеля	«Квадрат» (Square)
Число раундов	32	10, 12, 14

ГОСТ 28147-89 и AES обеспечивают примерно одинаковую криптостойкость

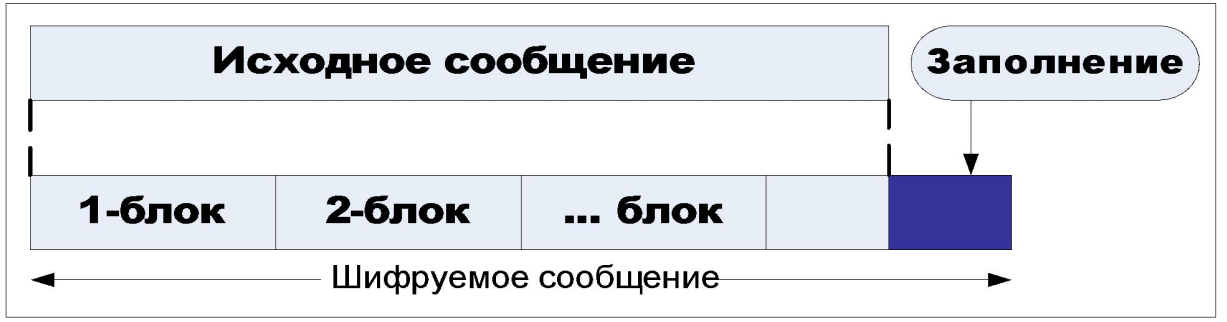
Таблица 5. Показатели быстродействия реализаций сравниваемых алгоритмов на языке Си.

	ГОСТ 28147-89	Rijndael, 14 раундов
Pentium 166	2.04 Мбайт/с	2.46 Мбайт/с
Pentium III 500	8.30 Мбайт/с	9.36 Мбайт/с

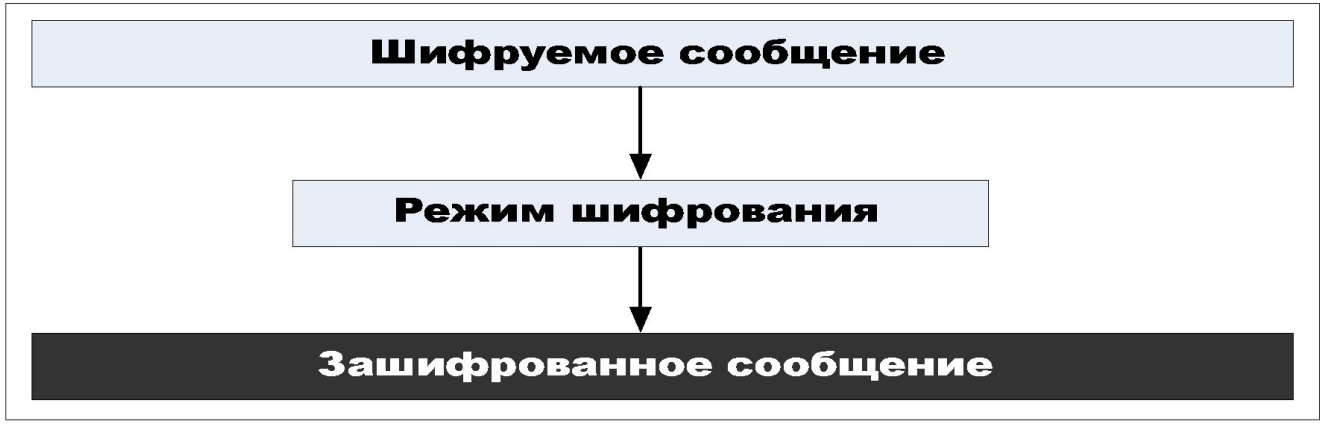
AES обладает несколько большим быстродействием

Шифрование сообщений больших размеров. Режимы шифрования

Этап 1 - заполнение



Этап 2 - шифрование



Режим шифрования — метод применения **блочного шифра**, позволяющий преобразовать последовательность блоков открытых данных в последовательность блоков зашифрованных данных. При этом для шифрации одного блока могут использоваться данные другого блока.
Обычно режимы шифрования используются для модификации процесса шифрования так, чтобы результат шифрования каждого блока был уникальным вне зависимости от шифруемых данных и не позволял сделать какие-либо выводы об их структуре.

Режимы шифрования блочных шифров

ECB - Electronic Codebook (электрнонная кодовая книга) - каждый блок незашифрованного текста шифруется независимо от остальных блоков, с применением одного и того же ключа шифрования. Типичные приложения - безопасная передача одиночных значений (например, криптографического ключа).

CBC - Chipher Block Chaining (сцепление блоков) - вход криптографического алгоритма является результатом применения операции XOR к следующему блоку незашифрованного текста и предыдущему блоку зашифрованного текста. Типичные приложения - общая блокоориентированная передача, аутентификация.

Режимы шифрования блочных шифров

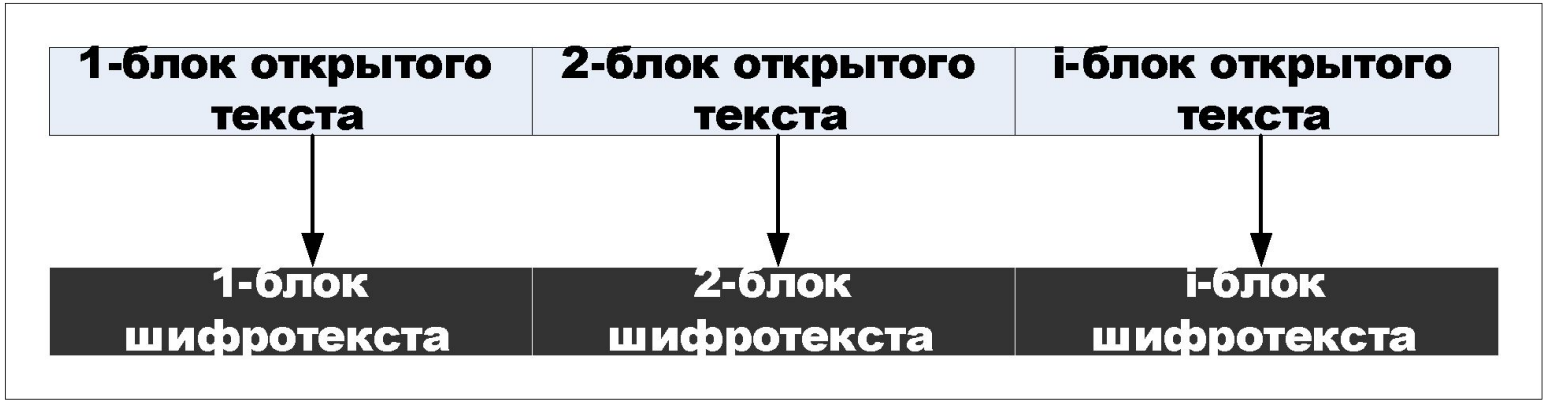
CFB - Chipher Feedback (обратная связь по шифру) - при каждом вызове алгоритма обрабатывается J битов входного значения. Предшествующий зашифрованный блок используется в качестве входа в алгоритм; к J битам выхода алгоритма и следующему незашифрованному блоку из J битов применяется операция XOR, результатом которой является следующий зашифрованный блок из J битов. Типичные приложения - потокоориентированная передача, аутентификация.

OFB - Output Feedback (обратная связь по выходу) - аналогичен CFB, за исключением того, что на вход алгоритма при шифровании следующего блока подается результат шифрования предыдущего блока; только после этого выполняется операция XOR с очередными J битами незашифрованного текста.

Типичные приложения - потокоориентированная передача по зашумленному каналу (например, спутниковая связь).

1. Режим ECB (режим электронной кодовой книги)

Данный режим является самым простым режимом, при котором незашифрованный текст обрабатывается последовательно, блок за блоком. В терминах ГОСТ 28147-89 это режим простой замены.

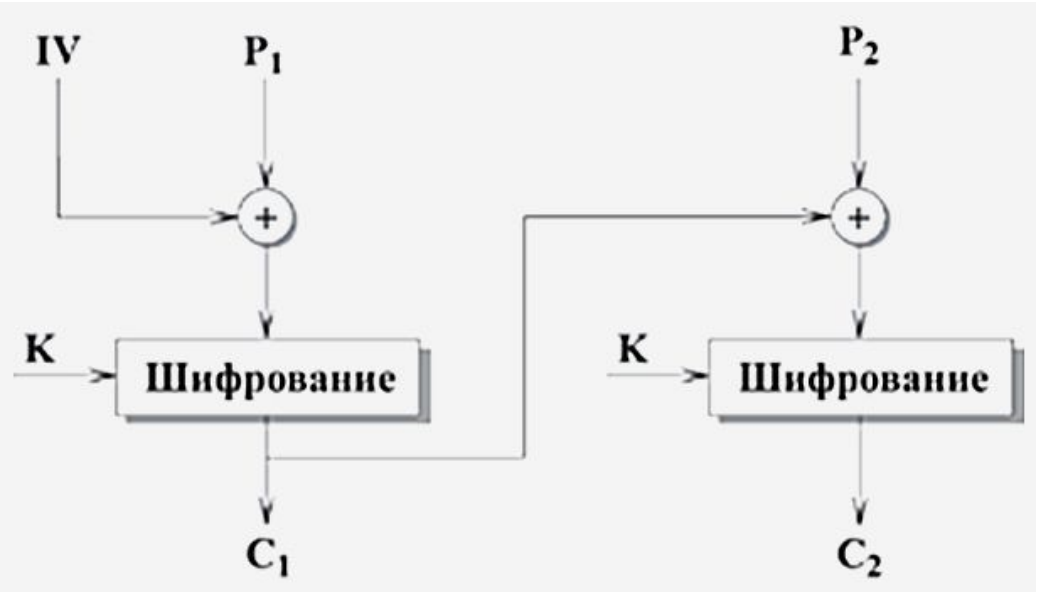


ECB-режим идеален для небольшого количества данных, например, для шифрования ключа сессии. Существенным недостатком ECB является то, что один и тот же блок незашифрованного текста, появляющийся более одного раза в сообщении, всегда имеет один и тот же зашифрованный вид. Вследствие этого для больших сообщений ECB режим считается небезопасным. Если сообщение имеет много одинаковых блоков, то при криптоанализе данная закономерность будет обнаружена.

2. Режим CBC (режим сцепления блоков)

В режиме CBC одинаковые незашифрованные блоки преобразуются в различные зашифрованные. Для этого в качестве входа алгоритма используется результат применения операции XOR к текущему незашифрованному блоку и предыдущему зашифрованному блоку.

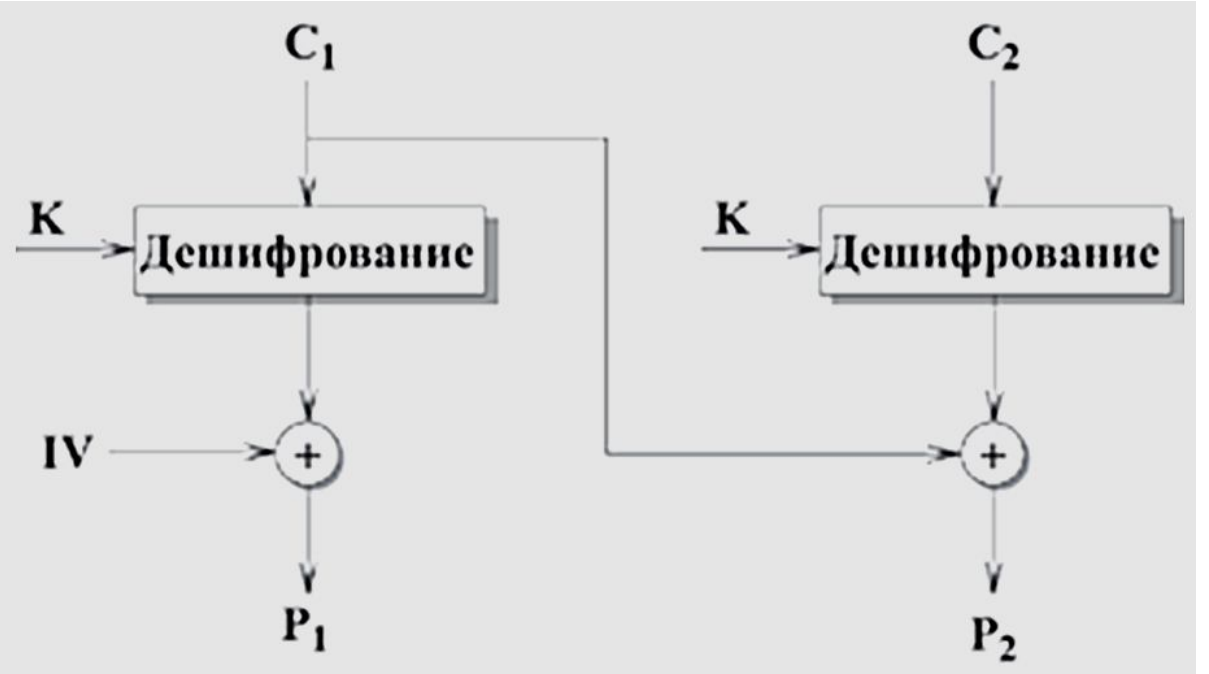
Зашифрование в CBC



Для получения первого блока зашифрованного сообщения используется инициализационный вектор (IV), или синхропосылка (в терминах ГОСТ 28147-89), для которого выполняется операция XOR с первым блоком незашифрованного сообщения.

Расшифрование в CBC

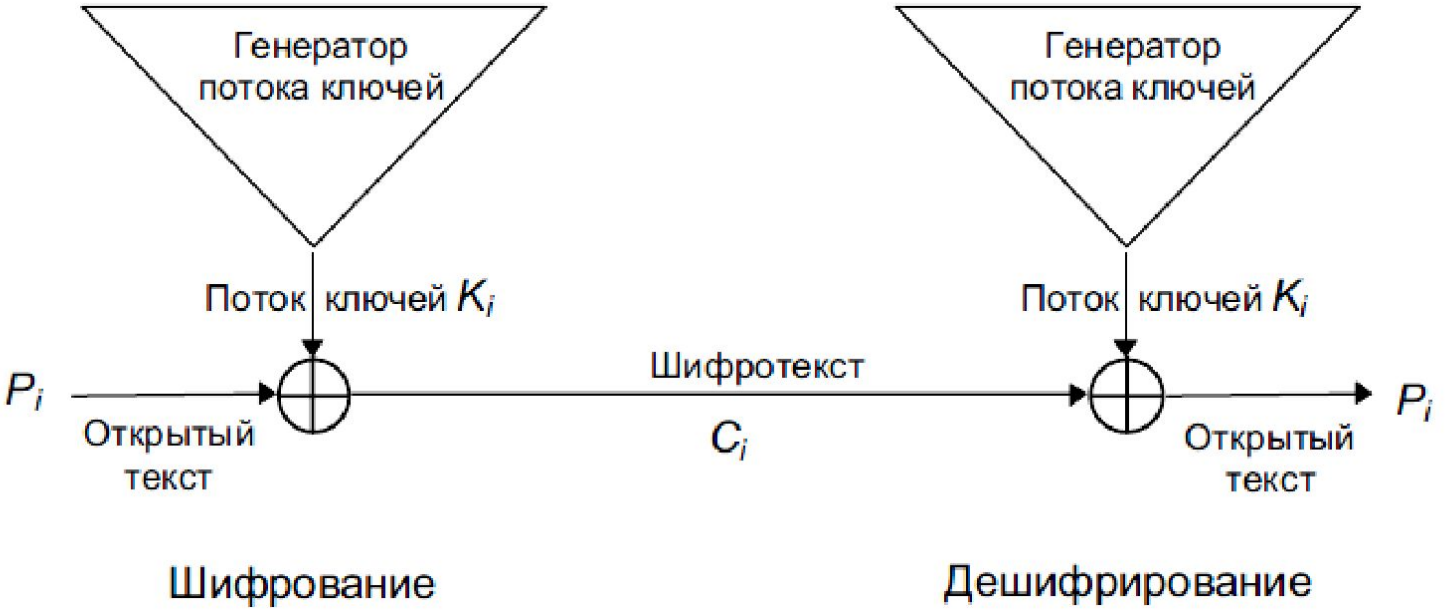
При расшифровании для IV выполняется операция XOR с выходом расшифрирующего алгоритма для получения первого блока незашифрованного текста.



IV должен быть известен как отправителю, так и получателю.

Потоковые шифры

При расшифровании для IV выполняется операция XOR с выходом расшифрирующего алгоритма для получения первого блока незашифрованного текста.



Потоковые шифры преобразуют информацию не блоком, а посимвольно. При этом используется XOR между текстом и гаммой. Гамма – потоковый ключ генерируется поточным шифром, чтобы получатель мог расшифровать сообщение надо чтобы генераторы получателя и отправителя работали синхронно. Ключ – задает параметры генерации гаммы.

Зашифрование $C_i=K_i$ xor P_i Расшифрование $P_i=K_i$ xor C_i

3. Режим CFB (режим обратной связи по шифру)

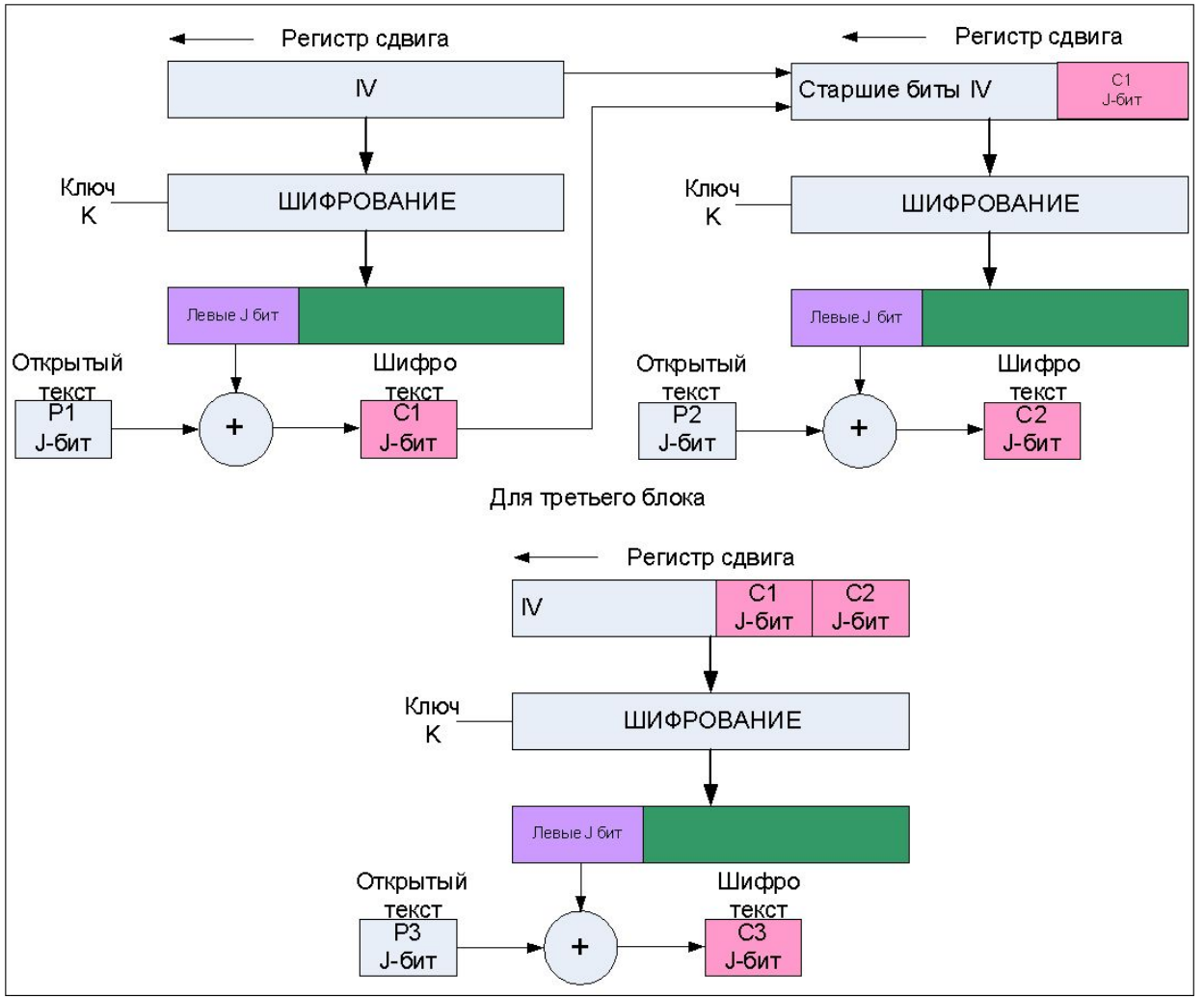
CFB – позволяет использовать блочный алгоритм как поточный. При этом:

1. Устраняется необходимость разбивать сообщение на целое число блоков достаточно большой длины.
2. Появляется возможность работать в реальном времени.
3. Зашифрованный текст будет той же длины, что и исходный.

Зашифрование в режиме CFB

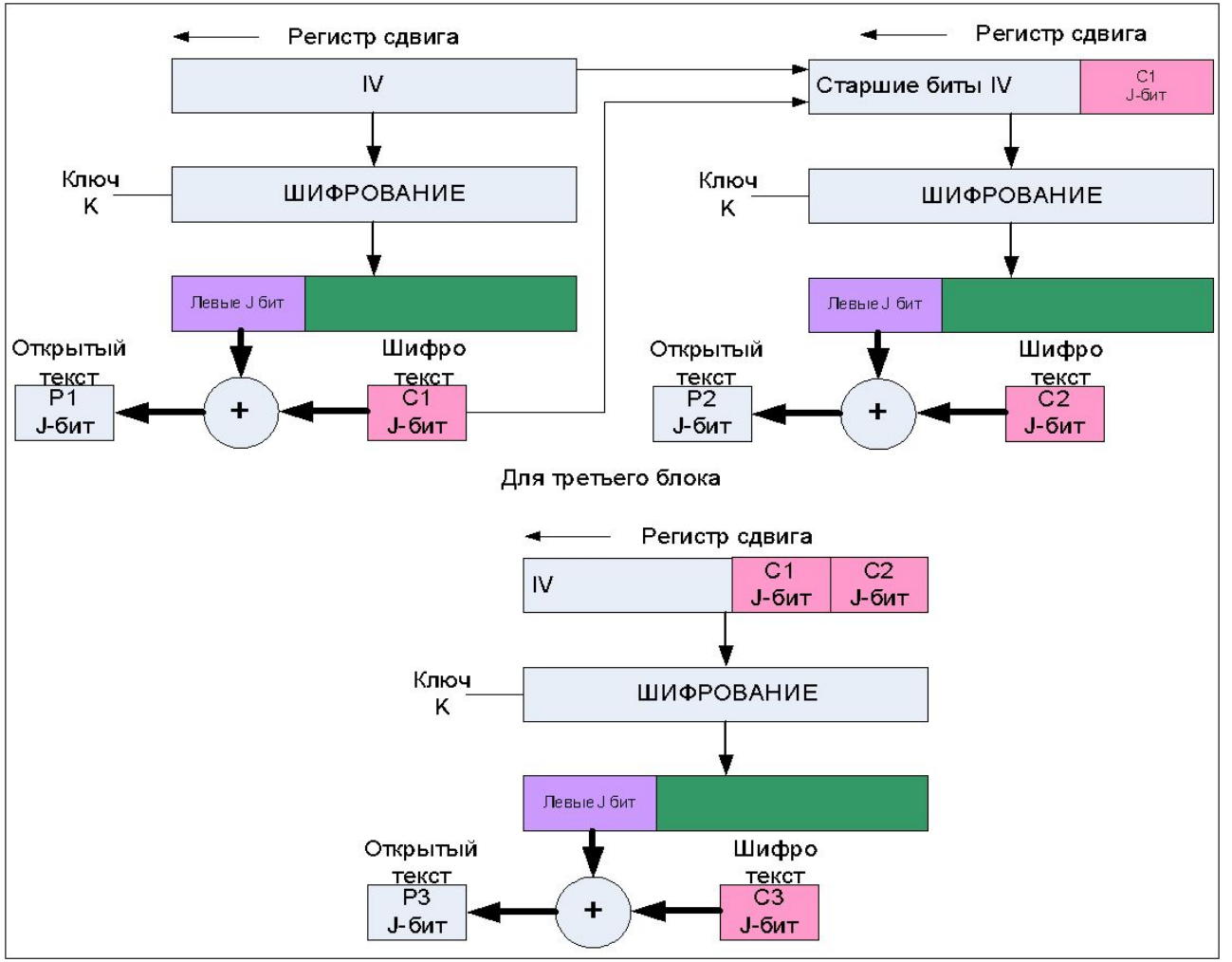
Пусть символ – J бит. Входом функции шифрования является регистр сдвига, который первоначально устанавливается в инициализационный вектор IV.
Для левых J битов **выхода** шифрования выполняется операция XOR с J битами незашифрованного текста Pi для получения первого блока зашифрованного текста Ci. Содержимое регистра сдвигается влево на J битов, и Ci помещается в правые J битов этого регистра.

Этот процесс продолжается до тех пор, пока не будет зашифровано все сообщение. Данный метод в ГОСТ 28147—89 называется "режим гаммирования с обратной связью".



Расшифрование в режиме CFB

При расшифровании используется аналогичная схема, за исключением того, что блоки зашифрованного текста по очереди извлекаются из канала или файла и являются входом алгоритма, а не выходом. Получатель должен знать IV



4. Режим OFB (обратная связь по выходу)

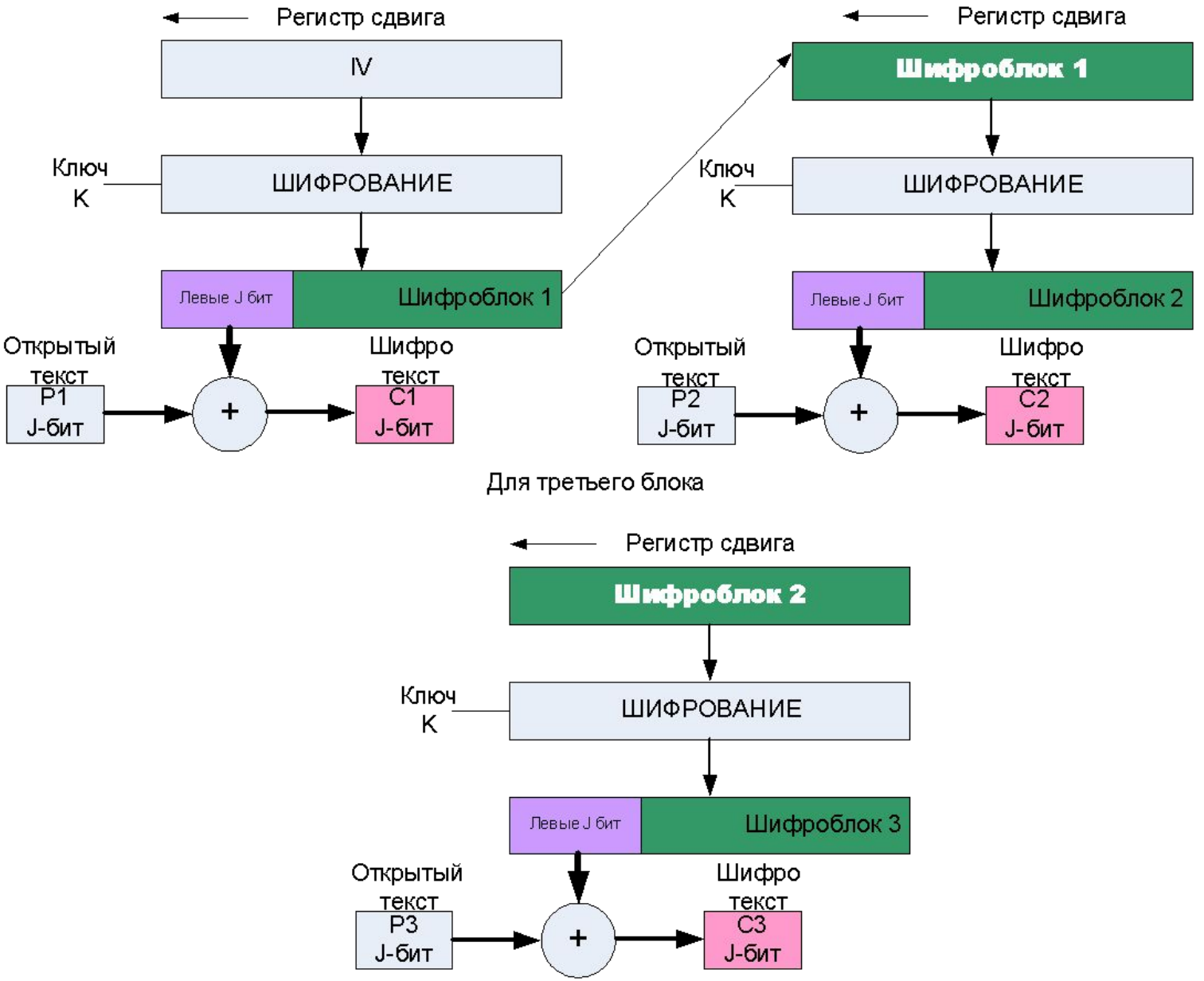
Данный режим подобен режиму CFB. Разница заключается в том, что выход алгоритма в режиме OFB подается обратно в регистр, тогда как в режиме CFB в регистр подается результат применения операции XOR к незашифрованному блоку и результату алгоритма.

Основное преимущество режима OFB состоит в том, что если при передаче произошла ошибка, то она не распространяется на следующие зашифрованные блоки, и тем самым сохраняется возможность дешифрования последующих блоков.

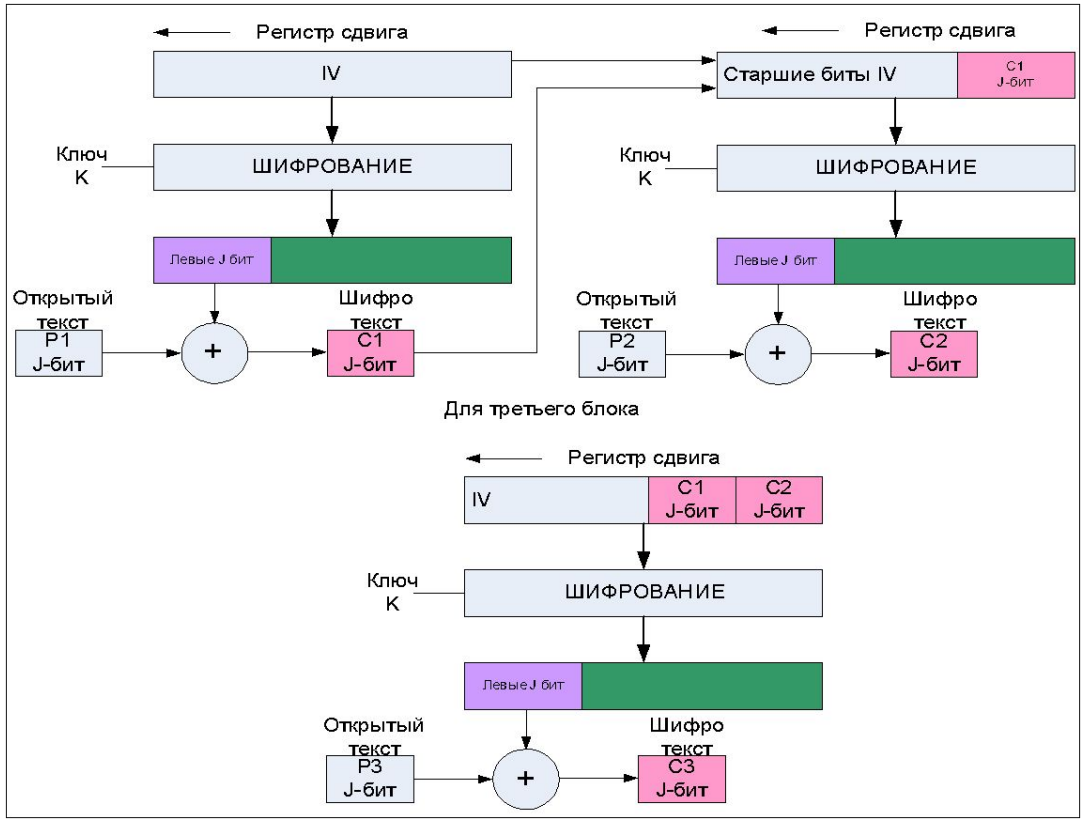
Например, если появляется ошибочный бит в C_i, то это приведет только к невозможности дешифрования этого блока и получения P_i. Дальнейшая последовательность блоков будет расшифрована корректно.

Недостаток OFB в том, что он более уязвим к атакам модификации потока сообщений, чем CFB.

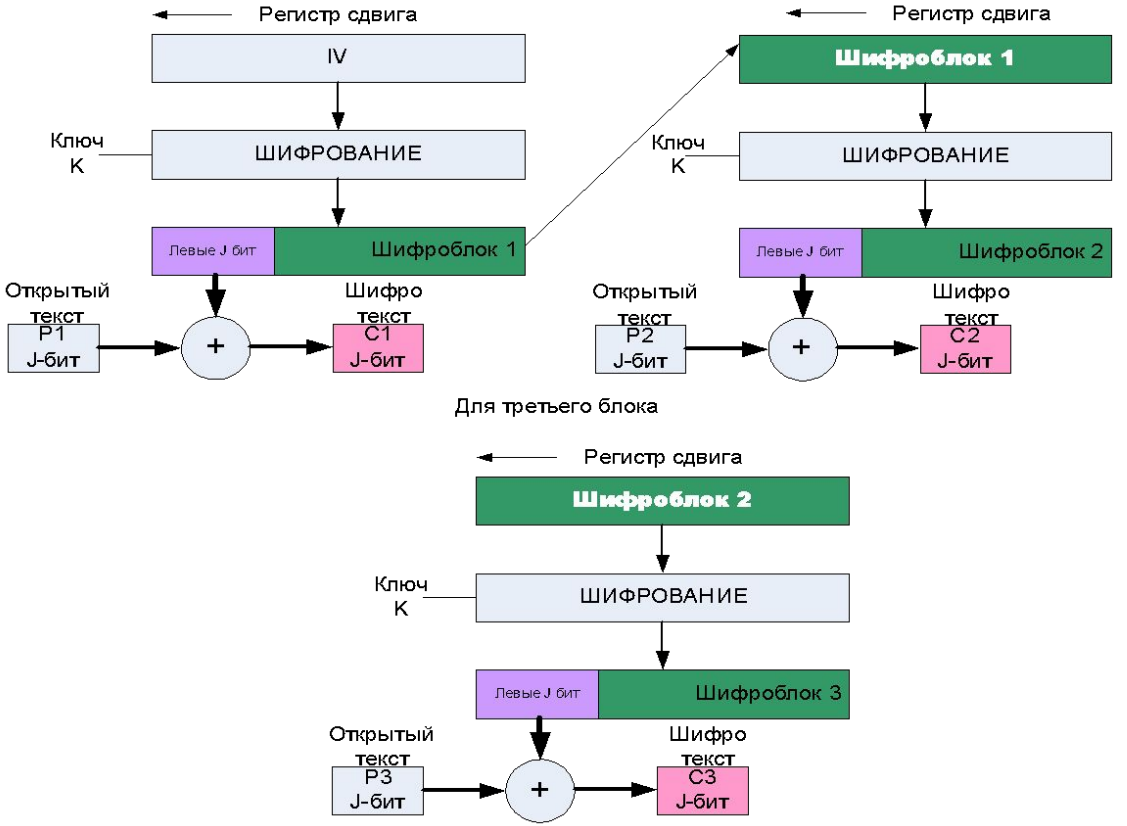
Зашифрование в режиме OFB



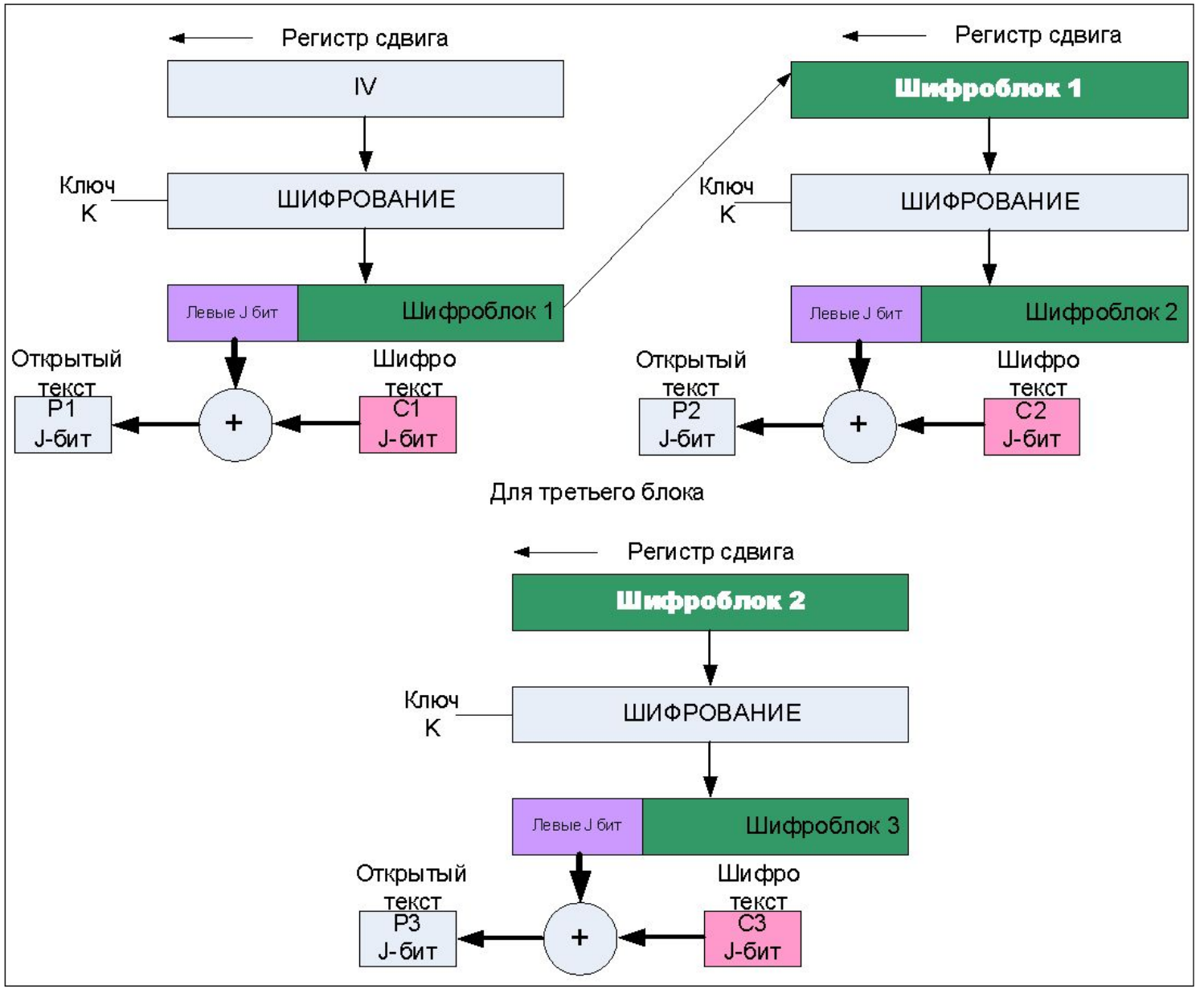
Зашифрование в режиме CFB



Зашифрование в режиме OFB



Расшифрование в режиме OFB



Создание случайных чисел

Случайные числа играют важную роль при использовании криптографии в различных сетевых приложениях, относящихся к безопасности.

Большинство алгоритмов сетевой безопасности, основанных на криптографии, использует случайные числа.
Например:
1. Схемы взаимной аутентификации.
2. Ключ сессии, созданный KDC или кем-либо из участников.
3. и т.д.

Требования к случайным числам

Двумя основными требованиями к последовательности случайных чисел являются **случайность** и **независимость**.

Случайность

Обычно при создании последовательности псевдослучайных чисел предполагается, что данная последовательность чисел должна быть случайной в некотором определенном статистическом смысле.

Следующие два критерия используются для доказательства того, что последовательность чисел является случайной:

1. Равномерное распределение: распределение чисел в последовательности должно быть равномерным т. е., частота появления каждого числа должна быть приблизительно одинаковой.
2. Независимость: ни одно значение в последовательности не должно зависеть от других.

Существуют тесты, показывающие, что последовательность чисел соответствует некоторому распределению, такому как однородное распределение.

Теста для "доказательства" независимости нет.
Но есть тесты для доказательства того, что последовательность является зависимой.

Источники случайных чисел

Источники действительно случайных чисел найти трудно:

1. Детекторы событий ионизирующей радиации,
2. Газовые разрядные трубки и имеющий течь конденсатор могут быть такими источниками.

Эти устройства в приложениях сетевой безопасности применяются ограниченно. Проблемы также вызывают грубые атаки на такие устройства.

Альтернативным решением является создание набора из большого числа случайных чисел и опубликование его в некоторой книге.
Тем не менее, и такие наборы обеспечивают очень ограниченный источник чисел по сравнению с тем количеством, которое требуется приложениям сетевой безопасности. Более того, хотя наборы из этих книг действительно обеспечивает статистическую случайность, они предсказуемы, так как противник может получить их копию.

Таким образом, шифрующие приложения используют для создания случайных чисел **специальные алгоритмы**. Эти алгоритмы детерминированы и, следовательно, создают последовательность чисел, которая не является статистически случайной. Тем не менее, если алгоритм хороший, полученная последовательность будет проходить много тестов на случайность. Такие числа часто называют **псевдослучайными числами**.

Генераторы псевдослучайных чисел

Первой широко используемой технологией создания случайного числа был алгоритм, предложенный Лехмером, который известен как метод линейного конгруента.

Этот алгоритм параметризуется четырьмя числами следующим образом:

m	Модуль (основание системы)	m > 0
a	Множитель	0 ≤ a < m
c	Приращение	0 ≤ c < m
X0	Начальное значение или зерно (seed)	$0 \le X_0 < m$

Последовательность случайных чисел $\{X_n\}$ получается с помощью следующего итерационного равенства:

$$X_{n+1} = (a \cdot X_n + c) \mod m$$

Если m, a и c являются целыми, то создается последовательность целых чисел в диапазоне 0≤Xn<m. Выбор значений для a, c и m является критичным для разработки хорошего генератора случайных чисел.

Генераторы псевдослучайных чисел

Если противник знает, что используется алгоритм линейного конгруента, и если известны его параметры ($a=7^5$, $c=0$, $m=2^{31}-1$), то, если раскрыто одно число, вся последовательность чисел становится известна.
Даже если противник знает только, что используется алгоритм линейного конгруента, знания небольшой части последовательности достаточно для определения параметров алгоритма и всех последующих чисел.
Предположим, что противник может определить значения X_0, X_1, X_2, X_3. Тогда :

$$X_1 = (a\ X_0 + c)\ \mathrm{mod}\ m$$
$$X_2 = (a\ X_1 + c)\ \mathrm{mod}\ m$$
$$X_3 = (a\ X_2 + c)\ \mathrm{mod}\ m$$

Эти равенства позволяют найти a, c и m.

Таким образом, хотя алгоритм и является хорошим генератором псевдослучайной последовательности чисел, желательно, чтобы реально используемая последовательность была непредсказуемой, поскольку в этом случае знание части последовательности не позволит определить будущие ее элементы.

Эта цель может быть достигнута несколькими способами. Например, использование внутренних системных часов для модификации потока случайных чисел.

Один из способов применения часов состоит в перезапуске последовательности после N чисел, используя текущее значение часов по модулю m в качестве нового начального значения.

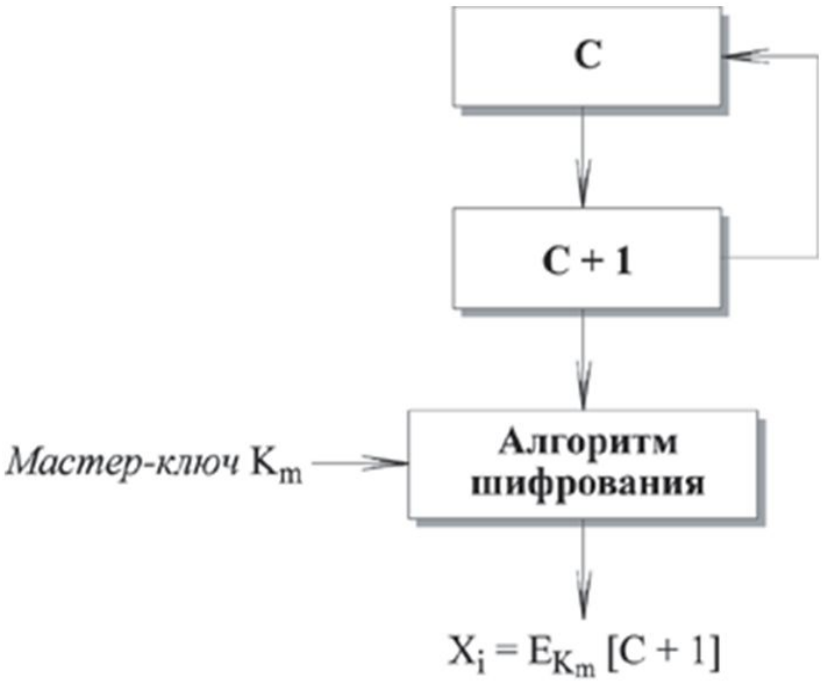
Другой способ состоит в простом добавлении значения текущего времени к каждому случайному числу по модулю m.

Криптографически созданные случайные числа

В криптографических приложениях целесообразно шифровать получающиеся случайные числа. Чаще всего используется три способа:
- Циклическое шифрование – режим счетчика
- Режим Output Feedback – обратная связь по выходу
- Генератор псевдослучайных чисел ANSI X9.17

Циклическое шифрование

Счетчик с периодом N используется в качестве входа в шифрующее устройство. Например, в случае использования 56-битного ключа DES может применяться счетчик с периодом 2^{56}. После каждого созданного ключа значение счетчика увеличивается на 1. Таким образом, псевдослучайная последовательность, полученная по данной схеме, имеет полный период: каждое выходное значение X_0, X_1,...X_{N-1} основано на различных значениях счетчика и, следовательно, $X_0 \neq X_1 \neq X_{N-1}$.



Так как мастер-ключ защищен, легко показать, что любой секретный ключ не зависит от знания одного или более предыдущих секретных ключей.
Для дальнейшего усиления алгоритма вход должен быть выходом полнопериодического генератора псевдослучайных чисел, а не простой последовательностью.

Режим Output Feedback DES

Режим OFB DES может применяться для генерации ключа, аналогично тому, как он используется для потокового шифрования.

Выходом каждой стадии шифрования является 64-битное значение, из которого только левые j битов подаются обратно для шифрования.

64-битные выходы составляют последовательность псевдослучайных чисел с хорошими статистическими свойствами.

Генератор псевдослучайных чисел ANSI X9.17

Один из наиболее сильных генераторов псевдослучайных чисел описан в ANSI X9.17. В число приложений, использующих эту технологию, входят приложения финансовой безопасности и PGP.

Алгоритмом шифрования является тройной DES. Генератор ANSI X9.17 состоит из следующих частей:

1. Вход: генератором управляют два псевдослучайных входа. Один является 64-битным представлением текущих даты и времени, которые изменяются каждый раз при создании числа. Другой является 64-битным начальным значением, оно инициализируется некоторым произвольным значением и изменяется в ходе генерации последовательности псевдослучайных чисел.
2. Ключи: генератор использует три модуля тройного DES. Все три используют одну и ту же пару 56-битных ключей, которая должна держаться в секрете и применяться только для генерации псевдослучайного числа.
3. Выход: выход состоит из 64-битного псевдослучайного числа и 64-битного значения, которое будет использоваться в качестве начального значения при создании следующего числа.

Генератор псевдослучайных чисел ANSI X9.17

DT_i - значение даты и времени на начало i-ой стадии генерации.

V_i - начальное значение для i-ой стадии генерации.

R_i - псевдослучайное число, созданное на i-ой стадии генерации.

K_1,K_2 - ключи, используемые на каждой стадии.

$$R_i = EDE_{K1,K2}\ [EDE_{K1,K2}[DT_i] \oplus V_i]$$

$$V_{i+1} = EDE_{K1,K2}\ [EDE_{K1,K2}\ [DT_i]\ R_i]$$

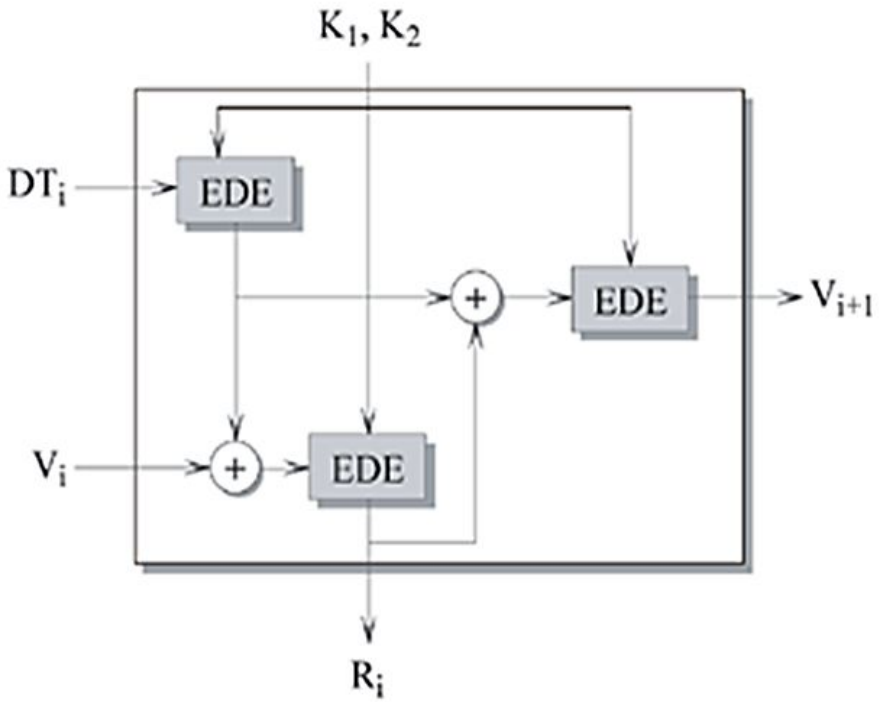


Схема включает использование 112-битного ключа и трех EDE-шифрований. На вход подаются два псевдослучайных значения: значение даты и времени и начальное значение очередной итерации, на выходе создаются начальное значение для следующей итерации и очередное псевдослучайное значение. Даже если псевдослучайное число R_i будет скомпрометировано, вычислить V_{i+1} из R_i невозможно, и, следовательно, следующее псевдослучайное значение R_{i+1}, так как для получения V_{i+1} дополнительно выполняются три операции EDE.

Домашнее задание по теме «Симметричная и асимметричная криптография»

1. Опишите, при каких условиях возможно получить новый открытый ключ (новый сертификат) для существующего закрытого ключа в случае с алгоритмом RSA. Почему этот процесс является нарушением принципов криптографии?
2. Опишите, с использованием каких алгоритмов (симметричных или ассиметричных) вы будете обеспечивать шифрование таблиц в БД? Определить режим работы алгоритма для шифрования данных в БД

Последовательность выполнения:

1. Создайте новый лист в Google Таблице и назовите его "Симметричная и асимметричная криптография".
2. Ответы на вопросы, указанные выше, разместите на листе в свободной форме.

NB! (ДЗ рекомендуется выполнять после вебинара «Асимметричные криптосистемы»)

ВОПРОСЫ

СПАСИБО ЗА ВНИМАНИЕ!

УСПЕШНОГО ОБУЧЕНИЯ!