Потоковые шифры

Особенности.

1.Операции зашифрования и расшифр. вып-ся поразрядно.

2.Каждый символ шифртекста получается в рез-те поразрядной операции слож.по модулю два символа откр.текста и символа ключа

3.Потоковый шифратор и деш-р требует задания *начального значения ключа*

4.Пот.шифры исп-ся в специальных приложениях и редко обсуждаются

Важнейшее достоинство ПШ перед блочными - высокая скорость шифрования - обеспечивается шифрование практически в реальном масштабе времени

Классический ПШ – **Шифр Вернама** (*One-time pad* — **схема одноразовых блокнотов,** 1917 г):

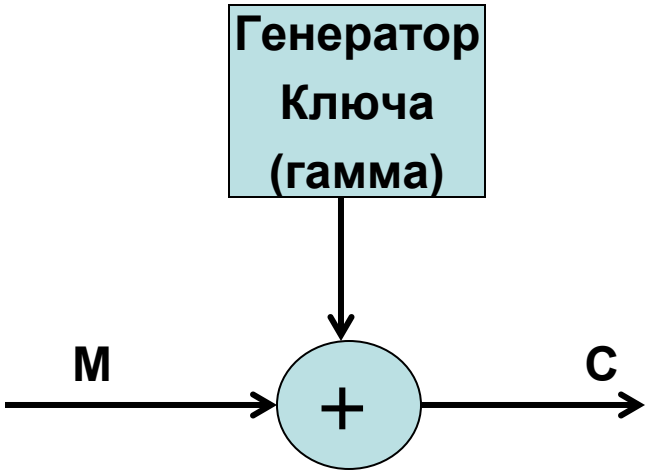
Зашифрование - открытый текст объединяется операцией «XOR» с ключом (одноразовым блокнотом или шифроблокнотом).

Ключ (**гамма**) должен обладать тремя критически важными свойствами:

- быть истинно случайным (последовательность, полученная с использованием любого алгоритма, является не истинно случайной, а псевдослучайной);
- совпадать по размеру с заданным открытым текстом;
- применяться только один раз.

В 1949 году К. Шеннон доказал абсолютную стойкость шифра Вернама - шифр Вернама является самой безопасной криптосистемой из всех возможных.

Идея гаммирования для ПШ



Генератор гаммы выдаёт ключевой поток (гамму): K=$k_1, k_2, k_3, \ldots, k_L$

Поток битов открытого текста:

M = $m_1, m_2, m_3, \ldots, m_L$.

Поток битов шифртекста : $c_i = m_i \oplus k_i$

Расшифрование производится операцией XOR между той же самой гаммой и зашифрованным текстом: $m_i = c_i \oplus k_i$

Если последовательность битов гаммы не имеет периода и выбирается случайно, то **взломать шифр невозможно.**

Типы потоковых шифров:

1. Синхронные –

- поток гаммы генерируется независимо от открытого текста и шифртекста;
- для успешного расшифрования необходимо синхронизировать ключ с шифртекстом;

Свойства:

1. Искажение одного символа в шифртексте искажает только один символ в расшифрованом тексте (**+**),
2. Защита от любых вставок и удалений шифртекста, так как они приведут к потере синхронизации и будут обнаружены (**+**)
3. Нарушение синхронизации (добавление или удаление символа) приводит к искажению всех символов после потери синхронизации (**-**)

2. **Самосинхронизирующиеся** (***асинхронные***) (1946 г) –

- значение ключа зависит либо от исходного (открытого) текста, либо от шифртекста;
- поток ключа создается функцией ключа и фиксированного числа знаков шифртекста (**N**): **внутреннее состояние генератора является функцией предыдущих N битов шифртекста** - генератор потока ключей (при расшифровании) , приняв **N** битов, автоматически синхронизируется с шифрующим генератором

Свойства:

- Так как каждый знак открытого текста влияет на следующий шифртекст, статистические свойства открытого текста распространяются на весь шифртекст (**+**),
- ошибочно удаленный или добавленный символ (бит) вызывает только ограниченное кол-во ошибочных символов в дешифрованном тексте, после чего правильный текст восстанавливается (**+**)
- каждому неправильному биту шифротекста соответствуют **N** ошибок в открытом тексте) (**-**)

Генератор ключа (ГК)

Эффективный ГК – главная проблема ПШ: генерирование длинных ПСП

Алгоритм на основе линейного конгруэнтного генератора; описывается рекуррентным соотношением:

$$x_{t+1} = (a*x_t + c) \bmod N,$$

x_0 – начальное значение ПСП, **a** – множитель, **c** – приращение, **N** - мощность алфавита

При **c=0** – мультипликативный конгруэнтный ген-р ПСП

Примеры параметров для PC с 32-разрядной архитектурой:

N= 2^{31} -1 = 2 147 483 647, a=16807; 630360016; 1078318381 4 1203248318; 397204094

Часто исп-е ГПСП:

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

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

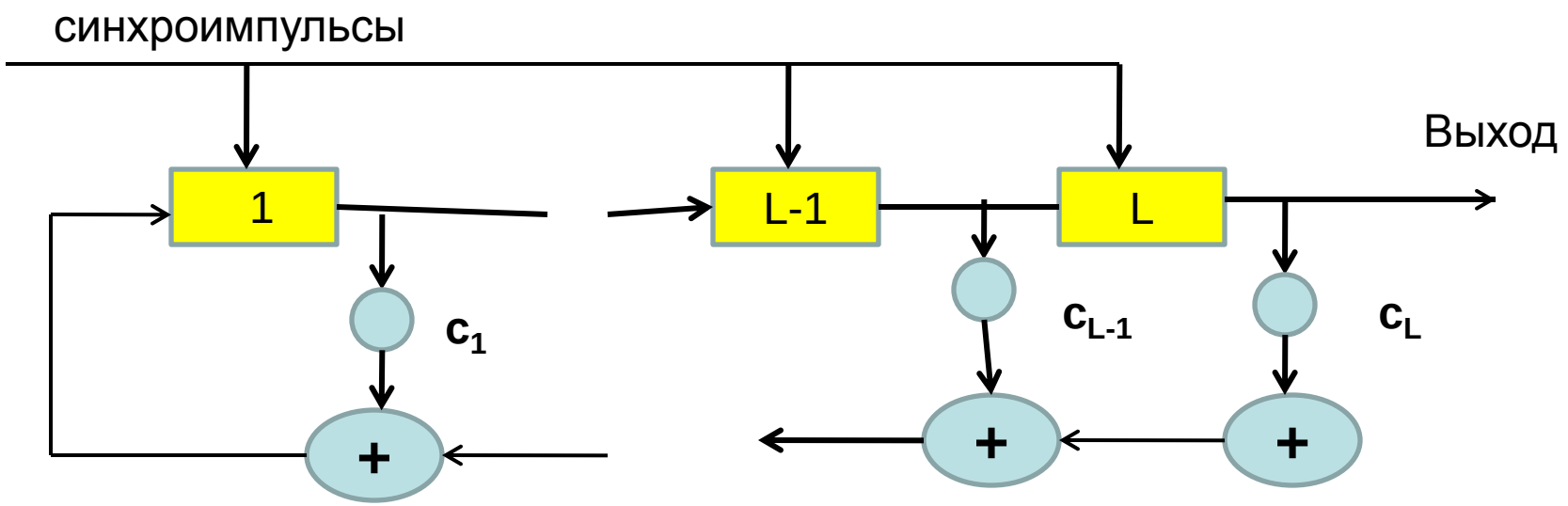
$$x_{t+1} = (1995*x_t + 1998*x_{t-1} + 2001*x_{t-2}) \bmod 2^{32} - 849,$$

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

Генераторы ПСП на основе регистров сдвига

РС – важнейший структурный компонент ЭЦВМ

РС состоит из триггеров и функций обратных связей (ФОС)



Выходная послед-ть определяется начальным состоянием каждого Тг (общее число – **L**: от **1** до **L**) и видом ФОС,

Чаще всего ФОС – XOR – РСЛОС (**регистр сдвига с линейной обратной связью**),

Период регистра сдвига — длина получаемой последовательности до начала её повторения

Свойства:

1.В течение каждой единицы времени (за такт)выполняются следующие операции:

содержимое ячейки **L** формирует часть выходной последовательности;

содержимое **i**-й ячейки перемещается в ячейку **i+1**

новое содержимое ячейки 1 определяется битом обратной связи, который вычисляется сложением по модулю с определёнными коэффициентами $\mathbf{c_i}$ битов ячеек .

2. Так как существует $\mathbf{2^L-1}$ разных ненулевых состояний регистра, то период последовательности, генерируемой РСЛОС при любом **ненулевом начальном состоянии**, не превышает $\mathbf{2^L-1}$.

3. Свойства ПСП зависят от **ассоциированного многочлена** :

$$\mathbf{C(x) = 1+c_1x+c_2x^2+ \ldots + c_Lx^L}$$

Его ненулевые коэффициенты называются **отводами**, (как и соответствующие ячейки регистра, поставляющие значения аргументов функции обратной связи).

•Важное свойство многочлена **C(x)** - приводимость.

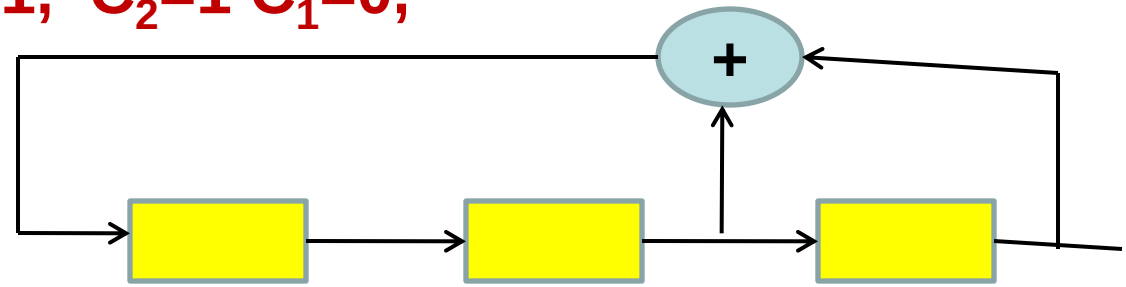
•Многочлен называется *приводимым*, если он может быть представлен как произведение двух многочленов меньших степеней с коэффициентами из данного поля (в нашем случае с двоичными коэффициентами).

Если нет, то многочлен называется *неприводимым*.

•Если многочлен является неприводимым, то период ПСП будет максимально возможным : $2^L - 1$

Пример. $C(x) = 1+c_2x^2+ c_3x^3 = x^3 + x^2 + 1 \rightarrow 320$

$C_3=1,\ C_2=1\ C_1=0,$



L=3

Многочлен $x^3 + x^2 + 1$ (320) является неприводимым.

Примеры других неприводимых многочленов:

210 $\rightarrow c_2x^2 + c_1x + 1$

310, 410, 520, ….., 84320, …

Определение периода ПСП

Обратная связь	Ячейка0	Ячейка1	Ячейка2	
1	0	0	1	1
0	1	0	0	2
1	0	1	0	3
1	1	0	1	4
1	1	1	0	5
0	1	1	1	6
0	0	1	1	7
1	0	0	1	

На выходе генератора буде последовательность: **1001011100101110010 11** …

Период равен $7 = 2^L - 1$

Задание 1. Записать выходную последовательность для генератора ПСП из предыд примера, если его начальное состояние будет:

а) 000,

б) 111

Задание 2. Построить генератор ПСП, заданный неприводимым многочленом 310. Записать выходную последовательность для генератора ПСП, если его начальное состояние будет:

а) 010,

б) 101.

Определить период.

Задание 3. Построить генератор ПСП, заданный многочленом 3210. Определить период.

Генератор ПСП на основе алгоритма BBS

Алгоритм BBS (от фамилий авторов — L. Blum, M. Blum, M. Shub) или *генератор с квадратичным остатком (1986 г.)*.
Сущность алгоритма.

1. Выбираются два больших простых числа **p** и **q**.

Читсла должны быть оба *сравнимы* с 3 по модулю 4, то есть при делении **p** и **q** на **4** должен получаться одинаковый остаток: **3**.

2. Вычисляется число **n = p* q**, называемое **целым числом Блюма** (см. алгоритм RSA).

3. Выбирается другое случайное целое число **x**, взаимно простое с **n**.

4. Вычисляется число $\mathbf{x_0 = x^2}$ mod **n**.

$\mathbf{x_0}$ называется стартовым числом генератора.

5. На каждом **i**-м шаге работы генератора вычисляется $x_{i+1} = x_i^2 \bmod n$.

Результатом **n**-го шага является один (обычно младший) бит числа $\mathbf{x_{i+1}}$

Пример.

Пусть p = 11, q = 19; убеждаемся, что

11 mod **4 = 3**, **19** mod **4 = 3**.

Тогда **n** = p* q = 11*19=209.

Выберем **x**, взаимно простое с **n**: пусть **x = 3**.

Вычислим стартовое число генератора $\mathbf{x_0}$:

$x_0 = x^2 \bmod M = 3^2 \bmod 209 = 9 \bmod 209 = 9$.

Вычислим первые десять чисел $\mathbf{x_i}$ по алгоритму BBS.

В качестве случайных бит будем брать младший бит в двоичной записи числа $\mathbf{x_i}$.

x_i:

$x_1 = 9^2 \bmod 209 = 81 \bmod 209 = 81$	младший бит:	1
$x_2 = 81^2 \bmod 209 = 6561 \bmod 209 = 82$	младший бит:	0
$x_3 = 82^2 \bmod 209 = 6724 \bmod 209 = 36$	младший бит:	0
$x_4 = 36^2 \bmod 209 = 1296 \bmod 209 = 42$	младший бит:	0
$x_5 = 42^2 \bmod 209 = 1764 \bmod 209 = 92$	младший бит:	0
$x_6 = 92^2 \bmod 209 = 8464 \bmod 209 = 104$	младший бит:	0
$x_7 = 104^2 \bmod 209 = 10816 \bmod 209 = 157$	младший бит:	1
$x_8 = 157^2 \bmod 209 = 24649 \bmod 209 = 196$	младший бит:	0
$x_9 = 196^2 \bmod 209 = 38416 \bmod 209 = 169$	младший бит:	1
$x_{10} = 169^2 \bmod 209 = 28561 \bmod 209 = 137$	младший бит:	1

Безопасность алгоритма BBS

- Основана на сложности разложения большого числа **n** на множители (это так называемая *задача факторизации*).
- Злоумышленник, зная некоторую последовательность, сгенерированную генератором BBS, не сможет определить ни предыдущие до нее биты, ни следующие - генератор BBS *не предсказуем ни в левом направлении* ни *в правом*.
- Алгоритм выдает действительно хорошую последовательность ПСЧ с большим периодом (при соответствующем выборе исходных параметров), что позволяет использовать его для криптографических целей при генерации ключей для шифрования
- **Существенным недостатком** алгоритма является невысокое быстродействие, что не позволяет использовать BBS при вычислениях в реальном времени

Алгоритм шифрования RC4

- Применяется в таких популярных протоколах, как **SSL/TLS** (для защиты Интернет-трафика) и **WEP** (для защиты WLAN-сетей).
- Создан Роном Ривестом в 1987 году в компании RSA Security как генератор потока ключевой информации с ключом переменной длины.
- Основные преимущества:
  - высокая скорость работы,
  - простота программной и аппаратной реализации.
- RC4 — фактически класс алгоритмов, определяемых размером его блока или слова – параметром **n**.
- Обычно **n=8**, но можно использовать и другие значения.
- Внутреннее состояние RC4 состоит из **массива размером 2^n** слов и **двух счетчиков**, каждый размером в одно слово.
- Два счетчика, оба при **n=8** - 8-битовые, назовем **i** и **j**.
- Все вычисления проводятся по модулю 2^n.

Алгоритм **RC4 состоит из двух этапов**:

1. Производится инициализация таблицы замен **S**.
2. Вычисляется ПСЧ.

Инициализация таблицы **S**: таблица заполняется последовательно числами от **0** до 2^n **-1**.

Ключ представляется в виде последовательности **n**-битовых слов, которыми заполняется другой массив **K**, такого же размера, как **S**. Если ключ оказался короче, чем надо, он повторяется нужное число раз. Затем выполняются следующие действия (алг.1):

1. $j = 0$; $i = 0$;
2. $j = (j + S_i + K_i) \bmod 2^n$;
3. поменять местами S_i и S_j;
4. $i = i + 1$;
5. если $i < 2^n$, то перейти на п.2

В результате выполнения алг.1 производится начальное заполнение таблицы **S**, начальное перемешивание значений *производится в зависимости от секретного ключа*.

Вычисление ПСЧ **(ПСЧ -** случайные n-битовые слова).

Счетчикам **i** и **j** присваивается начальное значение 0.
Для получения каждого нового значения z_i выполняются следующие действия (*алг. 2*):

$i = (i + 1) \bmod n;$
$j = (j + S_i) \bmod n;$
поменять местами S_i и S_j;
$a = (S_i + S_j) \bmod n;$
$z_i = S_a$.

Полученное n-битовое значение z_i может использоваться в качестве ключа для шифрования очередного n-битового блока входного потока данных.

Пример. Пусть секретный ключ состоит из шести 4-битовых (n=4) значений (приведем их в десятичном виде): 1, 2, 3, 4, 5, 6.
Сгенерируем последовательность чисел по алгоритму RC4.

1.Заполним таблицу **S** последовательно числами от 0 до 15; (15=2^n **-1)**.
Подготовим таблицу **K**, записав в нее ключ необходимое количество раз:

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

Перемешаем содержимое таблицы S (алгоритм 1).
Процесс выполнения представим в виде трассировочной таблицы, в которой укажем все производимые действия.
При выполнении вычислений необходимо помнить, что все операции сложения выполняются **по модулю 16**.

Пункт алг.	Выполняемое действие (по mod 16)	Новое значение i	Новое значение j
1	j = 0; i =0	0	
2	$j = j + S_i + K_i = 0 + 0 + 1 = 1$		1
3	Поменять местами S_i и S_j, то есть S_0 и S_1		
4	i = i +1	1	
5	i < 16, поэтому перейти на п.2		
2	$j = j + S_i + K_i = 1 + 0 + 2 = 3$		3
3	Поменять местами S_i и S_j, то есть S_1 и S_3		
4	i = i +1	2	
5	i < 16, поэтому перейти на п.2		
2	$j = (j + S_i + K_i) \bmod 16 = (3 + 2 + 3) \bmod 16 = 8$		8
3	Поменять местами Si и Sj , то есть S2 и S8		
4	i = i +1	3	
5	i < 16, поэтому перейти на п.2		
2	$j = (j + Si + Ki) \bmod 16 = (8 + 0 + 4) \bmod 16 = 12$		12
3	Поменять местами Si и Sj , то есть S3 и S12		
4	i = i +1	4	
5	i < 16, поэтому перейти на п.2		
2	$j = (j + Si + Ki) \bmod 16 = (12 + 4 + 5) \bmod 16 = 5$		5
3	Поменять местами Si и Sj , то есть S4 и S5		
4	i = i +1	5	
5	i < 16, поэтому перейти на п.2		
2	$j = (j + Si + Ki) \bmod 16 = (5 + 4 + 6) \bmod 16 = 15$		15

Таблица (продолжение)

3	Поменять местами Si и Sj , то есть S5 и S15		
4	i = i +1	6	
5	i < 16, поэтому перейти на п.2		
2	j = (j + Si + Ki) mod 16= (15 + 6 + 1) mod 16 = 6		6
3	Поменять местами Si и Sj , то есть S6 и S6		
4	i = i +1	7	
5	i < 16, поэтому перейти на п.2		
2	j = (j + Si + Ki) mod 16= (6 + 7 + 2) mod 16 = 15		15
3	Поменять местами Si и Sj , то есть S7 и S15		
4	i = i +1	8	
5	i < 16, поэтому перейти на п.2		
2	j = (j + Si + Ki) mod 16= (15 + 2 + 3) mod 16 = 4		4
3	Поменять местами Si и Sj , то есть S8 и S4		
4	i = i +1	9	
5	i < 16, поэтому перейти на п.2		
2	j = (j + Si + Ki) mod 16= (4 + 9 + 4) mod 16 = 1		1
3	Поменять местами Si и Sj , то есть S9 и S1		
4	i = i +1	10	
5	i < 16, поэтому перейти на п.2		
2	j = (j + Si + Ki) mod 16= (1 + 10 + 5) mod 16 = 0		0
3	Поменять местами Si и Sj , то есть S10 и S0		
4	i = i +1	11	
5	i < 16, поэтому перейти на п.2		
2	j = (j + Si + Ki) mod 16= (0 + 11 + 6) mod 16 = 1		1

Таблица (окончание)

3	Поменять местами Si и Sj , то есть S11 и S1		
4	i = i +1	12	
5	i < 16, поэтому перейти на п.2		
2	j = (j + Si + Ki) mod 16= (1 + 0 + 1) mod 16 = 2		2
3	Поменять местами Si и Sj , то есть S12 и S2		
4	i = i +1	13	
5	i < 16, поэтому перейти на п.2		
2	j = (j + Si + Ki) mod 16= (2 + 13 + 2) mod 16 = 1		1
3	Поменять местами Si и Sj , то есть S13 и S1		
4	i = i +1	14	
5	i < 16, поэтому перейти на п.2		
2	j = (j + Si + Ki) mod 16= (1 + 14 + 3) mod 16 = 2		2
3	Поменять местами Si и Sj , то есть S14 и S2		
4	i = i +1	15	
5	i < 16 , поэтому перейти на п.2		
2	j = (j + Si + Ki) mod 16= (2 + 7 + 4) mod 16 = 13		13
3	Поменять местами Si и Sj , то есть S15 и S13		
4	i = i +1	16	
5	i < 16 – неверно, поэтому закончить		

После выполнения алгоритма 1 получим таблицу **S**:

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

2.Генерация случайных 4-битовых слов.

Вычислим первые 5 чисел ПСП, используя алгоритм 2.

Результаты вычисления ПСП представлены в виде таблицы.

	Выполняемое действие (по mod 16)	Новое знач. **i**	Новое знач. **j**	Новое знач. **a**
Вычисление z_1	1. i = (i + 1) =0+1=1	1		
	2. j = (j + Si) mod 16=(0+13) mod 16=13		13	
	3. Поменять местами S1 и S13			
	4. a = (Si + Sj)mod 16=(7+13)mod 16=4			4
	5. z1 = S4 =**2**			
Вычисление z_2	1. i = (i + 1) =1+1=2	2		
	2. j = (j + Si) mod 16=(13+14) mod 16=11		11	
	3. Поменять местами S2 и S11			
	4. a = (Si + Sj)mod 16=(9+14)mod 16=7			7
	5. z2 = S7=**4**			
Вычисление z_3	1. i = (i + 1) =2+1=3	3		
	2. j = (j + Si) mod 16=(11+12) mod 16=7		7	
	3. Поменять местами S3 и S7			
	4. a = (Si + Sj)mod 16=(4+12)mod 16=0			0
	5. z3 = S0=**10**			
Вычисление z_4	1. i = (i + 1) =3+1=4	4		
	2. j = (j + Si) mod 16=(7+2) mod 16=9		9	
	3. Поменять местами S4 и S9			
	4. a = (Si+ Sj)mod 16=(3+2)mod 16=5			5
	5. z4= S5=**15**			

•В результате получили первые четыре значения ПСП: 2, 4, 10, 15.

•Понятно, что при **n=4** генерируемые числа будут иметь размер **4 бита**, то есть иметь значения **от 0 до 15**.

•В случае использования **n=8** таблица замен S должна состоять из 2^8=256 значений, а элементами таблицы замен должны быть числа от 0 до 255.

Размер счетчиков **i** и **j** должен также изменить до восьми бит (максимальное значение – 255).

Все вычисления в случае n=8 необходимо выполнять по модулю 256.

•Аналогичные изменения в алгоритме необходимо производить и при других значениях параметра **n**.