Министерство образования Республики Беларусь
Учреждение образования
«Гомельский государственный технический университет имени П.О. Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

И.А. Мурашко

ЗАЩИТА КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ

Конспект лекций по одноименной дисциплине
для студентов направления специальности
1–40 01 02 –01 – Информационные системы и технологии
(в проектировании и производстве)
дневной и заочной формы обучения

Гомель 2014

СОДЕРЖАНИЕ

ТЕМА 1. КРИПТОГРАФИЯ: ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ 4

1.1. Основные термины 4
1.2. Простейшие шифры 5
1.3. Криптоанализ 6
1.4. Безопасность данных 6
1.5. Криптосистемы 7
1.6. Требования к секретности и достоверности 7
1.7. Классификация криптосистем 8

ТЕМА 2. МАТЕМАТИЧЕСКИЕ ОСНОВЫ КРИПТОГРАФИИ 9

2.1. Целые числа 9
2.2. Алгоритмы получения простых чисел 10
2.3. Составные числа 11
2.4. Алгоритм Евклида 11
2.5. Бинарный алгоритм 12
2.6. Сравнения 13
2.7. Свойства сравнений 13
2.8. Лемма 14
2.9. Алгоритм возведения в степень 14
2.10. Алгоритм быстрого возведения в степень 14
2.11. Понятие вычета 15
2.12. Теорема Ферма 15
2.13. Функция Эйлера 15
2.14. Теорема Эйлера 16
2.15. Линейные сравнения 16
2.16. Решение линейных сравнений 17
2.17. Элементы теории информации 18
2.18. Энтропия 18
2.19. Код Хаффмана 18
2.20. Частотный анализ 19
2.21. Абсолютная секретность 19
2.22. Одноразовый блокнот (шифратор Вермана) 20

ТЕМА 3. АЛГОРИТМЫ ШИФРОВАНИЯ 22

3.1. Простейшие алгоритмы шифрования 22
3.2. Метод говорящих часов 23
3.3. Полибианский квадрат 23
3.4. Шифратор Цезаря 23
3.5. Омофонные шифраторы 24
3.6. Шифратор Виженера 25
3.7. Роторная машина 26
3.8. Устройства роторной машины 26
3.9. Ключи роторных машин 27
3.10. Электронный ротор 28

3.11. Реализация механизма вращения 29
ТЕМА 4. СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ. КРИПТОГРАФИЧЕСКИЙ СТАНДАРТ *DES* 31
4.1. Шифратор LUCIFER 31
4.2. Алгоритм DES 32
4.3. Таблица перестановки DES 33
4.4. Функция F (подстановка + перестановка) 34
4.5. S–бокс 35
4.6. Вычисление ключа K_i 36
4.7. Декодирование (дешифрование) 37
4.8. Слабые ключи DES 37
4.9. Модификации DES 38
ТЕМА 5. КРИПТОГРАФИЧЕСКИЕ АЛГОРИТМЫ *IDEA, BLOWFISH, AES*... 39
5.1. Алгоритм шифрования данных IDEA 39
5.1.1. IDEA. Общие сведения 39
5.1.2. IDEA. Операции над данными 40
5.1.3. IDEA. Процедура шифрования 41
5.2. Симметричный блочный шифратор BLOWFISH 41
5.2.1. BLOWFISH. Общие сведения 41
5.2.2. Преимущества BLOWFISH 42
5.3. Стандарт AES 42
5.3.1. AES. Общие сведения 42
5.3.2. AES. Основные параметры 43
5.3.3. Группы, кольца и поля. Поля Галуа 43
5.3.4. AES. Математические операции 44
5.3.5. AES. Умножение полиномов 45
5.4. Потоковые криптосистемы 46
5.4.1. Потоковые шифры 46
5.4.2. Общая структура синхронного потокового шифратора 47
5.4.3. Генераторы криптографических ключей 48
5.4.3.1. Конгруэнтные генераторы 48
5.4.3.2. LFSR 48
5.4.3.3. Характеристические полиномы LFSR 50
ТЕМА 6. АСИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ 51
6.1. Криптографические системы с открытым ключом 51
6.2. Алгоритм RSA 51
6.3. Криптостойкость алгоритма RSA 53
ТЕМА 7. ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ 54
7.1. Функция хеширования 54
7.2 Электронная цифровая подпись 55
7.3 Классическая схема создания цифровой подписи 55
7.4 Алгоритм цифровой подписи RSA 55

ТЕМА 1. КРИПТОГРАФИЯ: ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Вопросы:

1.1. Криптография: основные термины
1.2. Простейшие шифры
1.3. Криптоанализ
1.4. Безопасность данных
1.5. Криптосистемы
1.6. Требования к секретности и достоверности
1.7. Классификация криптосистем

1.1. Основные термины

Проблемой защиты информации путем ее преобразования занимается криптология.

Выделяют 2 направления криптологии:

- криптографию
- криптоанализ

Криптография занимается поиском и исследованием методов преобразования информации с целью её шифрования.

Криптоанализ – область знаний, которая занимается исследованием возможности расшифрования информации без знаний криптографических ключей.

В качестве информации, которая подлежит шифрованию и дешифрованию, будем рассматривать исходные тексты, созданные при помощи некоторого алфавита.

Алфавит – конечное множество элементов (знаков), используемых для представления исходных текстов.

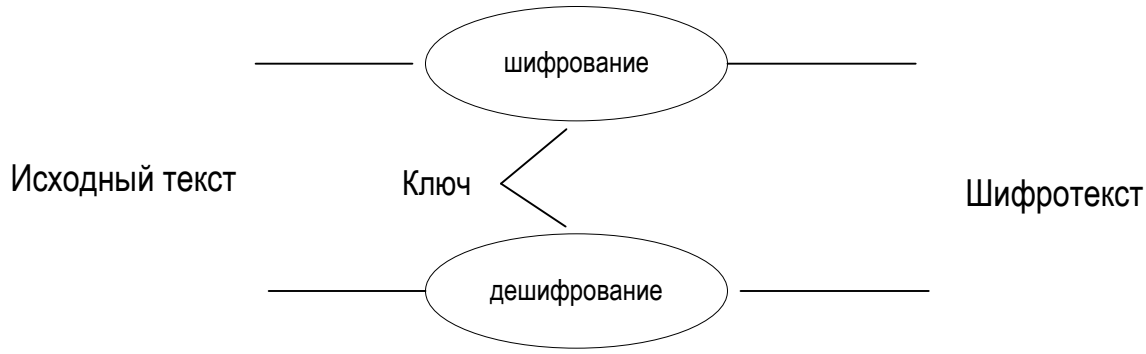
Исходный текст – упорядоченный набор из элементов алфавита.

Шифр – секретный метод записи, в соответствии с которым исходный текст преобразуется в шифротекст.

Шифрование (кодирование) – процесс преобразования исходного текста в шифротекст.

Дешифрование (декодирование) – обратный по отношению к шифрованию процесс.

Шифрование и дешифрование зависят от криптографического ключа, который выглядит следующим образом:



1.2. Простейшие шифры

Существует 2 вида шифров:

- перестановочные;
- подстановочные.

Перестановочные шифры выполняют перестановку символов в исходных данных.

Исходный текст ЭТО_ЛЕКЦИЯ при использовании шифра типа «железнодорожная изгородь» записывается:

Э				Л				И	
	Т		–		Е		Ц		Я
		О				К			

Читая по строкам слева направо, получаем шифротекст: ЭЛИТ_ЕЦЯОК.

Подстановочные шифры реализуют замену символов исходного текста на подстановочные элементы. В простейшем случае это символы исходного алфавита.

Например, сдвиг на 3 элемента – «шифр Цезаря».

$k = 3$

A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	…
X	*Y*	*Z*	*A*	*B*	*C*	*D*	*E*	…

Современные криптосистемы комбинирую процедуру подстановки и перестановки.

Код – специальный вид подстановочного шифра, который использует кодовую таблицу в качестве ключа.

		32	33	
25		север		
44			вчера	

1.3. Криптоанализ

Криптоанализ – это наука и практика, изучающая методы взлома шифра.

Шифр взломан, если возможно определить исходный текст или ключ от шифротекста, или определить ключ из пары: шифротекст – исходный текст.

Существует следующие **виды атак:**

1. атаки, использующие только шифротекст;
2. атаки с известным исходным текстом;
3. атаки с частичным текстом.

Атаки, использующие только шифротекст.

Криптоаналитик должен определить ключ только из перехваченного шифротекста. При этом возможно ему будут известны метод шифрования, язык исходного текста, предмет обсуждения и даже некоторые слова.

Атаки с известным исходным текстом.

Криптоаналитик знает несколько пар: шифротекст – исходный текст. Атаки с частичным текстом.

У криптоаналитика есть фрагменты исходного текста, соответствующие шифротексту. знает шифротекст соответствующий определённому исходному тексту.

Шифр безоговорочно безопасен (абсолютно секретен), если независимо от того, сколько шифротекста перехвачено, в нём всё равно недостаточно информации, чтобы однозначно определить исходный текст.

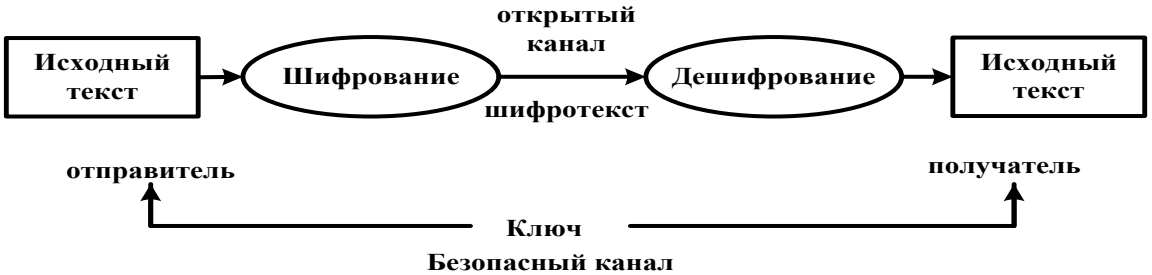
Шифр считается защищённым по вычислениям, если он не может быть взломан систематическим анализом доступных ресурсов.

1.4. Безопасность данных

Две **цели**, которые должны быть достигнуты при использовании криптографических систем:

1. скрытость (секретность) – предотвращение несанкционированного доступа к данным
2. достоверность (целостность) – предотвращение несанкционированного изменения данных.

Безопасный канал передачи данных представлен на рисунке:



Пассивный перехват – перехват сообщений без их изменений.

Активный перехват – умышленное изменение потока данных.

1.5. Криптосистемы

Криптосистемы состоят из **5 основных компонентов**:

1. М – пространство исходных текстов;
2. С – пространство шифротекстов;
3. К – пространство ключей;
4. Ек – пространство шифрующих преобразований (М → С);
5. *D*к – семейство дешифрующих преобразований (С → М).

Каждое шифрующее преобразование **Ек** определяется алгоритмом шифрования Е, общим для всех преобразований в семействе, и ключом К, отличающим его от других преобразований.

Аналогично и для ***D*к.**

Для заданного *К*: *Dк* – это взаимообратное преобразование к *Ек*.

$$D\text{к}\,(E\text{к}\,(M)) = M$$
$$E\text{к}\,(D\text{к}\,(C)) = C.$$

Основные требования к криптосистеме:

1. простота применения;
2. шифрующее и дешифрующее и преобразования должны быть корректны для любого ключа из пространства допустимых ключей;
3. безопасность системы должна зависеть только от секретности ключей, а не от секретности алгоритмов *E* и *D*.

1.6. Требования к секретности и достоверности

Требования к секретности:

1. для криптоаналитика должно быть вычислительно невозможно определение дешифрующего преобразования *D*к на основе перехваченного шифротекста даже, если соответствующий ему исходный текст известен.

2. для криптоаналитика должно быть вычислительно невозможно определение исходного текста из перехваченного шифротекста.

Требования к достоверности:

1. для криптоаналитика должно быть невозможно определение кодирующего преобразования Ек даже, если соответствующий ему исходный текст известен.

2. для криптоаналитика должно быть вычислительно невозможно нахождение шифротекста С’ такого, что *D*к (С’) является корректным исходным текстом.

Шифрование может осуществляться как программными, так и аппаратными средствами.

Аппаратная реализация.

- высокая скорость работы;
- высокая надежность;
- высокая стоимость.

Программная реализация:

- доступность;

- простота в использовании;
- дешевизна;
- низкая надежность.

Требования к криптосистемам:

1. зашифрованное сообщение должно поддаваться расшифровке только при наличии ключа.

2. число операций, необходимых для определения используемого ключа, должно быть не меньше, чем для общего числа возможных ключей.

3. число операций необходимых для подбора ключа методом перебора, должно иметь строгую нижнюю оценку и выходить за пределы возможностей современных вычислительных систем.

4. знание алгоритма шифрования не должно влиять на надёжность защиты.

5. незначительное изменение ключа должно приводить к существенным изменениям шифротекста.

6. дополнительные биты, вводимые в сообщение в процессе шифрования, должны быть надежно скрыты среди всего шифротекста.

7. длина шифротекста должна быть равна длине исходного текста.

8. любой ключ из множества допустимых ключей должен обеспечивать надёжную защиту информации.

9. алгоритм должен допускать как программную, так и аппаратную реализацию.

1.7. Классификация криптосистем

В соответствии с классификацией *Simmons* криптосистемы разделяют на:

- симметричные (одноключевые)
- ассиметричные (2 и более ключей)

В симметричных используется 1 ключ для шифрования и для расшифрования. Идеально подходит для шифрования личных данных.

Для создания баз данных с различными правами пользователя применяются многоключевые криптосистемы. Каждый пользователь имеет свои права доступа. Например, один пользователь может только писать данные в БД, другой – просматривать, третий – может делать и то, и другое.

ТЕМА 2. МАТЕМАТИЧЕСКИЕ ОСНОВЫ КРИПТОГРАФИИ

Вопросы:

2.1. Целые числа
2.2. Алгоритмы получения простых чисел
2.3. Составные числа
2.4. Алгоритм Евклида
2.5. Бинарный алгоритм
2.6. Сравнения
2.7. Свойства сравнений
2.8. Лемма
2.9. Алгоритм возведения в степень
2.10. Алгоритм быстрого возведения в степень
2.11. Понятие вычета
2.12. Теорема Ферма
2.13. Функция Эйлера
2.14. Теорема Эйлера
2.15. Линейные сравнения
2.16. Решение линейных сравнений
2.17. Элементы теории информации
2.18. Энтропия
2.19. Код Хаффмана
2.20. Частотный анализ
2.21. Абсолютная секретность
2.22. Одноразовый блокнот (шифратор Вермана)

2.1. Целые числа

В криптографии используются целые числа.

Целые числа делятся на:

- простые;
- составные.

Целое число d является делителем n, тогда и только тогда, когда можно записать произведение $\boldsymbol{n = k \cdot d}$ и записывается $d \mid n$.

Целое число p ($p \geq 1$) является простым, если его делителями являются только 1 и p.

Например: 2, 3, 5, 7, 11, 13, 17…и т.д.

Числа, имеющие более 2–х делителей являются составными.

Любое целое число n>1 может быть представлено единственным образом как произведение простых чисел в соответствующих степенях. Такое представление называется **каноническим разложением Евклида**.

Процедура получения такого разложения называется разложением на простые сомножители или **факторизацией числа**.

На настоящий момент не известны быстрые алгоритмы разложения чисел. На этом основаны большинство современных криптосистем.

Получение простых чисел является одной из основных задач криптографии.

Евклид показал, что существует бесконечное множество простых чисел, однако их удельный вес среди других чисел невелик.

Число простых чисел на сотню.

Кол–во	**1 –1000**
25	1–100
21	101–200
16	201–300
16	301–400
17	401–500
14	501–600
16	601–700
14	701–800
15	801–900
14	901–1000

Кол–во	**10 000 001–10 001 000**
2	10 000 001–10 000 100
6	10 000 101–10 000 200
6	10 000 201–10 000 300
6	10 000 301–10 000 400
5	10 000 401–10 000 500
4	10 000 501–10 000 600
7	10 000 601–10 000 700
10	10 000 701–10 000 800
9	10 000 801–10 000 900
6	10 000 901–10 001 000

2.2. Алгоритмы получения простых чисел

Один из первых алгоритмов был предложен во времена Евклида. Базируется на следующих теоремах:

Теорема 2.1.

Если целое число n (n >1) не делится ни на одно простое число не большее, чем $\sqrt{n}$, то это число есть простое число.

Тогда алгоритм получения простых чисел основано на теореме 2.5.

Теорема 2.2. (Эратосфена)

1) если в наборе целых ччисел 2, 3, 4… N удалить все числа, которые делятся на первые P простых чисел, тогда первое (наименьшее) неудалённое число будет простым.

2) если в наборе целых чисел 2, 3, 4…N удалить все числа, которые делятся на простые числа $\leq \sqrt{N}$, тогда все оставшиеся числа будут простыми числами, принадлежащими интервалу $]\sqrt{N}, N]$.

Этот метод называется **решето Эратосфена.**

Пример:

2, 3, 4, 5, 6, 7, 8, 9, 10.

$\sqrt{10} \approx 3$

2, 3 < $\sqrt{10}$ => вычеркиваем те числа, которые делятся на 2 и 3. Все оставшиеся после вычеркивания числа – простые.

Попытка формальным образом определить простые числа привела к отрицательному результату.

Простые числа Эйлера могут быть найдены по следующей формуле:

$$x^2 - x + 41 \qquad 0<x<40$$

Например:

$x = 7$ => 49–7+41 = 83 – простое число

Простые числа Ферма могут быть вычислены по формуле:

$$2^k + 1, \quad k = 2^r \qquad r = 0, 1, 2\ldots..$$

Простые числа Мерсенна могут быть найдены по формуле:

$$2^n - 1 \quad n \text{ – простое число.}$$

44–ое простое число найдено 4 сентября 2006 года и содержит 9808358 десятичных цифр. Это $2^{32582657} - 1$.

2.3. Составные числа

Составные числа могут быть представлены в канонической форме:

$$a = \prod_{i=1}^{\kappa} p_i^{\alpha_i}$$

где p_i – простые числа;

α_i – целый числа

Например:

$120 = 2^3 \cdot 3^1 \cdot 5^1$

Общим делителем чисел $a_1, a_2, a_3 \ldots a_N$ является целое число d такое, что $d|a_1$, $d|a_2$, …. $d|a_N$

Наибольший общий делитель (НОД) чисел $a_1, a_2, a_3 \ldots a_N$ – это наибольший делитель d_1, который может быть поделён любым делителем этих чисел.

Пример: НОД (6, 15, 27) = 3

2.4. Алгоритм Евклида

Проблема нахождения НОД является одной из наиболее важных в криптографии.

В частности НОД определяет взаимную простоту чисел.

$(a_1, a_2) = 1$ => эта запись означает, что a_1, a_2 – взаимно простые числа.

Например:

(3, 5) = 1, (3, 10) = 1, однако (3, 6) ≠ 1

Исторически одним из первых инструментов определения взаимно простых чисел стал **алгоритм Евклида**.

Он основан на следующей теореме: если $\boldsymbol{a = b \cdot q + r}$, тогда НОД чисел *a* и *b* равен НОД *b* и *r*.

(*a*, *b*) = (*b*, *r*)

Код алгоритма:

```
begin
    i:=1; g0:=a; g1:=b;
    while gi≠0 do
    begin
      gi+1:= gi–1 mod gi;
            i:= i+1;
    end;
            gcd:= gi–1;       // gcd – НОД
end;
```

Пример:

(1173, 323) = ?

1173 = 323*3 + 204

НОД (1173, 323) = (323, 204)

323 = 204*1 + 119

НОД (323, 204) = (204, 119)

204 = 119*1 + 85

НОД (204, 119) = (119, 85)

119 = 85*1 + 34

НОД (119, 85) = (85, 34)

85 = 34*2 + 17

НОД (85, 34) = (34, 17)

34 = 17*2 + 0

НОД (1173, 323) = 17

Одним из существенных недостатков алгоритма Евклида является применение операций деления. Это увеличивает его вычислительную сложность.

2.5. Бинарный алгоритм

Бинарный алгоритм базируется на **4 утверждениях**:

1. Если оба числа чётные, то НОД (*a*, *b*) = 2·НОД (a/2, *b*/2);
2. Если *a* – четное, *b* – нечетное, то НОД (*a*, *b*) = НОД (*a*/2, *b*);
3. НОД *(a, b)* = НОД *(b, a– b);*
4. если *a* и *b* – нечётные числа, то *a– b* является чётным числом.

Пример:

(1173, 323) = ?

(1173, 323) = (323, 850) = (323, 425) = (323, 102) = (323, 51) = (51, 272) = = (51, 136) = (51, 68) = (51, 34) = (51, 17) = (17, 34) = (17, 17) = 17.

Числа $a_1, a_2, a_3, \ldots a_n$ называются **взаимно простыми**, если их НОД = 1 $(a_1, a_2, \ldots a_n) = 1$

Числа $a_1, a_2, a_3, \ldots a_n$ называются **попарно взаимно простыми**, если для любого i, j НОД $(a_i, a_j) = 1$ $i \neq j$

Теорема:

Если $(a, b) = 1$, то для любых целых чисел n, m $\boldsymbol{(a^n, b^m) = 1}$.

Справедливо и обратное утверждение, если $(a^n, b^m) = 1$, то для любых чисел n, m $\boldsymbol{(a, b) = 1}$.

2.6. Сравнения

Два целых числа a и b сравнимы (конгруэнтны) по модулю натурального числа m, если при делении на m они дают одинаковые остатки.

Пример:

32 и 39 сравнимы по модулю 7

$32 = 4 \cdot 7 + 4$

$39 = 5 \cdot 7 + 4$

Записывается в следующем виде: $a \equiv b \bmod m$

$32 \equiv 39 \bmod 7$

Числа a и b **сравнимы по модулю m**, если их разность $a–b$ делится на m.

Числа a и b **не сравнимы по модулю m**, если их разность не делится на m.

Если $a = b \bmod m$, и $b < m$, то b называется **вычетом** a по модулю m.

2.7. Свойства сравнений

1. Если $a = b \bmod m$, то для любого целого k будет справедливо:
$\boldsymbol{ka = kb \bmod m}$

2. Если $ka = kb \bmod m$ и НОД $(k, m) = 1$, то $a = b \bmod m$

3. Если $ka = kb \bmod km$, то $a = b \bmod m$

4. Если $a = b \bmod m$ и $c = d \bmod m$, то $\boldsymbol{a + c = (b + d) \bmod m}$

5. Если $a_1 = b_1 \bmod m$, $a_2 = b_2 \bmod m$ и т.д., то
$\boldsymbol{a_1 + a_2 + \ldots + a_n = (b_1 + b_2 + b_n) \bmod m}$

6. Если $a = b \bmod m$ и $c = d \bmod m$, то $\boldsymbol{a*c = b*d \bmod m}$

7. Если $a = b \bmod m$, то для любого целого k $\boldsymbol{a^k = b^k \bmod m}$

2.8. Лемма

Лемма:

(a*b) mod m = [(a mod m)*(b mod m)] mod m,

где * – любая из следующих операций: сложение (+), вычитание (–), умножение (·).

Применение данного выражения существенно упрощает вычисление.

Пример:

(7·9) mod 5 = [(7 mod 5) · (9 mod 5)] mod 5 = [2·4] mod 5 = 3

2.9. Алгоритм возведения в степень

Пример: 3^5 *mod* 7.

Алгоритмы:

1. 3^5 *mod* 7 = 243 *mod* 7 = 34*7 + 5 = 5
2. В общем случае $\boldsymbol{a^z \bmod m}$ необходимо выполнить *z*–1 операцию умножений и 1 операцию деления.

   Алгоритм можно ускорить, если $3^5 = (3^2)^2 * 3$

Недостаток первых 2–х алгоритмов – большие промежуточные результаты.

3. На основе леммы ***(a*b) mod m = [(a mod m)*(b mod m)] mod m***) 3^5 *mod* 7 = $(3^2 \cdot 3^2 \cdot 3)$ *mod* 7 = [(3^2 *mod* 7)·(3^2 *mod* 7) ·3] *mod* 7 =

   = 2·2·3 *mod* 7 = 12 *mod* 7 = 5

2.10. Алгоритм быстрого возведения в степень

Алгоритм возвращает значение $x = a^z \bmod m$

```
begin
a1:=a;
z1:=z;
x:=1;
while z1≠0 do
begin
     while z1 mod 2 = 0 do
         begin
         z1:=z1div2;
         a1:=(a1·a1) mod m;
         end;
     z1:=z1–1;
     x1:=(x·a1) mod m;
end;
result:=x;
end.
```

*2.*11. Понятие вычета

Вычетом числа *a* по модулю *m* называют остаток от деления величины *a* на *m*.

Вычет и модуль в некоторых случаях совпадают.

r – вычет числа *a* по модулю *m*, $\mathbf{0 < r < m}$

Если набор из *m* целых чисел $\{a_i\}=\{a_1,a_2,\ldots,a_m\}$ формирует полный набор чисел, сравнимых по модулю *m* с каждым значением r_i $r_i = \{0, 1, 2,\ldots m–1\}$, то набор r_i называется полной системой вычетов по модулю *m*.

Пример:

{16, 12, 19, 48, 65} – полная система вычетов по модулю 5.

2.12. Теорема Ферма

Теорема: если *p* – простое число и a, p – взаимно простые числа *((a, p) = 1)*, то $\boldsymbol{a^{p-1} = 1 \bmod p}$.

Теорема Ферма справедлива только для простых p.

Обобщением данной теоремы является теорема Эйлера, которая основана на использовании функции Эйлера.

2.13. Функция Эйлера

Функцией Эйлера φ(*n*) (либо ψ(*n*), Φ(*n*)) целого числа $n \geq 1$ является количество целых чисел, которые меньше, чем *n* и взаимно просты с *n*.

n	1	2	3	4	5	6	7	8	9
φ(*n*)	0	1	2	2	4	2	6	4	6

φ(p) = $p - 1$ – функция Эйлера для простого числа

Если $n = p \cdot q$, где *p* и *q* – простые числа, то $\boldsymbol{\varphi(n) = \varphi(p) \cdot \varphi(q) = (p–1)\cdot(q–1)}$

Если *p* – простое число и $k > 0$ – целое число, то

$\boldsymbol{\varphi(p^k) = p^k – p^{k–1} = p^{k–1}(p–1)}$

Для произвольного случая необходимо представить целое число a в виде канонического разложения Евклида: $a = p_1^{\alpha 1} \cdot p_2^{\alpha 2} \cdot \ldots \cdot p_r^{\alpha r}$, тогда

$\boldsymbol{\varphi(a) = p_1^{\alpha 1}(1 – 1/p_1) \cdot p_2^{\alpha 2}(1 – 1/p_2) \cdot \ldots \cdot p_r^{\alpha r}(1 – 1/p_r)}$

2.14. Теорема Эйлера

Теорема: если $n \geq 0$ – целое число и НОД $(a, n) = 1$, то $\mathbf{a^{\varphi(n)} = 1 \ mod \ n}$

Пример:

$3^{12} \ mod \ 26 = ?$

$n = 26$

$\varphi(n) = \varphi(26) = \varphi(2 \cdot 13) = \varphi(2) \cdot \varphi(13) = 1 \cdot 12 = 12$

$3 \ \varphi^{(26)} = 1 \ mod \ 26$

2.15. Линейные сравнения

Под **линейным сравнением** понимают выражение вида $\boldsymbol{ax = b \ mod \ n}$, где a, b, n – целые числа, $b < n$.

Существует 3 возможных случая для линейного сравнения:

1. линейное сравнение не имеет решения
2. имеет 1 решение
3. имеет множество решений

1 случай:

Если НОД $(a, n) = d$ не является делителем b, то линейное сравнение $ax = b \ mod \ n$ не имеет решения.

Пример:

$2x = 1 \ mod \ 4$

$d = (2, 4) = 2 \Rightarrow$ 2 не является делителем 1 $\Rightarrow$ нет решения.

2 случай:

Если НОД $(a, n) = 1$, т.е. a и n являются взаимно простыми числами, то линейное сравнение $ax = b \ mod \ n$ имеет 1 решение.

Пример:

$2x = 1 \ mod \ 3$

$(2,3) = 1$

x = 2, т.к. $2*2 = 1 \ mod \ 3$

3 случай:

Если НОД $(a, n) = d$ и d является делителем b, то линейное сравнение $ax = b \ mod \ n$ имеет d решений.

Пример:

$6x = 4 \ mod \ 10$

$(6,10) = 2$

$2 \mid 4$

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

$6*4 = 4 \ mod \ 10$ и $6*9 = 4 \ mod \ 10$

2.16. Решение линейных сравнений

Линейное сравнение имеет 1 решение, когда:

1. b = 1 => $ax = 1 \bmod n$ $\quad x = a^{-1}$

x является мультипликативной инверсной величиной по отношению к *a*.

Для нее истина: $a \cdot a^{-1} = 1$

a^{-1} – взаимообратное по отношению к *a*.

Для этого случая $x = a^{\varphi(n)-1} \bmod n$

Пример:

$3x = 1 \bmod 7$,

$x = 3^{\varphi(7)-1} \bmod 7 = 3^5$

$x = 3^{7-2} \bmod 7 = 3^5 \bmod 7 = [(3^2)^2 \cdot 3] \bmod 7 = 4 \cdot 3 \bmod 7 = 5$

Если n – простое число, то $x = a^{n-2} \bmod n$

Пример:

$4x = 1 \bmod 9$

$x = 4^{\varphi(9)-1} \bmod 9 = 4^{6-1} \bmod 9 = 4^5 \bmod 9 = (4^2)^2 \cdot 4 \bmod 9 = 7 \cdot 7 \cdot 4 \bmod 9 = 7 \cdot 28 \bmod 9 = 7 \cdot$

2. b ≠ 1 $\quad x = b \cdot a^{\varphi(n)-1} \bmod n$

Пример:

$3x = 3 \bmod 7$

$(3, 7) = 1$

$x = 3 \cdot 3^{\varphi(7)-1} \bmod 7 = 3 \cdot 3^5 \bmod 7 = 3^6 \bmod 7 = (3^2)^3 \bmod 7 = 2^3 \bmod 7 = 1$

$3 \bmod 7 = 3$

Линейное сравнение имеет множество решений:

$(a, n) = d$, *d* является делителем *b* $d \mid b$

$ax = b \bmod n$ имеет *d* решений

$x_0 = b\, a^{\varphi(n)-1} \bmod n$

$$x_1 = x_0 + \frac{n}{d} \bmod n$$

$$x_{d-1} = x_0 + \frac{(d-1)n}{d} \bmod n$$

Пример:

$6x = 4 \bmod 10$

$(6, 10) = 2$, $2|4$

$3x = 2 \bmod 5$

$x_o = 2 \cdot 3^{5-2} \bmod 5 = 2 \cdot 3^3 \bmod 5 = 2 \cdot 3 \cdot 3^2 \bmod 5 = 4$

$x_1 = (x_o + n/d) \bmod 10 = (4 + 10/2) \bmod 10 = 9$

Проверка:

$6 \cdot 9 \bmod 10 = 54 \bmod 10 = 4$

$6 \cdot 4 \bmod 10 = 24 \bmod 10 = 4$

2.17. Элементы теории информации

Неоценимый вклад в криптографию внес основоположник теории информации – Клод Шеннон.

В 1949 г. он опубликовал свои теоретические исследования по криптографии.

Теория информации измеряет количество информации в сообщении по среднему количеству бит, необходимых для оптимального кодирования всех возможных сообщений.

Количество информации в сообщении измеряется **энтропией** сообщения, которая базируется на понятии количества информации.

Пусть x_1, x_2, …. x_n – это n возможных сообщений, возникающих с вероятностью $p(x_1)$, $p(x_2)$, …. $p(x_n)$, тогда получение сообщения x_i можно оценить количеством полученной информации, которое вычисляется как:

$$F(x_i) = -\log_2 p(x_i) = \log_2 \frac{1}{p(x_i)}$$

Маловероятное событие несет много информации, и наоборот, событие, вероятность которого равна 1, не несет информации.

$p(x_i) = 1$

$log_2\ 1/1 = log_2\ 1 = 0$.

2.18. Энтропия

Энтропия – среднее количество информации при получении одного из возможных сообщений.

Вычисляется по формуле:

$$H(x) = -\sum_{i-1}^{n} p(x_i)\log_2 p(x_i) = \sum_{i=1}^{n} p(x_i)\log_2 \frac{1}{p(x_i)}.$$

Для оптимального кодирования используют короткие коды для часто встречающихся сообщений и длинные коды – для редко встречающихся сообщений. Этот принцип применен в коде Морзе.

2.19. Код Хаффмана

Код Хаффмана является оптимальным кодом, ассоциированным с буквами, словами, машинными инструкциями или фразами.

Односимвольный код Хаффмана часто используется для минимизации больших файлов.

2.20. Частотный анализ

Достижения в теории информации позволили формальным образом исследовать исходные тексты, представленные на конкретном языке, и использовать эти результаты для взлома криптосистем.

Одним из таких методов анализа является **частотный анализ.**

Его **суть:** распределение букв в криптотексте сравнивается с распределением букв в алфавите исходного сообщения.

Вероятность успешного вскрытия повышается с увеличением длины криптотекста.

Существует множество различных таблиц распределения букв.

Английский алфавит		Русский алфавит	
буква	частота встречаемости	буква	частота встречаемости
E	0,1251	О	0,109
T	0,0925	Е	0,087
A	0,0804	А	0,075
O	0,0726	И	0,075
I	0,0726	Н	0,064
N	0,0709	Т	0,064
S	0,0654	С	0,055
R	0,0612	Р	0,048
H	0,055	В	0,046

В английском алфавите часто встречаются биграммы: *TH, EN*.

Никогда в осмысленных сообщениях не встречается биграмма OZ.

Доля имеющих смысл последовательностей букв любого языка понижается с увеличением этой последовательности. Например, в русском языке часто встречается ЕЕ, три ЕЕЕ встречаются только в 2–х словах (длинношеее, змееед).

2.21. Абсолютная секретность

Основным вкладом Клода Шеннона в теорию и практику защиты информации явилось его обоснование возможности создания абсолютных секретных криптосистем.

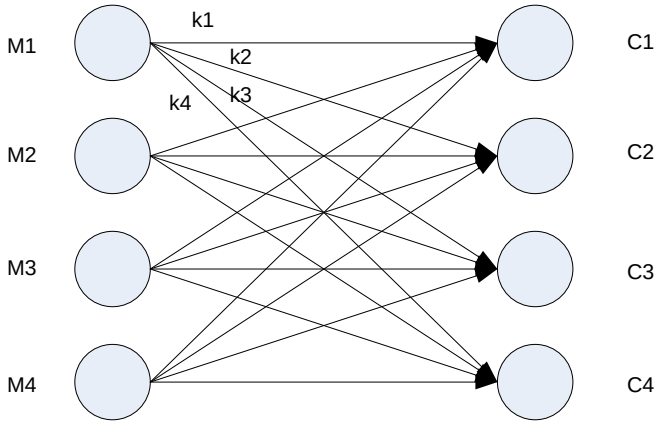
Он выделил **3 вида информации:**

1. исходное сообщение *M*, описываемое вероятностью появления $p(M)$, где сумма всех вероятностей равна 1.
2. зашифрованное сообщение (криптотекст *C*), описываемое вероятностью p (*C*), где сумма всех вероятностей равна 1.

3. ключи К, описываемые вероятностью использования *p(K),* где сумма всех вероятностей равна 1.

Если *Pc(M)* есть вероятность тог, что на основании исходного сообщения *M* было получено зашифрованное сообщение *C*, тогда абсолютная секретность будет достигнута тогда и только тогда, когда будет выполнятся: ***Pc(M) = P(M)***

Абсолютная секретность криптосистемы



$Pc\ (M) = P\ (M) = ¼$
$Pc\ (C) = P\ (C) = ¼$
$P\ (K_i) = ¼$

Криптоаналитик, перехватив одно из зашифрованных сообщений, не сможет определить, какой из исходных текстов был зашифрован.

Абсолютная секретность требует, чтобы количество ключей было, по крайней мере, такое же, сколько возможно исходных сообщений. В противном случае, если ключей меньше, чем число возможных исходных сообщений, криптоаналитик сможет получить некоторую дополнительную информацию.

2.22. Одноразовый блокнот (шифратор Вермана)

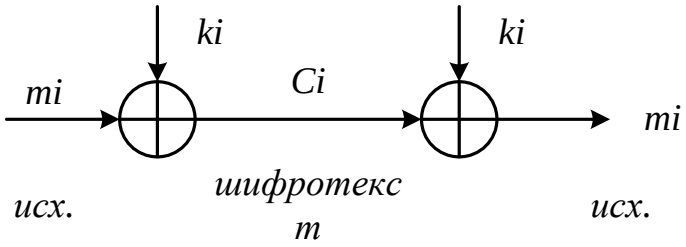
Шифр, использующий неповторяющуюся случайную последовательность элементов ключа, при его размерности, равной размерности исходного сообщения называется системой шифрования типа одноразовый блокнот.

Одноразовый блокнот – единственный шифр, позволяющий достичь абсолютной секретности.

Его реализация была впервые предложена Верманом в 1917 г.

В качестве ключа может быть использована детерминированная последовательность символов, полученная из некоторого литературного источника (например,

Библия, Конституция США и т.д.) либо при помощи генераторов псевдослучайных чисел.



m_i – символ исходного текста;
k_i – символ ключа;
C_i – символ шифротекста.
$\oplus$ – операция суммирования по модулю 2.
$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$
Нарушение синхронизации приводит к потере сообщения.

ТЕМА 3. АЛГОРИТМЫ ШИФРОВАНИЯ

Вопросы:

3.1. Простейшие алгоритмы шифрования
3.2. Метод говорящих часов
3.3. Полибианский квадрат
3.4. Шифратор Цезаря
3.5. Омофонные шифраторы
3.6. Шифратор Виженера
3.7. Роторная машина
3.8. Устройства роторной машины
3.9. Ключи роторных машин
3.10. Электронный ротор
3.11. Реализация механизма вращения

3.1. Простейшие алгоритмы шифрования

Выделяют 2 типа простейших шифров:

1. подстановочные
2. перестановочные

Подстановочный шифр ставит в соответствие одному символу исходного текста один символ шифротекста. Как правило, алфавиты исходного текста и шифротекста совпадают.

Перестановочные шифры меняют позиции символов исходного текста.

В общем виде:

М —(**Правила записи**)— **Фигура** —(**Правила чтения**)— **С**

Дешифрование:

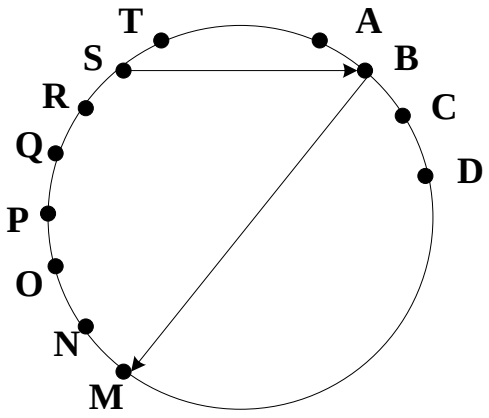
М —(**Правила чтения**)— **Фигура** —(**Правила записи**)— **С**

3.2. Метод говорящих часов

Метод говорящих часов был предложен *Angie Wimer*.

Суть метода:

Алфавит исходного текста записывается по окружности. Каждый символ соединяется двумя другими случайно выбранными хордами: одной входящей, одной выходящей.



3.3. Полибианский квадрат

Изобретен во *II* в. до н.э. Полибием.

Суть метода:

В таблица 5x5 случайном образом заносятся символы греческого алфавита (24 буквы). При шифровании каждая буква заменяется на букву, которая находится в том же столбце, только на строку ниже (для последней строки – верхняя строка).

α	ρ	κ	θ	φ
β	χ	ω	π	υ
δ	ψ	ο	γ	η
ζ	μ	ε	τ	σ
ν	ξ	ϖ		

3.4. Шифратор Цезаря

Шифратор использовался Цезарем в переписке с Цицероном около 50 лет до н.э.

Таблица подстановки присутствует в неявном виде, т.е. символ шифротекста вычисляется по математическому выражению:

$$C_i = (a_i + k) \bmod n$$

где a_i – символ исходного текста;

k – ключ;

n – мощность алфавита.

В методе Цезаря используется $k = 3$.

Развитием этого метода является метод, основанный на свойстве децимации.

$$C_i = (a_i + k) \bmod n$$

Децимация – выборка k–тых элементов.

Номера символов в шифротексте в k раз больше номеров символов исходного текста, где номер относится к алфавиту, а не исходному тексту.

0	1	2	3
A	*B*	*C*	*D*

0	3	6	9
A	*D*	*F*	*G*

Аффинное преобразование: $C_i = (a_i k_1 + k_2) \bmod n$

Используется 2 ключа: k_1 и k_2. Накладывается требование взаимной простоты $(k_1, n) = 1$

3.5. Омофонные шифраторы

Подстановочные шифры используют взаимно–однозначное соответствие, т.е. определенному символу исходного текста соответствует конкретный символ шифротекста. Это делает их уязвимыми для атак типа частотный анализ.

Система омофонов обеспечивает простейшую защиту от таких атак.

Суть омофонных шифраторов:

Каждый символ исходного текста имеет несколько подстановочных элементов. Процедура выбора подстановочного элемента должна выполняться случайно.

Пример:

Каждый символ кодируем двузначным числом в диапазоне 00 – 99.

A: 23, 25, 97, 89, 33, 11

B: 87, 41

C: 44, 77, 35, 51

При шифровании случайным образом выбираем один из вариантов:

C	*A*	*B*	или	*C*	*A*	*B*
77	89	87		51	25	41

3.6. Шифратор Виженера

Система шифрования Виженера впервые была опубликована в 1586 г. и является одной из наиболее известных многоалфавитных систем шифрования.

Особенностью шифратора Виженера является использование различных таблиц подстановки в зависимости от последовательности символов используемого ключа.

Основой шифра является таблица 9квадрат Виженера).

Строки и столбцы – алфавит. 1–ая строка – это исходный алфавит, каждая следующая строка – циклический сдвиг алфавита на 1 позицию влево.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
b	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
c	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
d	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
e	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
f	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
g	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
h	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
i	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
j	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
k	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
l	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
m	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
n	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
o	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
p	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
q	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
r	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
s	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
t	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
u	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
v	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
w	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
x	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
y	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
z	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Шифрование:

Исходное сообщения его записывают в строку, а под ним ключевое слово или фразу. Если ключ короче, чем сообщение, то его дублируют.

На каждом шаге шифрования в верхней строке таблицы находят очередную букву исходного сообщения, а в левом столбце – очередной символ ключа. В результате символ шифротекста будет находиться на пересечении данной строки и данного столбца.

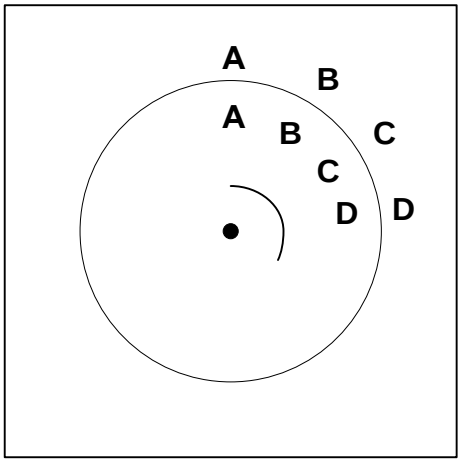
Пример*:*

M:	*C*	*R*	*Y*	*P*	*T*	*O*	*G*	*R*	*A*	*P*	*H*	*Y*
Ключ*:*	*A*	*B*	*R*	*A*	……							
C:	*C*	*S*	*P*	*P*	……..							

3.7. Роторная машина

Основная идея роторных машин лежит в многократных либо многоалфавитных заменах.

В 1568 г. Олберти издал рукопись, описывающую шифрующий диск, который определял многократные замены.



Поворот диска на один шаг меняет таблицу подстановки.

Важным развитием данного метода является динамическое изменение таблиц подстановки в процессе шифрования.

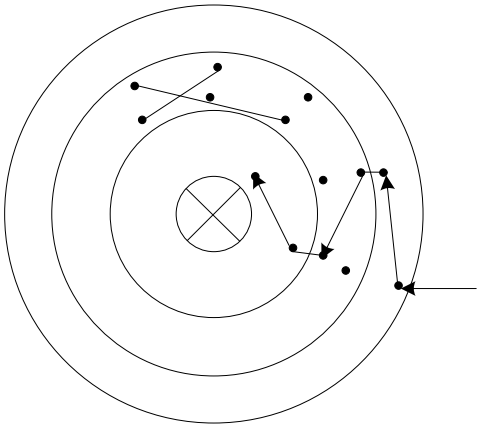
Машины шифрования:

1. *SICABA* (*M* – 134) – американская
2. *TYPEX* – английская
3. *PURPLE* – японская
4. *ENIGMA* – немецкая, самая распространенная

3.8. Устройства роторной машины

Главной деталью роторной машины является **ротор**, на внешней и внутренней стороне которого расположено по *n* контактов.

n – число символов исходного алфавита.



Каждый контакт на внешней стороне соединен с одним из контактов на внутренней стороне. Соединение определяет таблицу подстановки. Соединение может быть изменено при помощи перемычек.

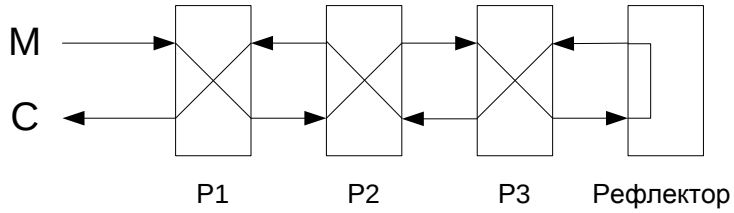
Роторная машина состоит из нескольких роторов и механизма их вращения.

После шифрования очередного символа происходит изменение положения роторов. Когда диск совершит полный круг, на одну позицию переместится следующий за ним и т.д. таким образом, для каждого символа исходного текста будет использоваться своя таблица подстановки.

Число таблиц подстановки для *m* роторов и алфавита из *n* символов равно $\boldsymbol{n^m}$**.**

Учитывая, что для английского алфавита число всевозможных таблиц подстановки ≈ 26!, то число роторов не превышает 10.

Для упрощения реализации роторных машин как правило один ротор дважды используется при шифровании.



M – исходный текст, C – шифротекст

При наличии трех физических роторов шифрование осуществляется шестью.

3.9. Ключи роторных машин

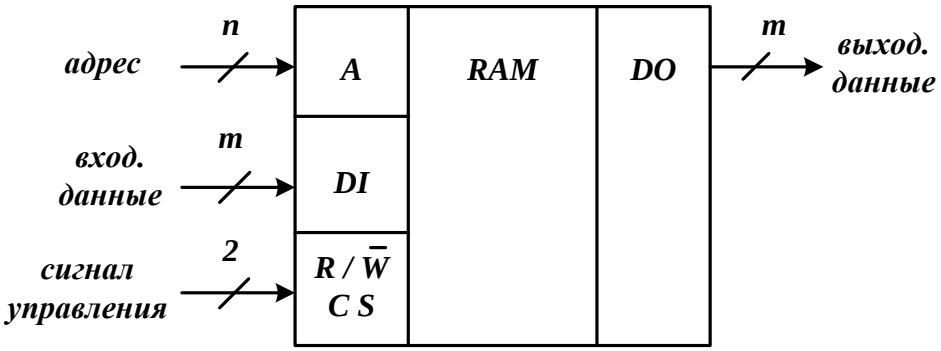
Ключом роторной машины может быть ее различные параметры, такие как:

- число роторов;
- порядок подключения роторов;
- коммутация каждого ротора;
- механизм перемещения ротора.

3.10. Электронный ротор

Развитием классической роторной машины стала ее электронная версия. Основным компонентом этой версии является запоминающее устройство, которое выполняет функцию ротора.

Схема запоминающего устройства



R/*W* – операция с памятью:

1 – чтение данных;

0 – запись данных.

CS – выбор кристаллов, разрешает работу данного АЗУ.

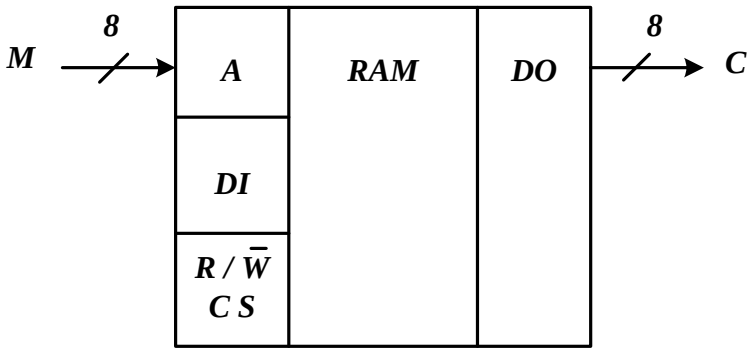
Пример:

Рассмотрим АЗУ, для которого $n = m = 8$

АЗУ способно хранить 256x8 бит.

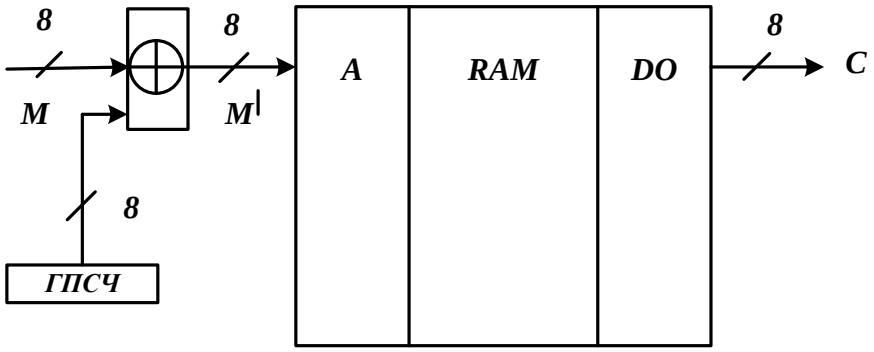
адрес	данные
0	01010101
1	11110000
2	01111000
…	
255	11111100

Содержимое АЗУ определяет таблицу подстановки.



Чтобы применять таблицу подстановки, необходимо занести новую информацию в память.

3.11. Реализация механизма вращения



ГСПЧ – генератор псевдослучайных чисел.

Очередной символ исходного текста поразрядно суммируется по модулю 2 с очередным псевдослучайным числом, которое формирует ГПСЧ.

Следующий символ исходного текста суммируется со следующим псевдослучайным числом и т.д. Этим обеспечивается механизм вращения ротора.

Пример:

Шифруем три символа А, идущие подряд.

Код символа А – 00000000.

В 3–х соседних тактах ГПСЧ формируются следующие значения:

11111111

01111111

00111111

AAA

Первый символ А суммируется поразрядно с 1–ым псевдослучайным числом.

$0 \oplus 0 = 0$

$0 \oplus 1 = 1$

$1 \oplus 0 = 1$

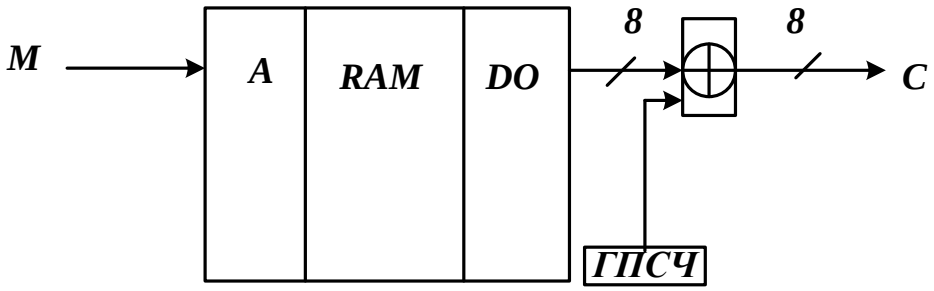
$1 \oplus 1 = 0$

1–ый символ А преобразовался в 11111111.

2–ой символ А: 01111111

3–ий символ А: 00111111

На следующем рисунке представлен еще один механизм реализации вращения.



Отличие от предыдущего механизма заключается в том. Что суммирование с очередным псевдослучайным числом происходит символа шифротекста, а не символа исходного текста.

Достоинства роторных машин:

1. высокая криптостойкость;
2. высокое быстродействие;
3. простота как программной, так и аппаратной реализации.

ТЕМА 4. СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ. КРИПТОГРАФИЧЕСКИЙ СТАНДАРТ *DES*

Вопросы:

4.1. Шифратор *LUCIFER*
4.2. Алгоритм *DES*
4.3. Таблица перестановки *DES*
4.4. Функция *F* (подстановка + перестановка)
4.5. *S*–бокс
4.6. Вычисление ключа K_i
4.7. Декодирование (дешифрование)
4.8. Слабые ключи *DES*
4.9. Модификации *DES*

4.1. Шифратор *LUCIFER*

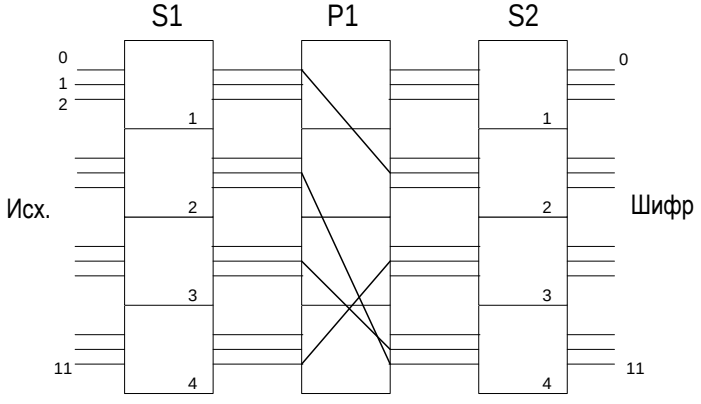
В настоящее время одним из самых распространенных является стандарт шифрования *DES*, основанный на простейших операций подстановки и перестановки (эта идея принадлежит Клоду Шеннону).

Впервые идея реализована в шифраторе *LUCIFER*, созданного на фирме *IBM Feistl*.

Шифратор *LUCIFER* преобразует исходное сообщение, попеременно применяя процедуру подстановки и перестановки.

На входе шифратора исходный текст 12 бит. Сначала он разбивается на 4 блока по 3 бита, затем при помощи таблицы подстановки каждому 3–хбитному блоку ставится в соответствие некоторый другой 3–хбитный символ. После чего при помощи процедуры перестановки перемешивает все 12 бит. Затем снова получают 4 блока по 3 бита и снова используют таблицу подстановки.

Схема шифрования



*S*1 – подстановка

*P*1 – перестановка

*S*2 – подстановка

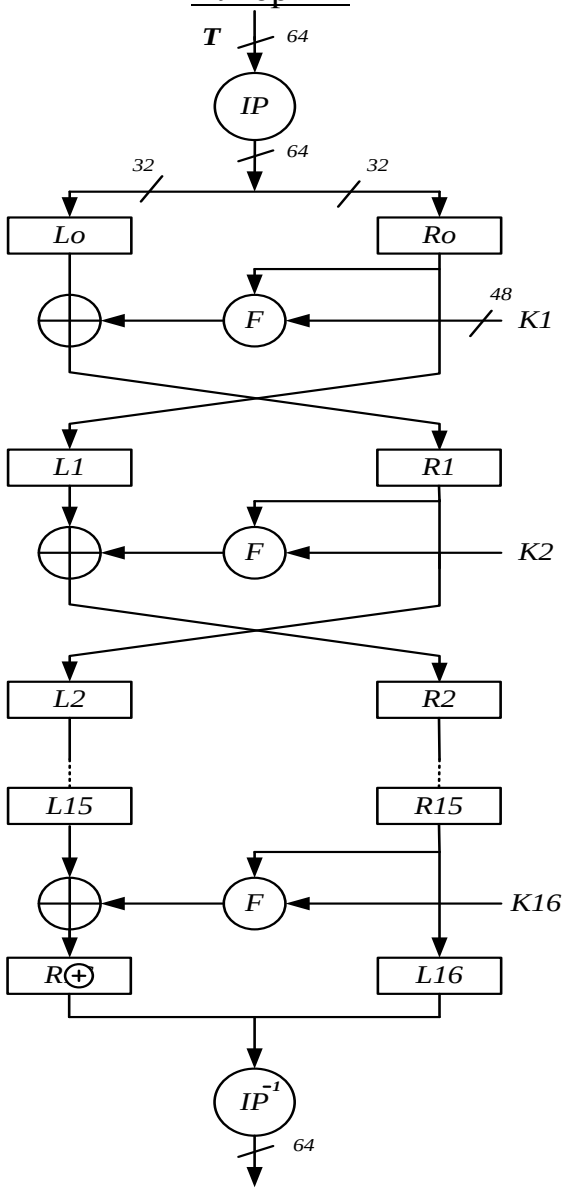
Применяя данную процедуру несколько раз, добиваются высокого качества шифрования.

4.2. Алгоритм *DES*

В 1997 г. Национальное бюро стандартов США представило стандарт шифрования *DES*, алгоритм которого был разработан специалистами *IBM* на базе шифра *LUCIFER*. Он предназначался для использования в несекретных приложениях правительства США и различных некоммерческих организациях. Алгоритм *DES* широко используется в банковской сфере, не используется в военной сфере.

DES шифрует 64–битный блок данных, используя 56–битный ключ.

Для шифрования и дешифрования используется один и тот же алгоритм.



4.3. Таблица перестановки *DES*

Входной 64–битный блок данных (T) перемешивается в соответствии с таблицей *IP*.

После прохождения через 16 итераций шифрования он перемешивается согласно обратной таблице перестановки IP^{-1}.

IP–таблица

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

IP^{-1}–таблица

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

Клетки таблиц нумеруются построчно слева направо, сверху вниз. Символы исходного текста t_1, t_2, t_3 будет заменены соответственно 58, 50, 42; при обратной табл. – на 40, 8, 48

После входной таблицы перестановки *IP* блок данных (64 бита) преобразуется в To (64 бита), т.е. $\mathbf{T_0 = IP\ (T)}$

Затем To делится на 2 части: L_0 и R_0. В *Lo* попадают первые 32 бита, в *Ro* – следующие 32 бита.

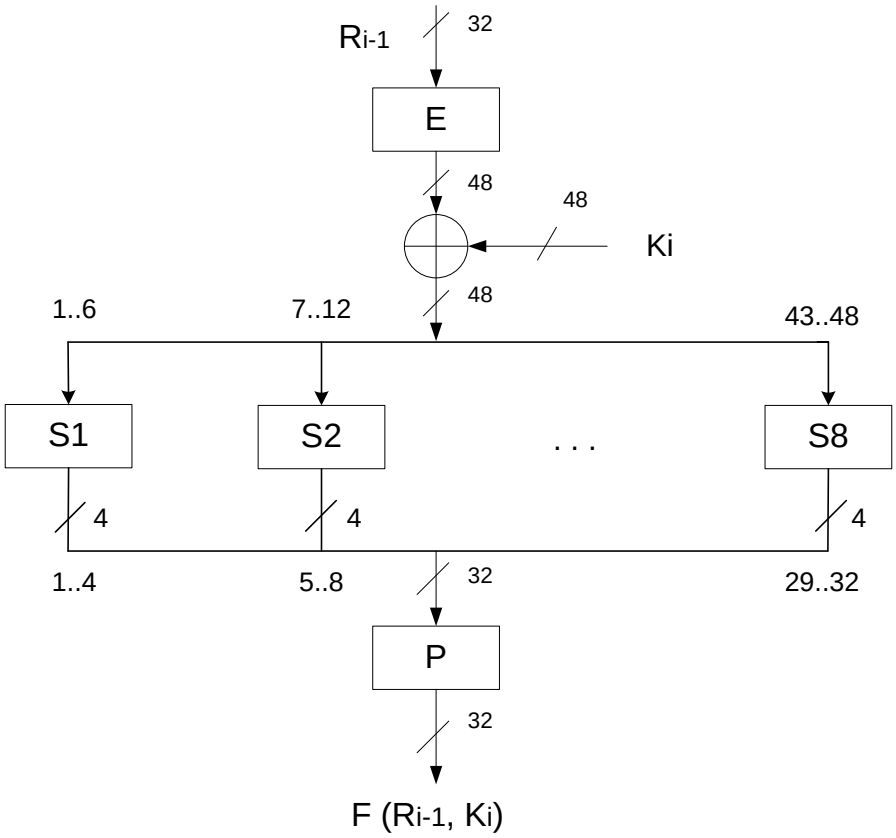
Тогда на *i*–той итерации шифрования:

$$\mathbf{L_i = R_{i-1}}$$

$$\mathbf{R_i = L_{i-1} \oplus F(R_{i-1}, K_i)}$$

где $\oplus$ – операция сложения по модулю 2

4.4. Функция *F* (подстановка + перестановка)



32–битный операнд R_{i-1} расширяется до 48 бит при помощи таблицы E.

Таблица E

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

Вход: $t_1 \quad t_2 \quad t_3$
Выход: $t'_{32} \quad t'_1 \quad t'_2$

Полученное 48–битное слово поразрядно суммируется по модулю 2 с ключом K_i. Результат разбивается на 8 блоков по 6 бит. Каждый 6–ти разрядный блок поступает на вход перестановочной функции, реализованной при помощи *S*–боксов.

Каждый *S*–бокс возвращает 4–битовое значение. Результаты объединяются в 32–битный блок и поступают на вход таблицы перестановки P.

Таблица P

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

4.5. *S*–бокс

S–бокс отображает 6–битовый блок $B_j = b_1\ b_2\ b_3\ b_4\ b_5\ b_6$ в соответствии со следующей таблицей.

Из B_j выделяются крайние биты b_1, b_6, которые являются номером строки. Оставшиеся биты определяют номер столбца.

b_1	b_6	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	**0**	*4*	*4*	*13*	*1*	*2*	*15*	*11*	*8*	*3*	*10*	*6*	*12*	*5*	*9*	*0*	*7*
0	**1**	*0*	*15*	*7*	*4*	*14*	*2*	*13*	*1*	*10*	*6*	*12*	*11*	*9*	*5*	*3*	*8*
1	**0**	*4*	*1*	*14*	*8*	*13*	*6*	*2*	*11*	*15*	*12*	*9*	*7*	*3*	*10*	*5*	*0*
1	**1**	*15*	*12*	*8*	*2*	*4*	*9*	*1*	*7*	*5*	*11*	*3*	*14*	*10*	*0*	*6*	*13*

Двоичные числа и их десятичные эквиваленты

0000 → 0	1000 → 8
0001 → 1	1001 → 9
0010 → 2	1010 → 10
0011 → 3	1011 → 11
0100 → 4	1100 → 12
0101 → 5	1101 → 13
0110 → 6	1110 → 14
0111 → 7	1111 → 15

Пример:
B_j = 1 1 0 1 1 1
1 1 – строк
1 0 1 1 – столбец
$S_j(B_j)$ = 14 = 1 1 1 0

4.6. Вычисление ключа K_i

Для каждой из 16 итераций шифрования и дешифрования используется новое значение ключа K_i (48 бит).

Вычисление ключей происходит по следующей схеме.

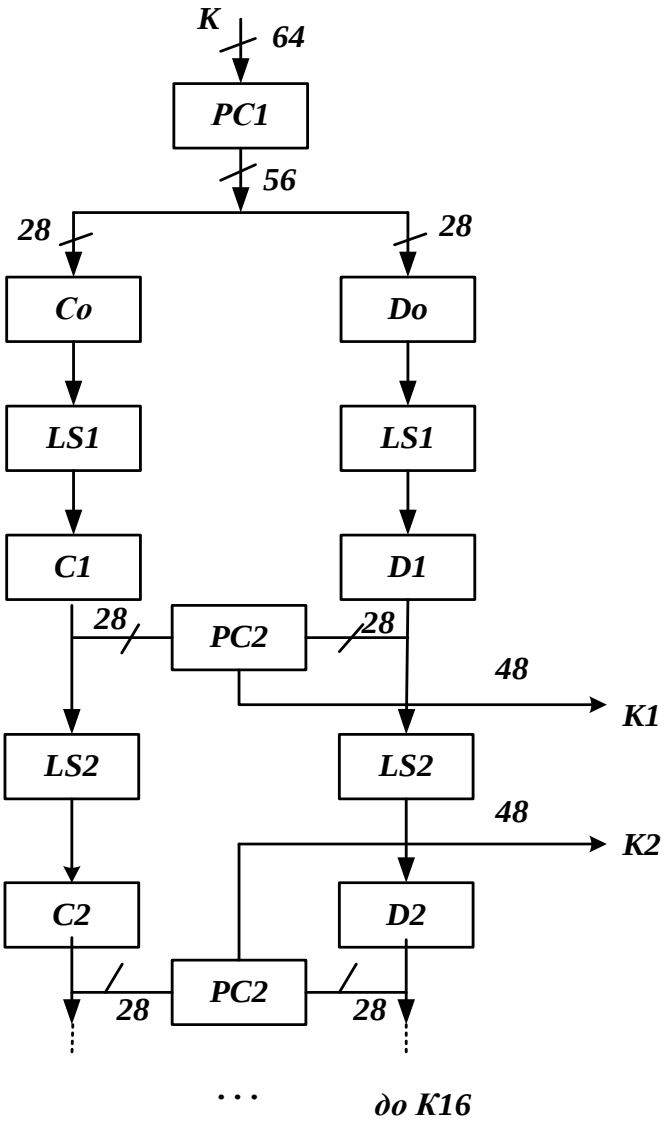


Таблица PC1

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
63	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

Таблица PC2

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

Исходный ключ К – 64 бита (56 бит информационных, 8 – контрольных: 8, 16, 24, 32, 40, 48, 56, 64).

Таблица PC1 удаляет контрольные разряды и перемешивает оставшиеся биты. Результат делится на 2 части пополам. Получаем Co и *Do*. Затем каждая из частей циклически сдвигается влево на число позиций, определяемых таблицей сдвига.

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

4.7. Декодирование (дешифрование)

Декодирование по стандарту *DES* выполняется по тому же самому алгоритму. Отличием является использование ключей. Ключ, который используется при шифровании на 1–ой итерации, в дешифровании используется на последней итерации и т.д.

2 ключевых момента:

1. процедуры перестановок *IP* и IP^{-1} взаимно обратные.

$IP(IP^{-1}(T)) = T = IP^{-1}(IP(T))$

2. при шифровании используется следующая последовательность:

$L_i = R_{i-1}$

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

При дешифровании используется обратная последовательность:

$R_{i-1} = L_i$

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

4.8. Слабые ключи *DES*

Для *DES* существуют слабые ключи, для которых $K_1=K_2=\ldots=K_{16}$ (ключи на каждой итерации одинаковые). Они возникают в тех случаях, когда все биты C_0 одинаковы:

$C_0 = 00\ldots.0$

$C_0 = 11\ldots.1$

и все биты D_0 так же одинаковы:

$D_0 = 00\ldots.0$

$D_0 = 11\ldots.1$

В 16–ричном виде эти 4 ключа:

0101…01

1F1F…1F

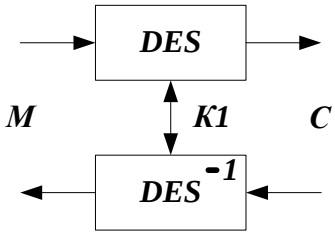
E0E0…E0

FEFE…FE

Выделяют **полуслабые ключи**, на основе которых формируются 2 внутренних ключа, каждый из которых используются по 8 раз. Они возникают в тех случаях, когда C_0 принимает значение 0101…01 или 1010…10, а D_0 – 0000…00 или 1111…11, и наоборот.

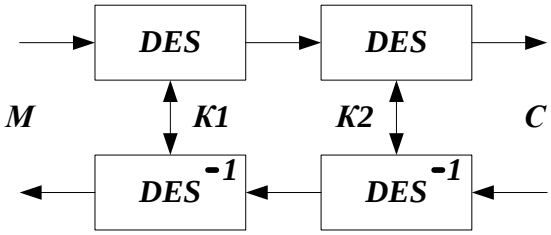
4.9. Модификации *DES*

На рисунке представлена типовая реализация *DES*. Для шифрования и дешифрования используется один ключ К. Однако разрядность ключа в 56 бит не всегда удовлетворяет пользователя.

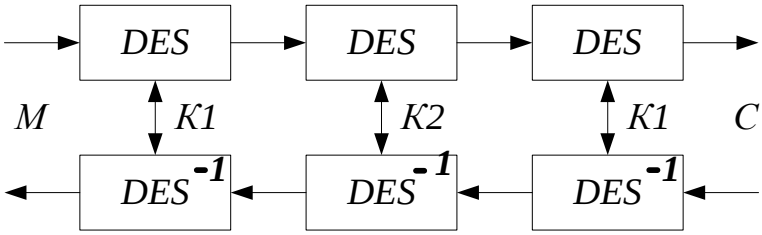


Поэтому появились модификации этого алгоритма.

–Двойной *DES.*



–Тройной *DES (TDES, Triple DES).*



Обе модификации позволяют увеличить разрядность ключа до 112 бит.

Предпочтение следует отдать последней модификации – рис. в), которая рассматривалась как кандидат на новый криптографический стандарт, но не была утверждена, т.к. придумали более сильные алгоритмы.

ТЕМА 5. КРИПТОГРАФИЧЕСКИЕ АЛГОРИТМЫ *IDEA, BLOWFISH, AES*

Вопросы:

5.1. Алгоритм шифрования данных *IDEA*
- 5.1.1. *IDEA*. Общие сведения
- 5.1.2. *IDEA*. Операции над данными
- 5.1.3. *IDEA*. Процедура шифрования

5.2. Симметричный блочный шифратор *BLOWFISH*
- 5.2.1. *BLOWFISH*. Общие сведения
- 5.2.2. Преимущества *BLOWFISH*

5.3. Стандарт *AES*
- 5.3.1. *AES*. Общие сведения
- 5.3.2. *AES*. Основные параметры
- 5.3.3. Группы, кольца и поля. Поля Галуа
- 5.3.4. *AES*. Математические операции
- 5.3.5. *AES*. Умножение полиномов

5.4. Потоковые шифры
- 5.4.1. Общая структура синхронного потокового шифратора
- 5.4.2. Генераторы криптографических ключей
- 5.4.3. Конгруэнтные генераторы
- 5.4.3.1. LFSR
- 5.4.3.2. Характеристические полиномы LFSR
- 5.4.3.3 Генераторы нелинейных последовательностей

5.1. Алгоритм шифрования данных *IDEA*

5.1.1. *IDEA*. Общие сведения

Международный алгоритм шифрования данных *IDEA* (*International Data Encryption Algorithm*) предложен в 1990 г. *J. Massey* как альтернатива стандарту *DES*. Как и *DES,* этот алгоритм основан на последовательности операций подстановок и перестановок.

При разработке алгоритма *IDEA* старались следующие **требования:**

1. длина блока данных – 64 бита.

К длине блока данных предъявляются взаимоисключающие требования:

- сложность алгоритма шифрования требует уменьшения длины блока данных;

- статистическая зависимость между блоками данных должна быть как можно меньше. Эта зависимость уменьшается с возрастанием длины блока данных.

Как компромисс выбрано 64 бита (в настоящее время это неофициальный стандарт).

2. длина ключа – 128 бит.

- чем длиннее ключ, тем выше криптостойкость;
- чем длиннее ключ, тем выше вычислительная сложность алгоритма.

3. запутанность – зависимость шифротекста от исходного текста должна быть максимально сложной и неочевидной.

4. распространение – каждый бит шифротекста должен зависеть от каждого бита исходного текста и от каждого бита ключа.

5.1.2. *IDEA*. Операции над данными

Для достижения качества шифрования были введены следующие **операции**:

1. побитовое сложение по модулю 2 16–разрядных операндов.
Обозначается – *XOR* или $\oplus$.

2. сложение целых 16–битных операндов по модулю $\mathbf{2^{16}}$.
Обозначается – $\boxtimes$.

3. умножение целых 16–битных операндов по модулю $\mathbf{2^{16}+1}$
Обозначается – $\odot$.

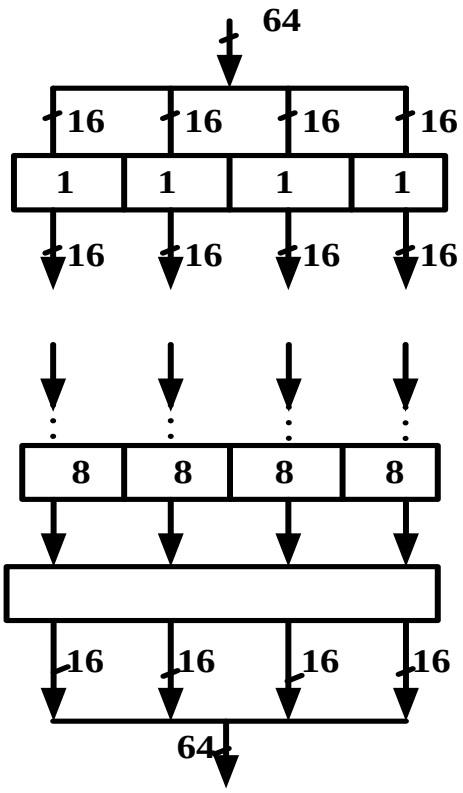
Пример:

$\oplus$				
x \ *y*	**00**	**01**	**10**	**11**
00	*00*	*01*	*10*	*11*
01	*01*	*00*	*11*	*10*
10	*10*	*11*	*00*	*01*
11	*11*	*10*	*01*	*00*

$\boxtimes$				
x \ *y*	**00**	**01**	**10**	**11**
00	*01*	*00*	*11*	*10*
01	*01*	*10*	*11*	*00*
10	*10*	*11*	*00*	*01*
11	*11*	*00*	*01*	*10*

$\odot$				
x \ *y*	**00**	**01**	**10**	**11**
00	*01*	*00*	*11*	*10*
01	*00*	*01*	*10*	*11*
10	*11*	*10*	*00*	*01*
11	*10*	*11*	*01*	*00*

5.1.3. *IDEA*. Процедура шифрования



Процедура шифрования состоит из 8–ми идентичных раундов и 9–го выходного. Используется 54 ключа (по 6 ключей в каждом раунде).

Для ускорения шифрования разработаны специальные аппаратные средства.

Для дешифрования используется взаимообратная процедура. Ключи дешифрования получаются из ключей шифрования по формулам:

$Z^{-1}_j \odot Z_j = 1 \ mod \ (2^{16} + 1)$.

$-Z_j \boxtimes Z_j = 0 \ mod \ 2^{16}$.

5.2. Симметричный блочный шифратор *BLOWFISH*

5.2.1. *BLOWFISH*. Общие сведения

Основные принципы *BLOWFISH* опубликованы в 1994 году Брюсом Шнейером как альтернатива стандарту *DES*.

При разработке *BLOWFISH* были выполнены следующие **требования:**

1. производительность – процедура шифрования и дешифрования требует минимум процессорного времени. Блок данных 64 бита обрабатывается за 18 циклов;
2. объём памяти – требуется меньше, чем 5 килобайт памяти;

3. простота реализации – не смотря на простую структуру, шифратор имеет высокую криптостойкость.
4. длина ключа – предполагает использование ключей различной разрядности: от 32 до 448 бит.
5. размерность блока данных – 64 бита.

В настоящее время *BLOWFISH* используется в большом количестве программных продуктов и имеет высокие оценки специалистов.

5.2.2. Преимущества *BLOWFISH*

1. По сравнению с *DES* значения *S*–матриц зависят от сеансового ключа.
2. На каждой итерации изменяются как младшие, так и старшие 32 бита.
3. Высокая скорость шифрования.

Алгоритм	**Количество циклов**
BLOWFISH	18
IDEA	50
DES	45
Triple DES	108

5.3. Стандарт *AES*

5.3.1. *AES*. Общие сведения

В 1997 г. американский институт стандартизации объявил конкурс на новый стандарт симметричного криптографического алгоритма.

Основные требования к кандидату:

1. длина ключа – не меньше, чем 128 бит;
2. высокая скорость шифрования;
3. простота программной и аппаратной реализации

Рассматривалось 5 кандидатов.

Алгоритм	Создатель	Страна	Скорость шифрования, Мбайт/*с*
MARS	*IBM*	США	8
RCG	*R. Rivest & Co*	США	12
Rijndael	*V.Rijmen, I.Daemen*	Бельгия	7
Serpert	ученые из различных университетов	Англия, Норвегия	2
TwoFish	Б. Шнейер	США	11

Быстродействие процессора – 200 МГц.

Все алгоритмы оказались достаточно стойкими и успешно противостояли всем известным методам взлома.

2 октября 2000 г. американский институт стандартизации объявил победителя – алгоритм *Rijndael*. С этого момента с алгоритма победителя сняты все патентные ограничения, т.е. его можно использовать в любом приложении и не платить создателю.

5.3.2. *AES*. Основные параметры

1. разрядность данных – 128 бит.

2. данные разбиты на 4 слова по 32 бита. $N_b = 4$

3. криптографический ключ может иметь длину: 128, 196, 256 бит.

$N_k = 4, 6, 8$

4. количество раундов шифрования может быть: 10, 12, 14.

В *AES* используются следующие **обозначения:**

1) все байты данных представляются в векторной форме b_7, b_6, …, b_0, Векторам соответствует полиномиальное представление:

$$b_7x^7 \oplus b_6x^6 \oplus \ldots \oplus b_1x \oplus b_0x^0 = \sum_{i=0}^{7} b_i x^i$$

Пример:

$00110101 = 0\cdot x^7+0\cdot x^6+1\cdot x^5+1\cdot x^4+0\cdot x^3+1\cdot x^2+0\cdot x^1+1\cdot x^0 = x^5+x^4+x^2+1$

+ – это сумма по *mod* 2.

2) входной блок данных (128 бит) разбивается на 16 байт a_0, a_1, … , a_{15}.

3) все внутренние преобразования выполняются над двумерными массивами, называемыми **состояниями** (*state*).

Состояние представляется как матрица $S_{r,c}$, где $r \le 4$, $c \le 6$.

Процесс обработки данных

in_0	in_1	in_2	in_3	=>	$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$	=>	out_0	out_1	out_2	out_3
in_4	in_5	in_6	in_7		$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$		out_4	out_5	out_6	out_7
in_8	in_9	in_{10}	in_{11}		$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$		out_8	out_9	out_{10}	out_{11}
in_{12}	in_{13}	in_{14}	in_{15}		$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$		out_{12}	out_{13}	out_{14}	out_{15}

5.3.3. Группы, кольца и поля. Поля Галуа

В элементарной арифметике используются 2 операции: сложение и умножение, важнейшим свойством которых является ассоциативность:

$$\boldsymbol{a * (b*c) = (a*b) * c}$$

* – одна из операций: сложение либо умножение

Среди всех возможных алгебраических систем, имеющих 1 ассоциативную операцию, самыми изученными являются группы.

Группой (*G*, *) называется некоторое множество *G* с бинарной операцией * на нём, для которых выполняются следующие **условия**:

1. операция * является ассоциативной: $a*(b*c) = (a*b)*c$ (для любых элементов, принадлежащих *G*)
2. В *G* существует единичный элемент **e**:

$$\mathbf{a*e = e*a = a}$$

3. для каждого a є *G* существует обратный элемент a^{-1}:

$$\mathbf{a * a^{-1} = e}$$

4. если для любого *a, b* є *G*: выполняется $a*b = b*a$, то группа *G* называется абелевой или коммутативной.

Кольцом (*R*, +, ·) называется множество *R* с 2–мя бинарными операциями: сложение и умножение, такими что:

1. *R*–абелева группа относительно операции сложения.

2. операция умножения ассоциативна: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

3. выполняется дистрибутивный закон: для любого *a, b,* c є *R*:

$a \cdot (b + c) = a \cdot b + a \cdot c$

Операции + и · не обязательно являются стандартными операциями сложения и умножения.

Простейшим примером кольца является множество целых чисел.

Для любого *a* є *R* $a \cdot 0 = 0 \cdot a = 0$

Если $p \neq 0$, $q \neq 0$, *p, q* є *R* и для них выполняется $p \cdot q = 0$, то они называются соответственно левым и правым делителем нуля.

В кольце без делителей нуля, если $a \cdot b = 0$ $\Rightarrow \begin{Bmatrix} a = 0 \\ b = 0 \\ a = b = 0 \end{Bmatrix}$

Поле есть кольцо с единицей, которое содержит:

- по крайней мере один элемент отличный от нуля;
- для любого $a \neq 0$ существует мультипликативно обратный элемент a^{-1}.

Ненулевые элементы поля *F* образуют группу по умножению. Делителей нуля в поле нет.

Если *a, b* є *F* и $b \neq 0$, то уравнение $a \cdot x = b$ имеет единственное решение.

Кольцо или поле коммутативно, если $a \cdot b = b \cdot a$.

Полем Галуа (*GF*) называется конечное коммутативное поле.

5.3.4. *AES*. Математические операции

При реализации *AES* применяют следующие математические операции, определенные над конечными полями (аргументами являются байты):

1. Сложение.

Пример:

1 0 0 1 1 1 0 0 = $x^7 + x^4 + x^3 + x^2$

0 0 1 1 0 1 0 1 = $x^5 + x^4 + x^2 + 1$

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

$$\begin{array}{r} 1\,0\,0\,1\,1\,1\,0\,0 \\ \oplus\ 0\,0\,1\,1\,0\,1\,0\,1 \\ \hline 1\,0\,1\,0\,1\,0\,0\,1 \end{array}$$

2. Умножение.

Умножение в $GF(2^8)$ выполняется по модулю примитивного полинома: $\boldsymbol{m(x) = x^8 + x^4 + x^3 + x + 1}$

Полином в векторной форме:

1 0 0 0 1 1 0 1 1

$x^8\ x^7\ x^6\ x^5\ x^4\ x^3\ x^2\ x^1\ x^0$

Пример:

$(x^6 + x^5 + x^4 + x + 1) \cdot (x^7 + x^5 + x^2 + 1)\ mod\ (x^8+x^4+x^3+x+1) = x^{13} + x^{11} + x^8 + + x^6 + x^{12} + x^{10} + x^7 + x^5 + x^{11} + x^9 + x^6 + x^4 + x^8 + x^6 + x^3 + x + x^7 + x^5 + x^2 + 1\ mod\ (x^8+x^4+x^3+x+1) = x^{13} + x^{12} + x^{10} + x^9 + x^6 + x^4 + x^3 + x^2 + x + 1 mod\ (x^8+x^4+x^3+x+1)$

$$\begin{array}{l} x^{13} + x^{12} + x^{10} + x^9 + x^6 + x^4 + x^3 + x^2 + x + 1 \ \big|\ x^8 + x^4 + x^3 + x + 1 \\ -\ x^{13} + x^9 + x^8 + x^6 + x^5 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \big|\ x^5 + x^4 + x^2 \\ \hline x^{12} + x^{10} + x^8 + x^5 + x^4 + x^3 + x^2 + x + 1 \\ -\ x^{12} + x^8 + x^7 + x^5 + x^4 \\ \hline x^{10} + x^7 + x^3 + x^2 + x + 1 \\ -\ x^{10} + x^6 + x^5 + x^3 + x^2 \\ \hline x^7 + x^6 + x^5 + x + 1 \end{array}$$

$(x^6 + x^5 + x^4 + x + 1) \cdot (x^7 + x^5 + x^2 + 1)\ mod\ (x^8+x^4+x^3+x+1) = x^7 + x^6 + x^5 + x + 1$

5.3.5. *AES*. Умножение полиномов

m (x) = 0 0 0 1 1 0 1 1

Выделяют операцию умножения полинома a(x) на x по модулю m(x):

a(x)·x $mod\ m$(x)

Операция реализуется следующим образом: байт множимого сдвигается влево на 1 разряд, крайний правый бит записывается 0. если выдвинутый бит равен 1, то результат сдвига поразрядно суммируется с байтом m(x).

Пример:

a (x)= $x^6 + x^5 + x^4 + x + 1$ = 0 1 1 1 0 0 1 1

b (x)= $x^7 + x^5 + x^2 + 1$ = 1 0 1 0 0 1 0 1

a(x) · b(x) $mod\ m$(x) = ?

$a(x) \cdot x^0 =$		0 1 1 1 0 0 1 1
$a(x) \cdot x =$	0	1 1 1 0 0 1 1 0
$a(x) \cdot x =$	1	1 1 0 0 1 1 0 0 0 0 0 1 1 0 1 1 1 1 0 1 0 1 1 1
$a(x) \cdot x^3 =$	1	1 0 1 0 1 1 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 1 0 1
$a(x) \cdot x^4 =$	*1*	0 1 1 0 1 0 1 *0* 0 0 0 1 1 0 1 1 *0 1 1 1 0 0 0 1*
$a(x) \cdot x^5 =$	*0*	*1 1 1 0 0 0 1 0*
$a(x) \cdot x^6 =$	*1*	*1 1 0 0 0 1 0 0* 0 0 0 1 1 0 1 1 *1 1 0 1 1 1 1 1*
$a(x) \cdot x^7 =$	*1*	*1 0 1 1 1 1 1 0* 0 0 0 1 1 0 1 1 *1 0 1 0 0 1 0 1*

1 0 1 0 0 1 0 1 $a(x)\cdot x^7$
1 1 1 0 0 0 1 0 $a(x)\cdot x^5$
1 1 0 1 0 1 1 1 $a(x)\cdot x^2$
0 1 1 1 0 0 1 1 $a(x)\cdot x^0$
1 1 1 0 0 0 1 1
$x^7 + x^6 + x^5 + x + 1$

5.4. Потоковые криптосистемы

5.4.1. Потоковые шифры

Отличительной особенностью потоковых шифров является использование криптографических ключей большой разрядности.

Различают **2 вида потоковых шифраторов:**

1. синхронные
2. самосинхронизирующиеся

В синхронных шифраторах очередной символ ключа не зависит от исходного текста или шифротекста.

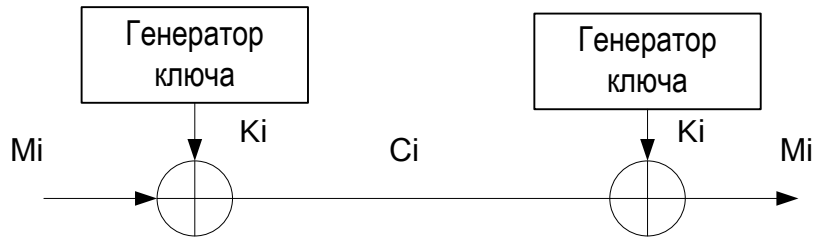
В самосинхронизирующихся шифраторах очередной символ ключа зависит от нескольких символов исходного текста или шифротекста.

Различия между потоковыми шифраторами и симметричными блочными шифраторами (*DES, AES, BLOWFISH, IDEA*):

1. блочный шифратор шифрует блок данных (64 или 128 бита), а в потоковых – за 1 такт шифруется 1 бит;

2. в потоковых шифраторах каждый бит шифротекста формируется как сумма по *mod* 2 очередного символа исходного текста и очередного символа ключа. В блочных шифраторах каждый бит шифротекста зависит от каждого бита исходного текста и каждого бита ключа (в пределах блока);
3. блочный шифратор не требует задания начального состояния для генераторов символов ключа, а в потоковых шифраторах – это обязательно;
4. блочные шифраторы используются как правило в коммерческом секторе, а потоковые – в специальных приложениях.

5.4.2. Общая структура синхронного потокового шифратора



Исходный текст преобразуется в двоичную последовательность.

Для генераторов ключей задается одинаковое начальное состояние и одинаковый алгоритм работы.

Символы исходного текста поступают на один вход сумматора по модулю 2, а символы ключа – на второй.

Символы шифротекста формируются на основании следующего выражения: $\boldsymbol{C_i = M_i \oplus K_i}$

Расшифровка: $\boldsymbol{C_i \oplus K_i = M_i \oplus K_i + K_i = M_i}$

Данная структура позволяет достичь высокой помехозащищённости, т.к. искажение 1 символа шифротекста приводит к искажению только 1 символа исходного текста.

С другой стороны, потеря синхронизации (вследствие пропадания символов или появления лишних) приводит к искажению всех последующих символов исходного текста с момента потери синхронизации.

Ключевым элементом любой потоковой системы шифрования является **генератор ключа**, который должен формировать непредсказуемые двоичные последовательности большой длины.

В качестве ключа может быть использована фиксированная двоичная последовательность (фрагмент книги, статьи и т.д.). Однако такой подход требует значительных аппаратных затрат на реализацию. Поэтому в качестве генератора ключа наибольшее распространение получили генераторы двоичных псевдослучайных последовательностей.

Шифратор Вермана – если использовать детерминированную последовательность символов ключа.

5.4.3. Генераторы криптографических ключей

5.4.3.1. Конгруэнтные генераторы

Наиболее часто для генерирования псевдослучайных чисел используется **линейный конгруэнтный генератор**, который работает на основании следующего выражения:

$$\mathbf{X_{t+1}= (}\boldsymbol{ax_t}\mathbf{ + C)}\ \boldsymbol{mod\ N}$$

где a ≠ 0 – это множитель;

C – приращение;

N – мощность алфавита.

Для его работы необходимо определить начальное значение x, т.е. x_0. После этого мы можем итерационно формировать псевдослучайные числа.

Если C = 0, то это выражение определяет мультипликативный конгруэнтный генератор.

Это выражение используется для программного формирования псевдослучайных чисел.

Для архитектур *IA*–32 наиболее часто в качестве *N* используют значения: $N=2^{32} - 1 = 2147483647$. При этом рекомендуемые значения a являются: 16807, 630360016, 112817.

Типовые конгруэнтные генераторы:

1. $x_{t+1} = (1176x_t + 1476x_{t-1} + 1776x_{t-2})\ mod\ 2^{32} - 5$

2. $x_{t+1} = (2^{13}(x_t + x_{t-1} + x_{t-2}))\ mod\ 2^{32} - 5$

3. $x_{t+1} = (2^{19}(x_t + x_{t-1} + x_{t-2}))\ mod\ 2^{32} - 1629$

Недостатком этих генераторов является сложность аппаратной реализации, т.к. используется операция умножения.

5.4.3.2. LFSR

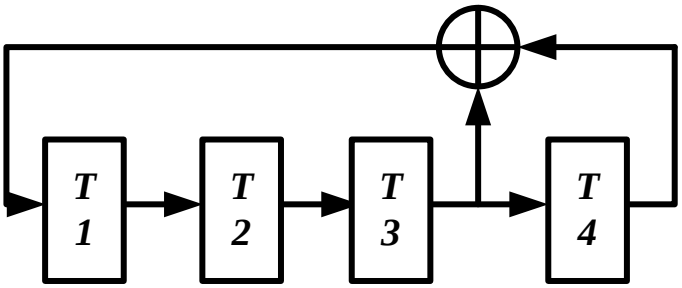
LFSR *– Linear Feedback Shift Register.*

Наибольшее распространение в качестве источника криптографического ключа для потоковых криптосистем получил генератор псевдослучайной последовательности максимальной длины. Он реализовывается на основе *LFSR* – сдвиговый регистр с линейной обратной связью.

Достоинства:

- простота программной и аппаратной реализации;
- хорошие статистические свойства (практически не отличается от случайных последовательностей);
- воспроизводимость – один и тот же фрагмент псевдослучайной последовательности можно многократно повторить, зная начальное состояние и порождающий полином..

Пример *LFSR* представлен на рисунке.



Пример:

1	0	0	0
0	1	0	0
0	0	1	0
1	0	0	1
1	1	0	0
0	1	1	0
1	0	1	1
0	1	0	1
1	0	1	0
1	1	0	1
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
1	0	0	0
0	1	0	0
$\{a_3\}$	$\{a_2\}$		$\{a_0\}$

$2^{14} - 1 = 15$

По выходу любого триггера формируется псевдослучайная последовательность максимальной длины – М–последовательность.

Вид последовательности определяется порождающим полиномом.

$\varphi(x) = x^4 \oplus x^3 \oplus 1$

Период М–последовательности определяется старшей степенью порождающего полинома.

$deg\ \varphi\ (x) = 4$

5.4.3.3. Характеристические полиномы *LFSR*

В общем случае для *m*–разрядного *LFSR* максимально возможный период псевдослучайной последовательности равен $\boldsymbol{L=2^{m-1}}$, при этом характеристический полином должен быть примитивным и неприводимым.
Аналог в целых числах – простое число.

Полином φ(x) является **примитивным,** если полином $x^к \oplus 1$ делится на полином φ(x) только при $к=2^m - 1$.

Полином φ(x) является **неприводимым**, если он не делится ни на один полином в степени меньше, чем *m*, где где *m* – старшая степень полинома φ(x). $m = deg\ \varphi(x)$

Для любой степени *m* существует $\frac{\Psi(2^m - 1)}{m}$ различных примитивных полиномов, где ψ – функция Эйлера.

m	4	6	7	10	15
число примитивных полиномов	2	6	18	60	1800

ТЕМА 6. АСИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

Вопросы:

6.1. Криптографические системы с открытым ключом
6.2. Алгоритм RSA
6.3. Криптостойкость алгоритма RSA

6.1. Криптографические системы с открытым ключом

Первые криптографические системы с открытым ключом появились в конце 1970–х годов. От классических алгоритмов они отличаются тем, что для шифрования данных используется один ключ (открытый), а для расшифрования – другой (секретный). Данные, зашифрованные открытым ключом, можно расшифровать только секретным ключом. Следовательно, открытый ключ может распространяться через обычные коммуникационные сети и другие открытые каналы. Таким образом, устраняется главный недостаток стандартных криптографических алгоритмов: необходимость использовать специальные каналы связи для распределения ключей. Разумеется, секретный ключ не может быть вычислен из открытого ключа.

6.2. Алгоритм *RSA*

В настоящее время лучшим криптографическим алгоритмом с открытым ключом считается *RSA* (по имени создателей: *Rivest, Shamir, Adelman*).

Наиболее важной частью алгоритма *RSA*, как и других алгоритмов с открытым ключом, является процесс создания пары открытый/секретный ключи. В *RSA* он состоит из следующих шагов.

1. Случайным образом выбираются два секретных простых числа, p и q, $p \neq q$.
2. Вычисляется $r=p*q$.
3. Вычисляется функция Эйлера $\varphi(r)=(p-1)*(q-1)$.
4. Выбираются открытый (*Ко*) и секретный (*Кс*) ключи, которые являются взаимно простыми с $\varphi(r)$ и удовлетворяют условию $(Ко * Кс) \bmod \varphi(r) = 1$. То есть, *Ко* является взаимообратным по модулю $\varphi(r)$ для *Кс*. Таким образом, ключом шифрования является пара значений ($Ко$, r). Ключом расшифрования является пара значений ($Кс$, r). Значение параметра r, так же как и значение ключа *Ко*, является общедоступной информацией в то время как значения параметров p, q и ключа *Кс* хранятся в секрете.

Чтобы зашифровать данные открытым ключом *Ко*, необходимо:

1) разбить исходный текст на блоки, каждый из которых может быть представлен в виде числа $M(i)$ в диапазоне $0 \ldots r-1$;

2) зашифровать последовательность чисел $M(i)$ по формуле:

$$C(i)=(M(i)Ko) \bmod r,$$

где последовательность чисел $C(i)$ представляет шифротекст.

Чтобы расшифровать эти данные секретным ключом K_C, необходимо выполнить следующие вычисления:

$$M(i)=(C(i)\text{Кс}) \bmod r.$$

В результате будет получено множество чисел $M(i)$, которые представляют собой исходный текст.

Приведем простой пример использования метода *RSA* для шифрования сообщения "*CAB*". Для простоты будем использовать малые числа (на практике используются намного большие числа).

1. Выберем $p=3$, $q=11$.
2. Вычислим $r=3\cdot11=33$.
3. Вычислим $\varphi(r)=(p-1)\cdot(q-1)=20$.
4. Выберем секретный ключ K_C, который является взаимно простым с $\varphi(r)$, например $K_C=3$.
5. На основе K_C и $\varphi(r)$ вычислим открытый ключ *Ко*. Для этого можно использовать расширение алгоритма Евклида. Расширенный алгоритм Евклида позволяет вычислить x_1 и y_1, при которых выполняется равенство $x_1\cdot a+ y_1\cdot b=d1$, где d1=НОД(a,b). Если a и b – взаимнопростые и $a>b$, то y_1 является взаимообратным для b по модулю a. То есть, $y_1 \cdot b \bmod a = 1$.

Используя данный алгоритм можно вычислить *Ко* положив a равным $\varphi(r)$ и b равным K_C:

В соответствии с алгоритмом получаем $Ko= y_1=7$.

6. Представим шифруемое сообщение как последовательность целых чисел в диапазоне 2...28. Пусть букве '*A*' соответствует число 2, букве '*B*' – число 3, а букве '*C*' – число 4. Тогда сообщение "*CAB*" можно представить в виде последовательности чисел {5, 3, 4}. Зашифруем сообщение, используя открытый ключ $Ko=7$:

$C(1) = (57) \bmod 33 = 78125 \bmod 33 = 14,$
$C(2) = (37) \bmod 33 = 2187 \bmod 33 = 9,$
$C(3) = (47) \bmod 33 = 16384 \bmod 33 = 16.$

7. Для расшифровки полученного сообщения {14, 9, 16} с помощью секретного ключа $KC=3$, необходимо:

$M(1) = (143) \bmod 33 = 2744 \bmod 33 = 5,$
$M(2) = (93) \bmod 33 = 729 \bmod 33 = 3,$
$M(3) = (163) \bmod 33 = 4096 \bmod 33 = 4.$

Таким образом, в результате расшифрования сообщения получено исходное сообщение {5, 3, 4} («*CAB*»).

6.3. Криптостойкость алгоритма *RSA*

Криптостойкость алгоритма *RSA* основывается на предположении, что исключительно трудно определить секретный ключ по открытому, поскольку для этого необходимо решить задачу о существовании делителей целого числа, то есть, найти множители параметра *r*. Данная задача не имеет эффективного (полиномиального) решения. Вопрос существования эффективного алгоритма решения данной задачи является до настоящего времени открытым. Традиционные же методы для чисел, состоящих из 200 цифр (именно такие числа рекомендуется использовать), требуют выполнения огромного числа операций (порядка 10^{23}).

ТЕМА 7. ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ

Вопросы:

7.1. Функция хеширования
7.2. Электронная цифровая подпись
7.3. Классическая схема создания цифровой подписи
7.4. Алгоритм цифровой подписи RSA

7.1. Функция хеширования

Функцией хеширования h называется преобразование данных, переводящее строку M произвольной длины в значение $m=h(M)$ (хеш–образ) некоторой фиксированной длины.

Хорошая хеш–функция должна удовлетворять следующим условиям:

1. Хеш–функция $h(M)$ должна быть чувствительна к любым изменениям входной последовательности M.

2. Для данного значения $h(M)$ должно быть невозможным нахождение значения M.

3. Для данного значения $h(M)$ должно быть невозможным нахождение $M' \neq M$ такого, что $h(M') = h(M)$.

4. Вероятность возникновения ситуации, называемой коллизией, когда для различных входных последовательностей M и M' совпадают значения их хеш–образов: $h(M) = h(M')$, должна быть чрезвычайно мала.

При построении хеш–образа входная последовательность M разбивается на блоки M_i фиксированной длины и обрабатывается по формуле: $H_i = f(H_{i-1}, M_i)$.

Хеш–значение, вычисленное в результате обработки последнего блока сообщения, становится хеш–значением (хеш–образом) всего сообщения.

В качестве примера рассмотрим упрощенный вариант хеш–функции следующего вида:

$$H_i = (H_{i-1} + M_i)^2 \bmod n,$$

где $n = p \cdot q$, p и q – большие простые числа, H_0 – произвольное начальное значение, M_i – i–й блок сообщения $M = \{M_1, M_2, \ldots, M_k\}$.

Пример. Вычислим хеш–образ для строки «ГГТУ». Для перехода от символов к числовым значениям будем использовать следующее соответствие: ‘А’ – 1; ‘Б’ – 2; ‘Г’ – 4; ‘Т’ – 20; ‘У’ – 21; ‘Я’ – 33.

Тогда сообщение M примет вид M ={4, 4, 20, 21}.

Выберем 2 простых числа p=17, q=19. Тогда модуль n=323.

Положим H_0=100.

$H_1=(H_0+M_1)^2 \bmod n=(100+4)^2 \bmod 323=10816 \bmod 323=157$.
$H_2=(H_1+M_2)^2 \bmod n=(157+4)^2 \bmod 323=81$.
$H_3=(81+20)^2 \bmod 323=188$.
$H_4=(188+21)^2 \bmod 323=\mathbf{76}$.
Таким образом, $h(M)=H_4=76$.

7.2 Электронная цифровая подпись

Электронная цифровая подпись для электронных документов играет ту же роль, что и подпись, поставленная от руки в документах на бумаге: это данные, присоединяемые к передаваемому сообщению, подтверждающие, что владелец подписи составил или заверил это сообщение. Получатель сообщения с помощью цифровой подписи может проверить, что автором сообщения является именно владелец подписи и что в процессе передачи не была нарушена целостность полученных данных.

При разработке механизма цифровой подписи возникают следующие задачи:

–формирование подписи таким образом, чтобы её невозможно было подделать;

–обеспечение возможности проверки того, что подпись действительно принадлежит указанному субъекту;

–предотвращение отказа субъекта от своей подписи.

7.3 Классическая схема создания цифровой подписи

При создании цифровой подписи по классической схеме отправитель должен выполнить следующие действия.

1. Вычислить хеш–образ m исходного сообщения M при помощи хеш–функции h.
2. Вычислить цифровую подпись S по хеш–образу сообщения с использованием секретного ключа K_c создания подписи.
3. Сформировать новое сообщение (M, S), состоящее из исходного сообщения и добавленной к нему цифровой подписи.

Получив подписанное сообщение (M', S), получатель должен выполнить следующие действия (принятое сообщение обозначено как M' по причине того, что оно могло быть преднамеренно либо случайно искажено в процессе передачи по каналу связи и может не совпадать с отправленным).

Вычислить хеш–образ m' сообщения M' при помощи хеш–функции h.

С использованием открытого ключа проверки подписи (K_o) извлечь хеш–образ m сообщения из цифровой подписи S.

Сравнить вычисленное значение m' с извлеченным из цифровой подписи значением хеш–образа m. Если хеш–образы совпадают, то подпись признается подлинной.

7.4 Алгоритм цифровой подписи *RSA*

Первой и наиболее известной во всем мире конкретной системой электронной цифровой подписи стала система *RSA*, математическая схема которой была разработана в 1977 г. в Массачусетском технологическом институте США.

Сначала необходимо вычислить пару ключей (секретный ключ и открытый ключ). Для этого отправитель сообщения (документа) выбирает два больших простых числа p и q, а затем находит их произведение

$$r = p \cdot q$$

и значение функции Эйлера от данного произведения

$$\varphi(r) = (p-1)\cdot(q-1).$$

Далее отправитель вычисляет значение K_o из условий:

$$K_o < \varphi(r),\ НОД(K_o, \varphi(r)) = 1$$

и значение K_c из условий:

$$K_c < \varphi(r),\ K_o \cdot K_c = 1\ mod\ \varphi(r).$$

Пара значений (K_o, r) является открытым ключом. Эту пару чисел автор передает партнерам по переписке для проверки его цифровых подписей. Значение K_c сохраняется автором как секретный ключ подписи.

Обобщенная схема формирования и проверки цифровой подписи *RSA* показана на рисунке 1.

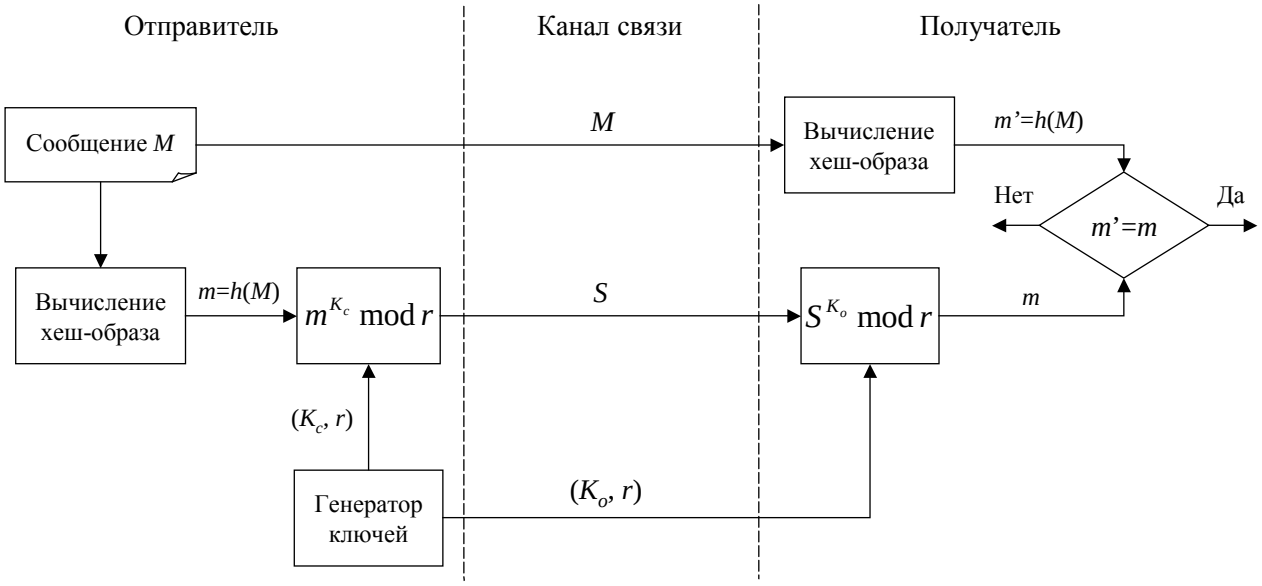


Рисунок 1 – Обобщенная схема цифровой подписи *RSA*

Допустим, что отправитель хочет подписать сообщение M перед его отправкой. Сначала сообщение M сжимают с помощью хеш–функции h в целое число m:

$$m = h(M).$$

Затем вычисляют цифровую подпись S под электронным документом M, на основе хеш–образа m и секретного значения K_c:

$$S = m^{K_c} \bmod r\ .$$

Для возведения в степень можно воспользоваться алгоритмом быстрого возведения в степень по модулю, позволяющим вычислить

$$x = a^z \bmod n.$$

Пара (M, S) передается получателю как электронный документ M, подписанный цифровой подписью S, причем подпись S сформирована обладателем секретного ключа K_c.

После приема пары (M', S) получатель вычисляет хеш–образ сообщения M' двумя различными способами. Прежде всего, он восстанавливает хеш–образ m,

применяя криптографическое преобразование подписи S с использованием открытого ключа K_o:

$$m = S^{K_o} \bmod r .$$

Кроме того, он находит результат хеширования m' принятого сообщения M' с помощью такой же хеш–функции h:

$$m' = h(M).$$

Если вычисленные значения совпадают, то есть:

$$S^{K_o} \bmod r = h(M') ,$$

то получатель признает пару (M', S) подлинной. Фальсификиция сообщения при его передаче по каналу связи возможна только при получении злоумышленником секретного ключа K_c либо за счет проведения успешной атаки против хеш–функции. При использовании достаточно больших значений p и q определение секретного значения K_c по открытому ключу (K_o,r) является чрезвычайно трудной задачей, соответствующей по сложности разложению модуля r на множители. Используемые в реальных приложениях хеш–функции обладают характеристиками, делающими атаку против цифровой подписи практически не осуществимой. Пример – хеш–функция SHA–1, принятая в США в качестве стандарта в 1995 году, формирующая 160–битовый хеш–образ при обработке сообщения блоками по 512 бит. Вероятность коллизии при использовании данной хеш–функции составляет 2^{-160} или приблизительно $6.84 \cdot 10^{-49}$.