**Билет №1**

**1.1 Бинарные алгоритм**. Развитие алгоритма Евклида-бинарный алгоритм. Этот алгоритм-расширение алгоритма Евклида и основан на утверждениях:

Если и a и b чѐтные числа, тогда (a,b)=2(a/2,b/2);

Если a чѐтное, а b нечѐтное число, то (a,b)=(a/2,b);

В соответствии с теоремой 3.8 (a,b)=(b,a-b);

Если и a и b нечѐтные числа, то a-b является чѐтным числом.

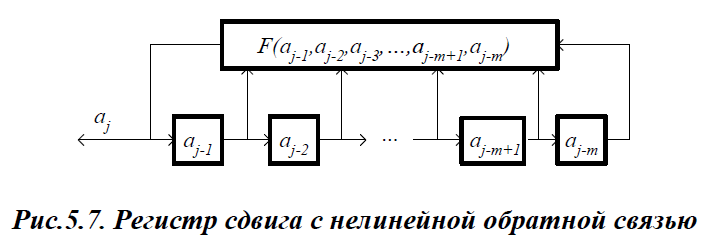
**Tеорема 3.8**. Если a=bq+r, тогда наибольший общий делитель чисел a,b равен наибольшему общему делителю чисел b,r, то есть (a,b)=(b,r).

Пример: НОД (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.

**1.2 Генераторы нелинейных М-последовательностей.**

В общем случае генераторы нелинейных последовательностей строятся с использованием нелинейных зависимостей для получения очередного выходного значения. Чаще всего для этих целей используется регистры сдвига с нелинейной обратной связью *F*. Структура подобного генератора включает в себе два основных блока, а именно сдвиговой регистр на один разряд вправо и комбинационную схему, реализующую нелинейную функцию, описывающую обратную связь. Аналогично регистрам сдвига с линейной обратной связью (*LFSR*) подобные устройства называются ***регистрами сдвига с нелинейной обратной связью*** (***Nonlinear Feedback Shift Register*** (***NFSR***))

****

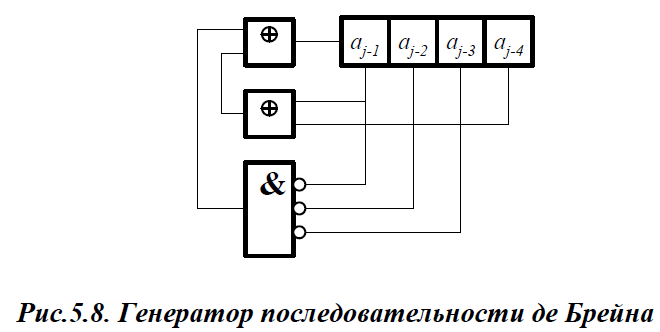
**0 0 0 1 1 1 1 0 1 0 1 1 0 0 1**

**0 0 1 1 1 1 0 1 0 1 1 0 0 1 0**

Хорошо изученной разновидностью подобных генераторов являются генераторы ***последовательностей де-Брейна*** (***De-Bruijn***), характерной чертой, для которых является генерирование всевозможных *2m* двоичных комбинаций, где *m* является разрядностью регистра сдвига, на котором формируется последовательность де Брейна. Существует процедура преобразования *LFSR* в *NFSR* генерирующего последовательность де Брейна. В этом случае нелинейная функция *F(aj-1,aj-2,aj-3,…,aj-m+1,aj-m)* образуется как

сумма по модулю два функции *f(aj-1,aj-2,aj-3,…,aj-m+1,aj-m)*, описывающей линейную обратную связь LFSR, с нелинейной функцией *g(aj-1,aj-2,aj-3,…,aj-m+1)=a\*j-1,a\*j-2,a\*j-3,…,a\*j-m+2,a\*j-m+1*. Здесь *a\*i=1**ai*.

Вид нелинейной функции *g(aj-1,aj-2,aj-3,…,aj-m+1)=a\*j-1,a\*j-2,a\*j-3,…,a\*j-m+2,a\*j-m+1* свидетельствует о том, что данная функция принимает единичное значение только на двух наборах переменных (двух состояний регистра сдвига), а именно на наборе 000…001 и 000…000. Отметим, что в последовательности кодов формируемых LFSR присутствует только код 000…001, и нет больше кодов, у которых все первые *m-1* бита, принимают нулевое значение. Кроме того, независимо от вида конкретного примитивного порождающего полинома, функция *f(aj-1,aj-2,aj-3,…,aj-m+1,aj-m),* описывающая линейную обратную связь LFSR, всегда принимает единичное значение на наборе 000…001. Таким образом, при появлении на регистре сдвига набора 000…001 обе функции и *g* и *f* принимают единичное значение, а *F* соответственно нулевое в силу чего на регистре сдвига будет получен код 000…000, который будет являться следующим набором аргументов для указанных функций *g*, *f* и *F*. В следующем цикле *g* будет ровняться единице, а *f* нулю, тогда *F*=1, и на регистре сдвига будет получен код 100…000. ***Пример 5.6.*** Простейшим примером генератора последовательности де Брейна является генератор представленный на рисунке 5.8.

****

Данный генератор синтезирован на базе LFSR соответствующего порождающему полиному *(x)=1**x1* *x4* линейная обратная связь, которого описывается функцией *f(a1,a2,a3,a4)*=*a1**a4*. Нелинейная функция *F(aj-1,aj-2,aj-3, …,aj-m+1,aj-m)* для данного примера принимает вид *F(aj-1,aj-2,aj-3,…,aj-m+1,aj-m)= a1**a4**(a\*1a\*2a\*3).* Диаграмма состояний генератора приведена в таблице.

Последовательность де Брейна, как видно из приведенного примера, может быть получена путем добавления нулевого символа к последовательности *2m-1* бит, таким образом, что на периоде ее повторения образовалась серия из *m* нулей. Такая последовательность соответственно имеет период равный *2m* и характеризуется наличием на интервале повторения всевозможных кодов длиной *m*, каждый из которых встречается только один раз. Последнее свойство и определило интерес к использованию последовательностей де Брейна для различных приложений в криптографии.

**1.3 Алгоритм цифровой подписи DSA.**

Алгоритм DSA (Digital Signature Algorithm – алгоритм цифровой

подписи) был предложен Национальным институтом стандартов и технологий в

августе 1991. Данный алгоритм вместе с криптографической хеш-функцией

SHA-1 является частью DSS (Digital Signature Standard – стандарт цифровой

подписи) − криптографического стандарта электронной цифровой подписи,

используемой в США. DSA основан на трудности вычисления дискретных

логарифмов и базируется на схеме, первоначально представленной Эль-

Гамалем и Шнорром.

Алгоритма цифровой подписи DSA состоит в следующем. Сначала

необходимо получить секретный и открытый ключи, для этого выполнить

следующие действия:

1. Выбрать большое простое число *q.*

2. Выбрать простое число *p* такое, что *q* является делителем (*p*−1).

3. Подобрать число *g* такое, что для него верно *g* = *h*(*p*−1)/*q* mod *p*, где *h* −

некоторое произвольное число из интервала (1, *p*−1), и при этом *g*>1. В

большинстве случаев значение *h* = 2 удовлетворяет этому требованию.

4. Закрытый ключ отправителя *х* выбирается случайно из интервала (0, *q*).

5. Открытый ключ вычисляется из закрытого ключа по формуле:

*у* = *gx* mod *p*. (3.5)

Вычислить *у* по известному *х* довольно просто (используя алгоритм

быстрого возведения в степень). Однако, имея открытый ключ *у*,

вычислительно невозможно определить *х*, который является дискретным

логарифмом *у* по основанию *g*.

Открытой информацией являются значения *p*, *q* и *y*, закрытой – *х*. При

этом значения *p* и *q* могут быть общими для группы пользователей, а значеие *y*

и *х* – для каждого свое.

Подпись сообщения выполняется по следующему алгоритму:

1. Получаем хеш-образ исходного сообщения *h*(*M*). При использовании

формулы *Hi* = (*Hi−1* + *Mi*)^2 mod *n*, вычисления необходимо выполнять по модулю числа *q*.

2. Выбирается случайное число *k* из (0, *q*), уникальное для каждого

подписи.

3. Вычисляется значение *r* и *s* по формулам:

*r* = (*g^k* mod *p*) mod *q*,

*s* = *k^(*−1)\*(*h*(*M*)*+x*\**r*) mod *q*. (3.6)

4. Если одно из полученных значений *r* или *s* будет равно 0, то

необходимо повторить вычисления для другого значения *k*. Иначе, подписью

будет пара значений (*r*, *s*).

27

Таким образом сообщение с подписью будет иметь вид {*M*, *r*, *s*}.

Для того чтобы проверить подлинность подписи, сначала из полученного

сообщения {*M*’, *r*, *s*}вычисляется хеш-образ *h*(*M*’), после чего находят значение

*v*, используя формулы 3.7. Подпись признается подлинной, если *v* = *r*.

*w* = *s^(*−1) mod *q*,

*u*1 = *h*(*M*)\**w* mod *q*,

*u*2 = *r*\**w* mod *q*,

*v* = (*g^u*1\**y^u*2 mod *p*) mod *q.*

(3.7)

Приведем пример данного алгоритма подписи. Возьмем приведенное

выше сообщение "БГУИР", хеш-образ которого равен 93. Далее сгенерируем

открытый и закрытый ключи для создания подписи. Для этого выберем

случайные простые числа *q* и *p*, пусть они будут равны соответственно 107 и

643. Как видно *p*−1 (642) делится на *q* (107) без остатка. Тогда число будет *g*

равно 64. Далее выберем случайное число *x* = 45, которое будет секретным

ключом и храниться в секрете, и вычислим для него открытый ключ по

формуле 3.5: *у* = = *gx* mod *p =* 6445 mod 643 = 181. Значение *y* является открытой

информацией.

Вычислим цифровую подпись для сообщения. Для этого возьмем его

хеш-образ *h*(*M*) = 93, сгенерируем случайное число *k* = 31, и вычислим *r*, *s* по

формулам 3.6:

31

1 ( ) 1

( mod )mod (64 mod 643)mod107 36,

1

( ( ) )mod (93 45 36)mod107 31 1713mod107 38.

31

*k*

*q*

*r g p q*

*s k*− *h m x r q* ϕ −

= = =

= + ∗ = + ∗ = ∗ =

Так как оба полученных значения *r* и *s* не равны 0, то подпись будет

равна паре значений (36, 38). И отправляемое сообщение будет иметь вид:

{БГУИР, 36, 38}.

Для проверки подлинности подписи получатель выполняет следующие

действия. Сначала он вычисляет хеш-образ сообщения "БГУИР", которое равно

93. Далее вычисляет значение *v* по формулам 3.8.

*w* = *s*−1 mod *q* = 38105 mod 107 = 31,

*u*1 = *h*(*M*)\**w* mod *q* = 93\*31 mod 107 = 101,

*u*2 *= r*\**w* mod *q* = 36\*31 mod 107 = 46,

*v* = (*gu*1\**yu*2 mod *p*) mod *q* = (64101\*18146 mod 643) mod 107 = 36.

Так как *r*= *v* (36 = 36), то подпись является подлинной.

**1.4 Лексическая стеганография.**

ЛС - кодирование данных блоками текста на уровне слова или лексики.

Самый простой подсознательный канал на естественном языке это выбор слов. Первая лексическая связь-синонимия. Вторая лексическая связь - Количество смешанной системы счисления. оно является одним из способов кодирования секретного положения в предложении, если мы думаем о положении в числовых терминах. Последняя категория данных, скрывающихся в тексте предполагает изменение самих слов. Семантические методы аналогичны синтаксическому методу. Эти методы назначают двум синонимам первичную или вторичную ценность в предложении.

**Билет №2**

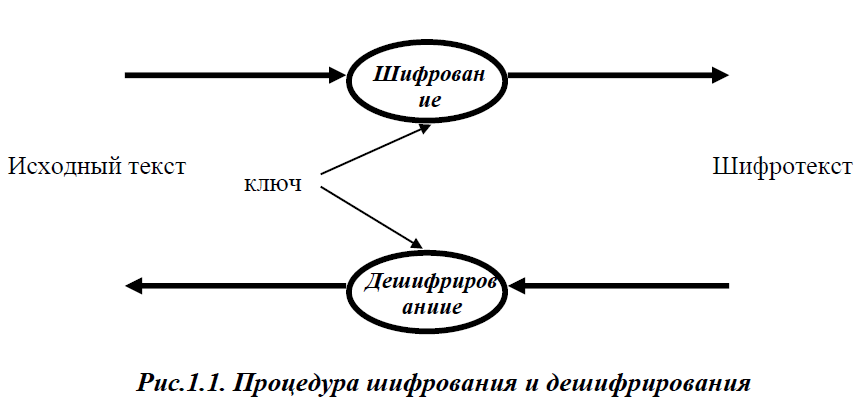
**2.1** **Введение. Криптографическое преобразование информации.**

Проблемой защиты информации путем ее преобразования занимается ***криптология*** (***cryptology***). В переводе с гречецкого языка ***kr*y*p*to*s*** означает тайный, а ***logos*** - сообщение. ***Криптология*** разделяется на два направления - ***криптографию*** и ***криптоанализ***. Цели этих направлений прямо противопо-ложные. ***Криптография*** – занимается поиском и исследованием методов преоб-разования информации с целью ее шифрования. В криптографии различают два основных преобразования это непосредственно ***шифрование*** и обратный ему процесс ***дешифрирование***. Целью криптографии является обеспечение конфиденциальности (секретности) и аутентичности (подлинности) передаваемых сообщений ***Криптоанализ*** - это область знаний, которая занимается исследованием возможностей расшифровывания информации без знания криптографических ключей. В качестве информации подлежащей шифрованию и дешифрированию, будут рассматриваться ***исходные тексты*** (открытые тексты) и ***зашифрованные тексты*** (шифротексты), построенные с использованием некоторых ***алфавитов***. Под этими терминами в дальнейшем будем понимать следующее. ***Алфавит*** - конечное множество элементов (знаков) используемых для представления текстов (***messages***). ***Исходный текст* (*Plaintext***) - упорядоченный набор из элементов алфавита исходных текстов. ***Зашифрованный текст*** (***Ciphertext) -*** упорядоченный набор из элементов алфавита зашифрованных текстов. ***Шифр*** – это секретный метод записи, в соответствии с которым исходный текст преобразуется в зашифрованный текст (шифротекст). ***Шифрирование*** (***Enciphering***) – это процесс преобразования исходного текста в шифротекст. ***Дешифрирование*** (***Deciphering***) – это обратный процесс по отношению к шифрованию, в результате которого шифротекст преобразуется в исходный текст. Шифрование и дешифрирование зависят от криптографического ключа или ключей, количество которых зависит от класса используемой криптосистемы. Различают два основных класса криптосистем:

- симметричные одноключевые криптосистемы (с секретным колючем);

- асимметричные двухключевые криптосистемы (с открытым ключом).

На следующем рисунке приведена схема, поясняющая процедуру шифрования и дешифрирования для общего случая произвольной криптографической системы.

****

**2.2 Сравнения. Модулярная арифметика**.

Два целых числа *a* и *b* ***сравнимы*** (***конгруэнтны***) ***по модулю*** натурального числа *m*, если при делении на *m* они дают одинаковые остатки. Другими словами, *a* и *b* ***сравнимы по модулю*** *m*, если их разность *a – b* делится на *m*. Например, 32 и 39 сравнимы по модулю 7, так как 32 = 7∙4 + 4, 39 = 7∙5 + 4. Утверждение *a* и *b* сравнимы по модулю *m* записывается в виде: *a**b* ***mod*** *m.* Последнее выражение называется ***сравнением (конгруэнцией)****.*

***Пример 3.11.*** Следующие равенства являются сравнениями *32**5 mod 9; 48**12 mod 9; 17**7 mod 5.* Числа *a* и *b* ***несравнимы по модулю*** *m*, если их разность *a – b* не делится на *m*. Этот факт обозначается неравенством *a≡/b mod m.* Очень часто для представления сравнения *a≡b mod m* используют символ равенства (=), тогда *a=b mod m* означает, что *a* и *b* сравнимы по модулю *m,* а *a≠b mod m,* что не сравнимы по модулю *m.* В случае, когда в сравнении *a=b mod m* величина *b<m, b* называется ***вычетом*** *a* по модулю *m.* Для сравнений существует несколько полезных лемм:

***Лемма 3.1.*** Если *a**b* ***mod*** *m*, тогда для любого целого *k* будет справедливо *ka**kb* ***mod*** *m.*

***Лемма 3.2.*** Если *ka**kb* ***mod*** *m*, и *(k,m)=1*, то *a**b* ***mod*** *m.*

***Лемма 3.3.*** Если *ka**kb* ***mod*** *km*, где *k* и *m* любые целые числа, то *a**b* ***mod*** *m.*

***Лемма 3.4.*** Если *a**b* ***mod*** *m*, и *c**d* ***mod*** *m*, то *a+c**b+d* ***mod*** *m.*

***Лемма 3.5.*** Если *a1**b1* ***mod*** *m*, и *a2**b2* ***mod*** *m*,…, *an**bn* ***mod*** *m,* то *a1+a2+a3+…+an*  *b1+b2+b3+…+bn* ***mod*** *m.*

***Лемма 3.6.*** Если *a**b* ***mod*** *m*, и *c**d* ***mod*** *m*, то *a·c**b·d* ***mod*** *m.*

***Лемма 3.7.*** Если *a1**b1* ***mod*** *m*, и *a2**b2* ***mod*** *m*,…, *an**bn* ***mod*** *m,* то *a1·a2·a3·… ·an* *b1·b2·b3·… ·bn* ***mod*** *m.*

***Лемма 3.8.*** Если *a**b* ***mod*** *m*, то для любого целого *k>0* будет справедливо *ak**bk* ***mod*** *m.* Теория сравнений эффективно используется для выполнения различных вычислений при реализации криптографических алгоритмов. Основой для реализации вычислительных процедур является так называемая ***модулярная арифметика***. Модульная арифметика основана на следующей классической лемме, используемой во многих технических приложениях. 50

***Лемма 3.9.*** *(a**b)* ***mod*** *m = [(a* ***mod*** *m)**(b* ***mod*** *m)]* ***mod*** *m,* где  это любая из следующих операций *“+“, “-”* или *“×”.* Приведенная лемма показывает, что вычисление *(a**b)* ***mod*** *m* в модульной арифметике даѐт тот же результат, что и в случае обычной целочисленной арифметике, однако существенно упрощает вычисление результата *(a**b)* ***mod*** *m.* ***Пример 3.12.*** *7·9* ***mod*** *5 =[(7* ***mod*** *5)·(9* ***mod*** *5)]* ***mod*** *5=3.* Заметим, что принцип модульной арифметики также применяется и для возведения в степень, так как операция возведения в степень равносильна повторяющемуся выполнению операции умножения.

**2.3 Графическая стеганография. LSB стеганография.**

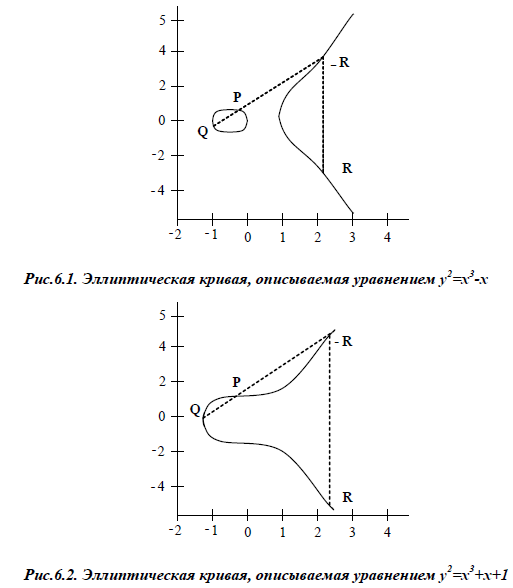
**Графическая стеганография** - скрытая передача информации через графическте файлы путём сохранения в тайне самого факта передачи. LSB (наименьший значащий бит) — суть этого метода заключается в замене последних значащих битов в контейнере (изображении) на биты скрываемого сообщения. Разбивается байт секретного сообщения на 4 двухбитовые части и заменяются полученными фрагментами младшие биты изображения. Такая замена в общем случае не заметна человеческому глазу. Более того, многие старые устройства вывода, даже не смогут отобразить такие незначительные перемены.

Для формата RGB каждый пиксель должен быть представлен 3 байтами (красный, зеленый, синий) 24 бит. Эксперименты показывают до 50% биты имеют случайный характер, и могут быть заменены.

**2.4 Эллиптические кривые. Свойства. Аналитические соотношения.**

Понятие эллиптических кривых является относительно новым понятием в криптографии, однако весьма активно обсуждаемым как один из кандидатов на последующее широкое применение для практических нужд защиты информации. ***Эллиптические кривые*** (***Elliptic Curve***) в общем случае описываются кубическими уравнениями вида *y2+axy+by=x3+cx2+dx+e.*

Где *a, b, c, d* и *e* представляют собой действительные числа, которые удовлетворяют достаточно простым требованиям. Определение эллиптических кривых включает понятие элемента *O* называемого ***бесконечным элементом*** (***infinite element***) или нулевым элементом (***zero element***). На следующих рисунках 6.1 и 6.2 приведены два примера эллиптических кривых.



Эллиптические кривые над множеством действительных чисел определяются парами чисел (*x,y*) задающих координаты ***точек*** эллиптической кривой. На практике чаще всего применяются эллиптические кривые, описываемые уравнением *y2=x3+ax+b,* где *x, y*, *a* и *b* являются действительными числами.

Если для эллиптической кривой описываемой уравнением *y2=x3+ax+b* выполняется условие *4a3 + 27b2* *0*, тогда эллиптическая кривая *y2= x3+ax+ b* может быть использована для задания множества действительных точек описываемых данной кривой. Это множество составляет группу точек, включающую также и специальную точку *O* над которыми могут быть 132 определены различные операции. Главным элементом эллиптической кривой является точка (***point***) *P*=(*x,y*), определяемая ее координатами *x* и *y*, принадлежащими этой кривой. По определению различают точку *P*=(*x,y*) эллиптической кривой и инверсную ей точку -*P*=(*x,-y*). Основополагающим свойством эллиптических кривых является следующее свойство. Секущая прямая линия всегда пересекает эллиптическую кривую только в трех точках. Исключением является случай вертикальной секущей прямой, для которой третьей точкой является бесконечный элемент *O*. Приведенное свойство позволяет определить операцию сложения двух точек эллиптической кривой. Предположим, что *P* и *Q* являются двумя различными точками эллиптической кривой и *P* не равняется *-Q*. Для сложения двух точек *P* и *Q*, проводится прямая линия через эти две точки. Согласно свойству эллиптических кривых приведенному выше эта линия пересечет эллиптическую кривую только в еще одной точке, называемой *-R*. Точке *-R* всегда соответствует симметричная ей точка *R* (см. рис.6.1 и рис.6.2). Таким образом, операция сложения над группой задаваемой точками эллиптической кривой определяется как *P + Q = R*. Приведенное определение операции сложения является геометрической интерпретацией операции сложения точек эллиптической кривой. Для случая, когда точка *Q* равняется *–P* прямая линия является вертикальной линией, которая пересекается только в этих двух точках, поэтому в основном для этих целей и был определен бесконечный элемент (нулевой элемент) *O*. Тогда согласно определению, *P + (-P) = O*. Как результат данного равенства, *P + O = P*. Элемент *O* называется аддитивным элементом (***additive identity***) либо нулевым аддитивным элементом группы определяемой эллиптической кривой. Все эллиптические кривые имеют аддитивный элемент. Более детально основные свойства операции сложения приведены ниже.

1.Элемент *O* является нулевым аддитивным элементом, для которого выполняется равенство *O=-O.* Тогда соответственно для любой точки *P* эллиптической кривой получим *P+O=P.* 2.Прямая линия, проведенная через точки *P* и *–P* является вертикальной линией. Тогда согласно определению *P+(-P)=O.*

3.Для *P*  *Q* и *P* - *Q* результатом сложения двух точек будет являться третья точка, принадлежащая эллиптической кривой, то есть *P+Q=R.*

4.В случае, когда *P* = *Q,* то есть когда точка *P* складывается сама с собой и *P*  (*x,0*), в точке *P* проводится касательная линия к эллиптической кривой. Эта линия пересечет эллиптическую кривую только в одной точке -*R.* Тогда результатом удвоения точки P будет точка *R* симметричная точке -*R.* Таким образом *P+P=2P=R.* 133

5.Для случая, когда *P* = (*x,0*), касательная к эллиптической кривой в этой точке будет представлять собой вертикальную линию. Тогда, согласно определению, для такой точки *2P = O*. Для случая вычисления *3P* будем иметь *2P + P*. Выполнив подстановку вместо *2P* нулевого элемента *O,* окончательно получим *P + O = P.* Таким образом, *3P = P*. Для *P* = (*x,0*) будут выполняться равенства *3P = P, 4P = O, 5P = P, 6P = O, 7P = P,* и так далее.

6.Операция умножения точки *P* на положительное целое число *k* определяется как сумма *k* точек *P*, так что *kP=P+P+P+…+P*. Несмотря на то, что геометрическая интерпретация операции сложения точек эллиптических кривых является достаточно наглядной иллюстрацией арифметики эллиптических кривых, он не может быть использован для практической реализации вычислений. Для этих целей используется алгебраическое определение операции сложения двух точек *P*=(*xP,yP*) и *Q*=(*xQ,yQ*), которое однозначно вытекает из геометрической интерпретации этой операции.

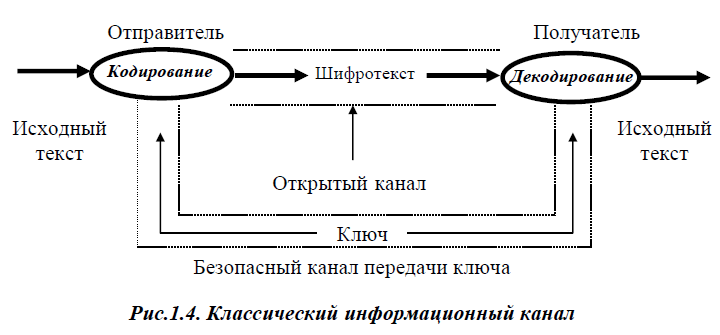
1.Если *P* и *Q* являются различными точками эллиптической кривой и *P* не равняется *–Q*, тогда результатом операции сложения *P+Q=R* является точка *R*=(*xR,yR*), координаты которой вычисляются по следующим соотношениям *s=(yP-yQ)/(xP-xQ); xR =s2- xP -xQ; yR =-yP+s(xP –xR).*

2. Удвоение точки *P* будет осуществляться подобным образом *s=(3xP2+a)/(2yP); xR =s2- 2xP; yR =-yP+s(xP –xR).*

Величина *a* используемая в последнем выражении является одним из параметров эллиптической кривой описываемой уравнением *y2=x3+ax+b*. Очевидно, что даже аналитические соотношения, определяющие операцию сложения точек эллиптической кривой, не позволяют достичь высокой степени точности вычислений. Вычисления являются, как правило, медленными, а результат не точен, что неприемлемо для криптографии, где вычисления должны давать абсолютно точный результат, а процедура вычисления результата выполняться за реальное время.

**Билет №3**

**3.1 Классический информационный канал. Формальное описание криптосистемы.**

****

В общем случае криптосистема описывается пятью компонентами.

1. Пространством исходных текстов, M.

2. Пространством шифротекстов, C.

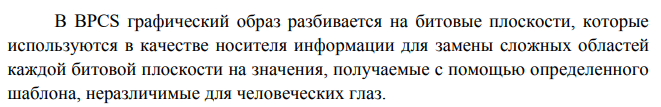
3. Пространством ключей, K.

4. Семейством шифрующих (кодирующих) преобразований, Ek: M => C.

5. Семейством дешифрирующих (декодирующих) преобразований, Dk: C => M.

+ конспект

**3.2 BPCS стеганография.**

****

Впервые при разработке была предложена мера сложности графического образа. Была предложена сложность блока двоичных значений одного из разрядов пикселей изображений. Впервые был определён блок графического изображения 8х8 пикселей. Блоки бывают последовательными либо распределёнными по графическому образу. В качестве меры сложности блока двоичных значений определена характеристика (альфа), которая вычисляется как количество реальных измерений от чёрного к белому и всех измерений цветов.

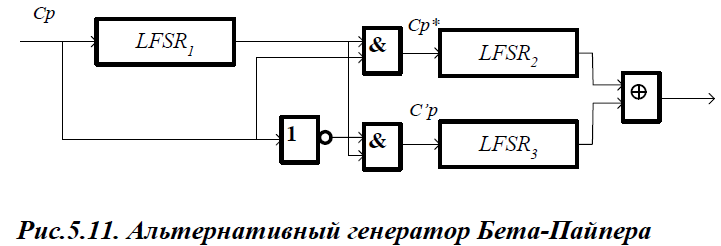
(альфа) = k/(2\*m(m-1)). k – переходы с чёрного на белый, m – max количество измерений.

Авторы предложили классификацию всех множеств двоичных образов на детерменированные и шумоподобные. В качестве характеристики используют (альфа).

+ конспект.

**3.3 Комбинированные генераторы ключа. Генератор «Stop and Go».**

Видоизмененная структура генератора Бета-Пайпера (***Alternating Stop-and-Go generator***) имеет большее распространение и также основана на использовании трех генераторов *M* последовательностей *LFSR*1, *LFSR*2 и *LFSR*3. Подобно, как и в предыдущих случаях *M* последовательности, генерируемые тремя генераторами, имеет различные взаимно простые периоды. Как видно из рис. 5.11 *LFSR*1 управляет синхронизацией *LFSR*2 и *LFSR*3. Причем если один из них выполняет микрооперацию сдвига (идет) то второй в это время не выполняет (стоит). Выходная последовательность, как и в предыдущих случаях, формируется как сумма по модулю два выходных последовательностей формируемых *LFSR*2 и *LFSR*3.

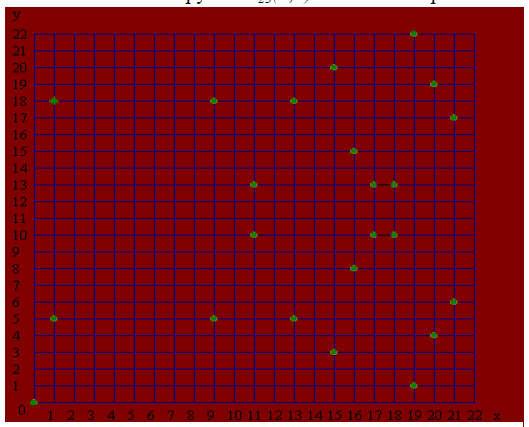


**3.4 Основные положения эллиптической криптографии. Эллиптические группы.**

Для практического применения в криптографии используются конечные поля (поля Галуа) *GF(M)* и *GF(2m).* Так, например конечное поле *GF(M)* оперирует целыми числами от *0* до *M-1,* где *M,* как правило, представляет собой большое простое число, а все вычисления выполняются по модулю *M*.

Если в уравнении *y2=x3+ax+b* параметры *a* и *b* принадлежат конечному полю *GF(M)* и оно не содержит повторяющихся сомножителей (*4a3 + 27b2* *0* 134

*mod M*), то эллиптическая кривая, описываемая таким уравнением, определяет эллиптическую группу *EM(a,b)* над полем Галуа *GF(M).* Эллиптическая группа *EM(a,b)* также может быть описана как *y2=x3+ax+b mod M*. Она включает точки, принадлежащие эллиптической кривой *y2=x3+ax+b* координаты которых являются элементами конечного поля *GF(M).* ***Пример 5.8***. Рассмотрим эллиптическую группу *EM(a,b)=E23(1,0)*, описываемую эллиптической кривой *y2=x3+x* в конечном поле *GF(23).* Данной группе принадлежит точка (*9,5*) так как ее координаты удовлетворяют уравнению *y2=x3+x mod 23.* Действительно, подставив в уравнение *y2 mod M = x3 + x mod M* значения *x=9* и *y=5,* последовательно получим *52 mod 23 = 93 + 9 mod 23, 52 mod 23 = 729 + 9 mod 23, 25 mod 23 = 738 mod 23, 2 = 2.* Таким образом, точка (*9,5*) принадлежит эллиптической группе *E23(1,0)*. Всего этой группе принадлежит 23 точки, и сама группа представляется как {*O* (*0,0*) (*1,5*) (*1,18*) (*9,5*) (*9,18*) (*11,10*) (*11,13*) (*13,5*) (*13,18*) (*15,3*) (*15,20*) (*16,8*) (*16,15*) (*17,10*) (*17,13*) (*18,10*) (*18,13*) (*19,1*) (*19,22*) (*20,4*) (*20,19*) (*21,6*) (*21,17*)}. Графически эллиптическая группа *E23(1,0)* показана на рис. 6.3.

****

Приведенный пример иллюстрирует структуру эллиптической группы в целом. Можно отметить две характерные черты эллиптической группы. Во-135

первых, их природа носит достаточно случайный характер, а во-вторых, не все целочисленные значения конечного поля *GF(23),* равно как и не все возможные точки с координатами, принадлежащими *GF(23)* входят в эллиптическую группу. Общее количество точек эллиптической группы определяется в соответствии с утверждением Хоссего (Hassego).

****

**Билет №4**

**4.1 Требования к криптографической системе. Требования секретности и достоверности. Классификация криптосистем.**

1)Зашифр. сообщение должно читаться только при использовании ключа.

2)Ошибки в шифровании не должны приводить к потере информации.

3)Число операций, необходимых для определения использованного ключа шифрования по фрагменту шифрованного текста и соответств. ему фрагменту открытого текста должно быть >= числа возможных ключей.

4)Число операций, необходимых, для расшифровки путем перебора всевозможных ключей должно иметь строгую нижнюю границу и выходить за пределы возможностей современных компьютеров

5)Знание алгоритмов шифрования не должно влиять на надежность защиты.

6)Незначительные изменения ключа должно приводить к существенному изменению вида шифрованного текста.

7)Структурные элементы алгоритма шифрования должны оставаться неизменными.

8)Дополнительные биты, вводимые в сообщение, в процессе шифрования должны быть полностью и надежно скрыты в шифрованном тексте.

9)Длина шифр. текста должна быть равна длине открытого текста.

10)Не должно быть простых и легко устанавливаемых зависимостей между ключами, используемых последовательно в процессе шифрования.

11)Любой ключ из множества возможных должен обеспечивать надежную защиту информации.

12)Алгоритм должен допускать программную и аппаратную реализацию, а изменение длины ключа не должно приводить к ухудшению алгоритма шифрования.

**Классификация криптосистем**

1) Симметричные

a. по конструктивным принципам

-блочные крипт. Системы

- поточные крипт. системы

b. по виду криптографического преобразования

-шифры замены

-композиционные шифры

-шифры перестановки

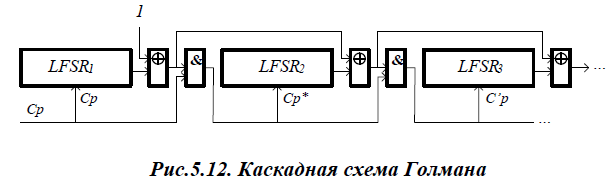
2) Ассиметричные

**4.2 Графическая стеганография. Run-Length механизм.**

**Run-length encoding, RLE или Кодирование повторов** -простой алгоритм шифрования данных, кот. оперирует сериями данных(последовательностями) в которых один и тот же символ встречается несколько раз подряд. При кодировании строка одинаковых символов, составляющих серию, заменяется строкой, которая содержит сам повторяющийся символ и количество его повторов.WWWWWBWWWBBBWWW -> 5W1B3W3B3W.

**4.3 Комбинированные генераторы ключа. Каскадный генератор Gollmann.**

Дальнейшим развитием идеи внесения нелинейных зависимостей по управлению синхронизацией является каскадная схема Голмана (***Gollmann Cascaded Key stream generator***), которая позволяет достигать желаемого качества в зависимости от количества используемых каскадов. Для случая трех каскадов подобный генератор приведен на рис.5.12.

****

Основным достоинством приведенной схемы является возможность ее адаптации для достижения, желаемого качества. Отметим, что синхронизация второго и последующих каскадов управляется предыдущими каскадами. Выполнение микрооперации сдвига либо ее отсутствие для конкретного *LFSR* завит от состояний на выходах всех предыдущих генераторов *M* последовательностей. В тоже время необходимо отметить, что данная микрооперация выполняется для каждого *LFSR* с вероятностью 0,5.

**4.4 Аутентификация ключей.**

1.субъект запрашивает доступ в систему и вводит личный идентификатор и пароль.

2. Введенные неповторимые данные поступают на сервер (А), где сравниваются с эталонными.

3. При совпадении данных с эталонными (А)признаётся успешной, при различии - субъект перемещается к шагу 1.

Ключ может передаваться 2-мя способами:

1. Незашифрованно, в открытом виде;

2.С исп-ем шифр-ния.В этом случае ключ передаётся защищённо.

+конспект

**Билет №5**

**5.1 Криптосистемы с открытым ключом. Характеристика подобных систем.**

Первые криптографические системы с открытым ключом или

ассиметричные криптосистемы появились в конце 1970-х годов. От

классических симметричных алгоритмов они отличаются тем, что для

шифрования данных используется один ключ, обычно его называют открытый

или публичный ключ, а для дешифрования – другой, секретный или закрытый,

ключ.

Диффи и Хеллман, которые впервые предложили и описали

криптосистему с открытым ключом, выявляют следующие требования к

криптосистемам:

1. Вычислительно легко создавать пару открытый ключ (*Ko*), закрытый

ключ (*Kc*).

2. Вычислительно легко, имея открытый ключ и незашифрованное

сообщение *М*, создать соответствующее зашифрованное сообщение: *С* = *Еko*[*М*].

3. Вычислительно легко дешифровать сообщение, используя закрытый

ключ: *М* = *DKc*[*C*] = *DKc*[*EKo*[*M*]].

4. Вычислительно невозможно, зная открытый ключ *Ko*, определить

закрытый ключ *Kc*.

5. Вычислительно невозможно, зная открытый ключ *Ko* и зашифрованное

сообщение *С*, восстановить исходное сообщение *М*.

Можно добавить шестое требование, хотя оно не выполняется для всех

алгоритмов с открытым ключом:

6. Шифрующие и дешифрующие функции могут применяться в любом

порядке, т.е. *М* = *ЕKo*[*DKc*[*M*]] = *DKc*[*EKo*[*M*]].

Таким образом, данные, зашифрованные открытым ключом, можно

расшифровать только секретным ключом. Следовательно, открытый ключ

может распространяться через обычные коммуникационные сети и другие

открытые каналы, что устраняет главный недостаток стандартных

криптографических алгоритмов: необходимость использовать специальные

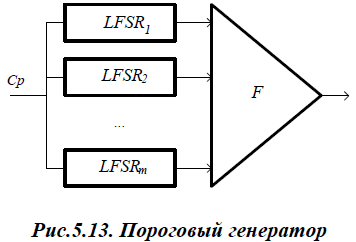
каналы связи для распределения ключей.

**5.2 Алгоритм цифровой подписи ElGamal.**

Конспект.

**5.3 Комбинированные генераторы ключа. Пороговый генератор.**

Большое количество *LFSR* использует так называемый ***пороговый генератор*** (***Threshold generator***), приведенный на рис.5.13. Количество (*m*) *LFSR* выбирается нечѐтным тогда нелинейная функция *F* формирует выходное значение 0 или 1 в зависимости от соотношения единичных и нулевых значений на выходах генераторов *M* последовательностей. Если количество единичных значений больше нулевых на выходе генератора формируется единичное значение, в противном случае выходное значение ровняется нулю.



Подобно, как и в предыдущих случаях, периоды *M* последовательностей должны быть взаимно простыми, а степени порождающих их полиномов иметь сравнимые величины и, как правило, больше 10.

**5.4 Открытая система распределения ключей (Алгоритм Diffie, Hellman).**

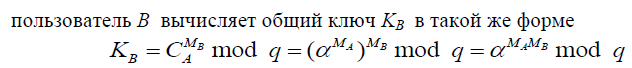
Открытая ***система распределения ключей*** (***public key distribution algorithm***), предложенная ***W.Diffie*** и ***M.Hellman,*** впервые была опубликована в 1976 году и предназначалась для распределения ключей по открытому каналу. Авторы предложили алгоритм, основанный на возведении в степень по модулю *q,* где *q* является простым числом. Стойкость предложенного алгоритма основывается на сложности вычисления дискретного логарифма над полем Галуа *GF(q)* с *q-1* элементами *{1,2,...,q-1}.* Рассмотрим пару взаимно инверсных функций: *C=**M (*mod *q); M=*log*C (*mod *q)* над полем *GF(q),* где *0<M, C<q,* соответственно*, q* простое целое число*,* а  примитивный элемент (корень) *GF(q).* Вычисление *C* из *M* осуществляется простым возведением в степень, что вычислительно не составляет труда. В тоже время вычисление дискретного логарифма над *GF(q)* процедура намного более сложная и требует для своего решения больших вычислительных затрат. Для больших значений *q* эта задача вычислительно неразрешима. На этом факте и основан рассматриваемый алгоритм, как впрочем, и большинство из известных криптосистем с открытым ключом. Сущность открытой системы распределения ключей состоит в том, что два пользователя используя открытый канал, обмениваются криптографическим ключом, который будет известен только им, двоим. Злоумышленник при попытке получения значения ключа столкнется с проблемой дискретного логарифма над полем *GF(q).* Предварительно оба пользователя *A* и *B* обмениваются параметрами (,*q*) конечного поля, где  является примитивным элементом поля, а модуль *q* представляет собой большое простое число. Эта информация является открытой для всех. Далее последовательно выполняются следующие действия.

1.Каждый из двух пользователь *A* и *B* генерирует независимое случайное число. Пользователь *A* число *MA*, а пользователь *B* число *MB* равновероятно выбранное из набора целых чисел *{1,2,...,q-1},* и каждый из них хранит свое число в секрете.

2.Оба пользователя осуществляют следующие вычисления: *A* вычисляет*,* а *B* вычисляет.

3.Затем по открытому каналу *A* посылает в адрес *B* результат своего вычисления *CA,* а *B* значение *CB*. Отметим невозможность восстановления величин *MA*, и *MB* на основании доступных для злоумышленника значений *CA,* и *CB* даже если параметры конечного поля (,*q*) ему известны.

4.Пользователь *A* вычисляет *KA,* получая *CB* из открытого файла и выполняя вычисления. 



Таким образом, оба пользователя *A* и *B* обменялись секретным ключом *KA=KB,* что следует из приведенных выше соотношений. Отметим, что значения *MA*, и *MB* являются аналогами секретного ключа пользователей *A* и *B*, а *CA,* и *CB* аналогами их открытых ключей. Как результат приведенных действий с секретными и открытыми ключами пользователей оба они получают криптографический ключ *K*=*KA=KB,* который может быть использован ими для секретного общения используя, например DES.

**Билет №6**

**6.1 Алгоритмы шифрования: Перестановочные шифры.**

Суть методов перестановки состоит в том, что исходное сообщение *M*

делится на блоки данных, состоящих из *d* символов *M* = *m*1,.*..,md* ,*md*+1,…,*m2d*,...,

после чего полученные блоки исходного текста преобразуются в соответствии с

некоторым вектором, функцией или фигурой. В результате получим *Ek*(*M*) =

= *mf*(*1),.*..,*mf*(*d*),*mf(d*+1),...,*mf*(*2d*),...

Простейшим представителем этого метода является метод

"железнодорожной изгороди", при использовании которого исходное

сообщение преобразуется в соответствии с фигурой, напоминающей по форме

железнодорожную изгородь, откуда и пошло его название. В этом случае

символы исходного текста записываются в виде, напоминающем по форме

забор, а символы зашифрованного текста считываются из полученной записи

построчно. Пример шифрование этим методом приведен на рис. 1.1.

*M* = это лабораторная работа по киоки

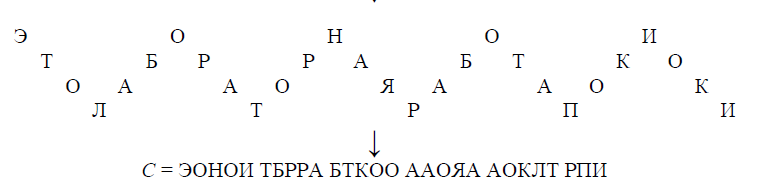


Рис. 1.1. Шифрование методом "Железнодорожная изгородь"

*M* = "Это лабораторная работа по КиОКИ" является исходным текстом, а

*C* = "ЭОНОИ ТБРРА БТКОО ААОЯА АОКЛТ РПИ" – соответствующим ему

шифротекстом. "Высота изгороди" *K* является ключом, который в приведенном

примере равен 4. Для того чтобы расшифровать полученный текст, необходимо

выполнить действия обратные выполненным при шифровании и использовать

тот же ключ.

Одной из наиболее известных модификаций метода перестановки

является "столбцовый метод", при котором исходное сообщение записывается

в таблицу построчно, а затем считывается оттуда столбцами согласно

некоторому вектору, задающему порядок считывания. Этот вектор может быть

задан с помощью ключевого слова или фразы, буквам которого назначаются

номера в соответствии с алфавитом, а если буква встречается несколько раз, то

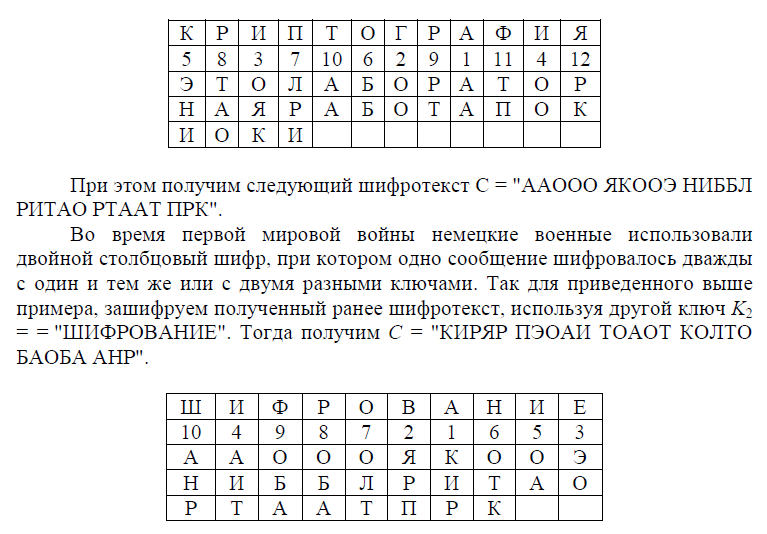
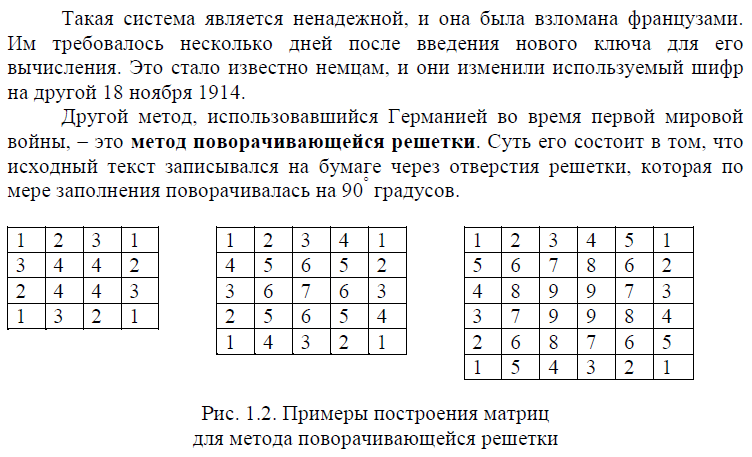
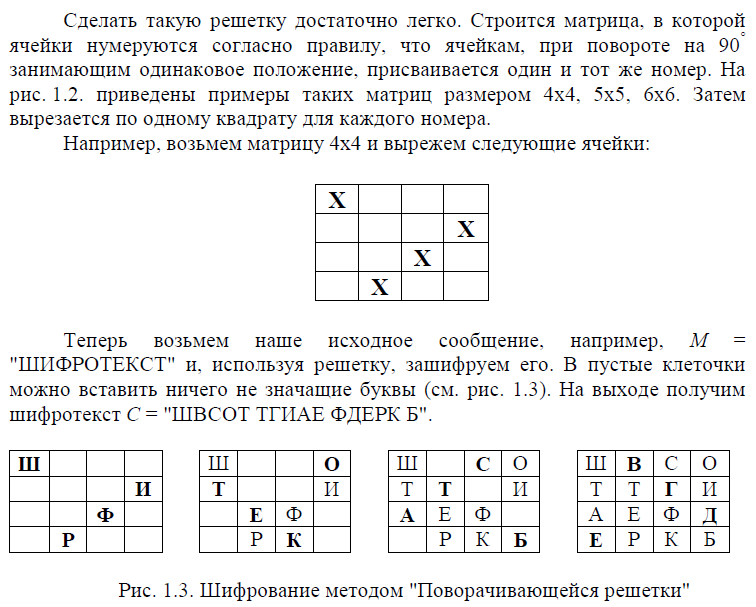
нумерация определяется порядком следования повторяющихся букв в

ключевом слове. Например, пусть у нас есть исходный текст *M* = "Это

лабораторная работа по КиОКИ" и ключ *K* = "КРИПТОГРАФИЯ". Запишем

текст в таблицу и считаем его по столбцам, порядок считывания которых задан

ключом *K* = "КРИПТОГРАФИЯ":

**  **

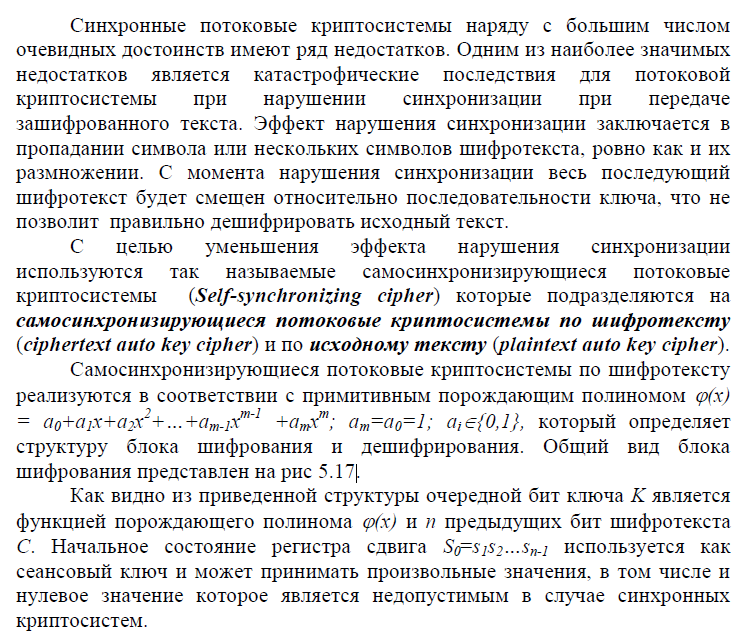
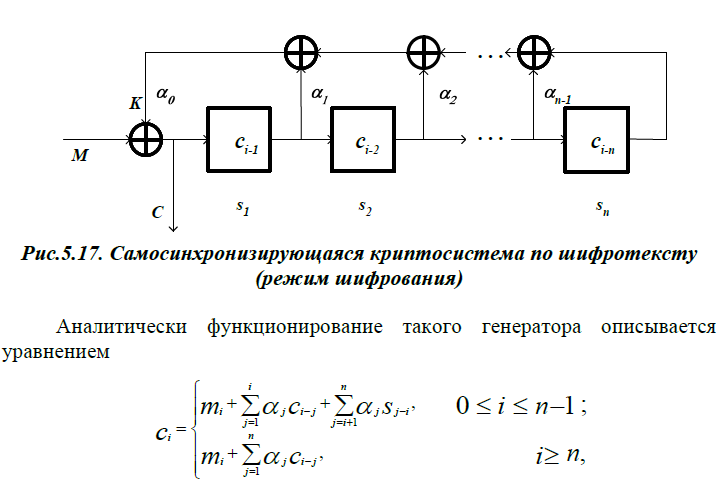
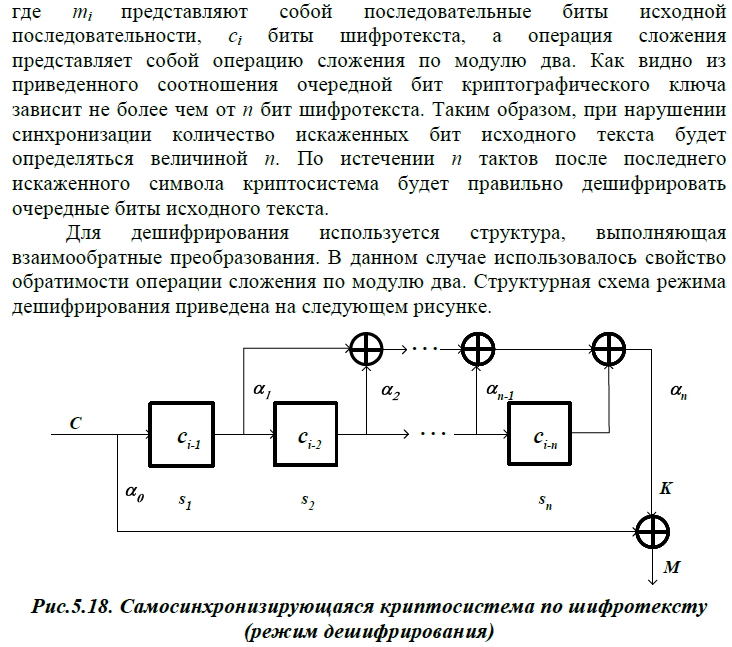
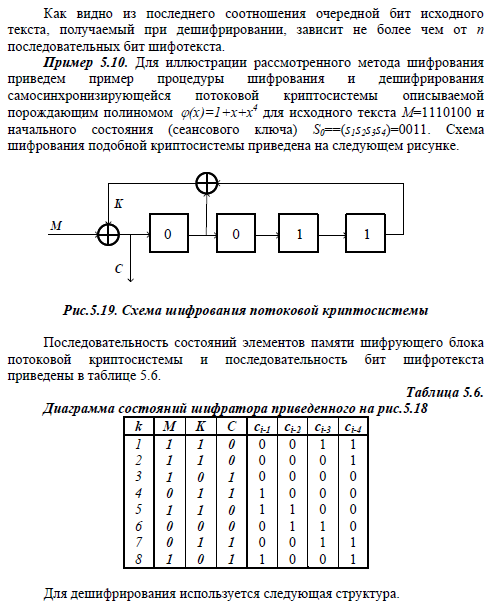
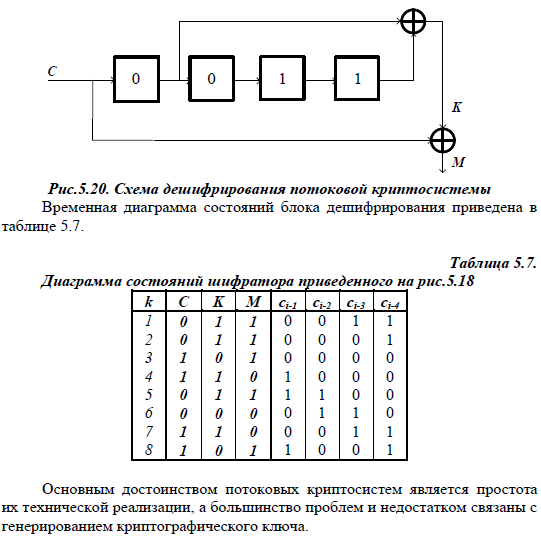
**6.2 Способы решения линейных cравнений: 1 случай (нет решений)**

**Tеорема 3.16.** Если наибольший общий делитель d чисел a и n (d=(a,n)) не является делителем b, то линейное сравнение ax=b **mod** n не имеет решений.

**Доказательство:** Предположим противоположное, что есть решение x0, которое удовлетворяет линейному сравнению, то есть ax0=b **mod** n. Согласно теореме 3.16 d является делителем a и n, откуда следует, что d должен являться делителем ax0 и nq, так же как и делителем ax0–nq=b. Здесь q есть положительное целое число. Тогда получим противоречие: по условию теоремы d не является делителем b, а в случае наличия решения x0 d должно является делителем b, значит, линейное сравнение ax=b **mod** n не имеет решений, если d=(a,n) не является делителем b.

**Пример 3.22.** Линейное сравнение 2x=1 mod 4 не имеет решения, так как наибольший общий делитель d=2 чисел a=2 и n=4 (2=(2,4)) не является делителем b=1. Действительно, ни одно из возможных целочисленных значений x<4 не удовлетворяет сравнению2x=1 mod 4.

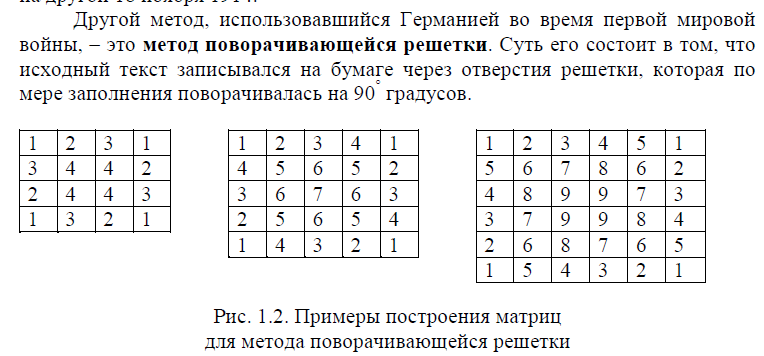
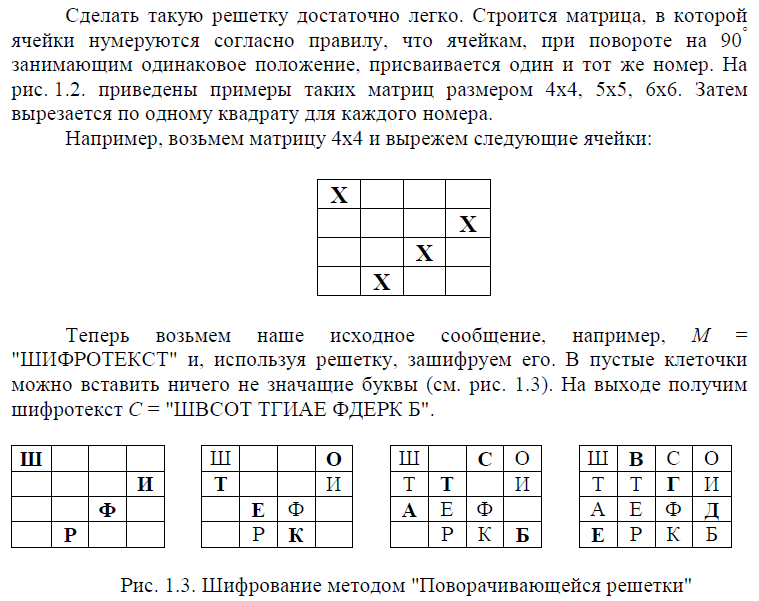
**6.3 Самосинхронизирующиеся потоковые криптосистемы.**

**    **

**6.4 Эллиптические группы. Порядок точки эллиптической группы.(-)**

**Билет №7**

**7.1 Алгоритмы шифрования: Алгоритм вращающихся квадратов.**

** **

**7.2 Способы решения линейных сравнений: 2 случай (одно решение).**

**Tеорема 3.17*.*** Если наибольший общий делитель d чисел a и n равен единице (a,n)=1, то есть, a и n являются взаимно простыми числами, то линейное сравнение ax=b **mod** n имеет одно решение.

**Доказательство:** Предположим, что существует полная система вычетов {0,1,2,…,n-1} по модулю n. Тогда согласно тому, что a и n взаимно простые целые числа, множество чисел {0·a, 1·a, 2·a,…, (n-1)·a} образует полную систему вычетов по модулю n. Тогда среди всех целых чисел есть одно ax0 и только одно с вычетом равным b.#

**Пример 3.18.** Линейное сравнение 2x=1 mod 3 имеет одно решение x0=2, так как a=2 и n=3 являются взаимно простыми числами.

**7.3 Функция Эйлера.**

**Функцией Эйлера (Euler’s)** **(n)** целого числа n1 является количество целых чисел, которые меньше чем n и взаимно просты с n. Для небольших значений n эта функция принимает следующие значения (1)=0, (2)=1, (3)=2, (4)=2, (5)=4, (6)=2, (7)=6, (8)=4, (9)=6, (10)=4, (11)=10,…. Несложно показать, что согласно приведенному определению, если n есть простое число p, то (p)=p-1. Для функции Эйлера справедлив ряд утверждений и теорем. Одной и наиболее часто используемых в криптографии теорем относительно функции Эйлера является следующая теорема.

**Tеорема 3.12.** Если n=pq, где p и q (pq) простые числа, то (n)=(p)(q) =(p-1)(q-1).

**Дoказательство:** Имеем множество {0,1,2,…,pq-1} из pq целых чисел, которые меньше чем n=pq. Все эти числа взаимно простые с n=pq, за исключением (p-1) чисел {q,2q,3q,…,(p-1)q}, кратных q, (q-1) чисел {p,2p,3p,…,(q-1)p}, кратных p и 0. Тогда, (pq)=pq-(p-1)-(q-1)-1=pq-p-q+1=(p-1)(q-1).#

**Пример 3.17.**(10)=(25)=(2)(5)=14=4.

**Tеорема 3.13.** Если p простое число, и k>0 целое число, то (pk)=pk-pk-1 =pk-1(p-1).

**Доказательство:** Множество целых чисел, которые меньше чем pk и не взаимно простые с pk, включает числа {p,2p,3p,…,(pk-1-1)p}. Это значит, что среди pk-1 чисел, меньших, чем pk, исключая ноль, есть pk-1-1 целых, не взаимно простых с pk. Tогда, (p)=pk-1-(pk-1-1)= pk-pk-1.#

**Пример 3.18.**(8)=(23)=23 –22=8-4=4.

**Tеорема 3.14.** Функция Эйлера является мультипликативной для произведения целых чисел, если эти числа являются взаимно простыми, то есть (nm)=(n)(m), если (n,m)=1. 54

Данная теорема позволяет сформулировать алгоритм вычисления функции Эйлера для произвольного целого числа a. Первоначально представим число в виде канонического разложения Евклида a=p11 p22…prr, тогда в силу того что pi iи pjj являются взаимно простыми для любых i≠j, используя предыдущую теорему получим (a)=(p11)(p22)…(prr)=(p11-p11-1)(p22-p22-1)…(prr-prr-1)=a(1-1/p1)(1-1/p2) …(1-1/pr).

**Пример 3.19.** (2700)=? 270=223352. (2700)=2700(1-1/2)(1-1/3)(1-1/5)=720.

**Теорема 3.15.** (**Euler’s**). Если n0 положительное целое число, и (a,n)=1, где a целое, то a(n)=1 **mod** n.

**Доказательство:** Пусть {r1,r2,r3,…,r(n)} представляет собой приведенную систему вычетов по модулю n, тогда для (a,n)=1 числа ar1, ar2, ar3,…, ar(n) образуют ту же приведенную систему вычетов. То есть, ar1=r mod n, ar2=r mod n, ar3=r mod n,…,ar(n)=r mod p, где {r,r,r,…,r} это есть те же значения вычетов {r1 ,r2 ,r3,…,r(n)}, переставленные в ином порядке. Используя аналогичный подход, который применялся при доказательстве теоремы 3.11, перемножив правую и левую части сравнений ar1=r mod n, ar2=r mod n, ar3=r mod n, …,ar(n) =r mod p, получим: a(n) r1 r2  r3,…r(n)= r r  r,… r(n) mod n. Учитывая, что {r1,r2, r3,…,r(n), m}=1, окончательно получим a(n) =1 **mod** n.#

**Пример 3.20.** 310 mod 11=?. Согласно теореме Ферма 310=1 mod 11, где p=11 есть простое число, и a=3 есть целое взаимно простое с p=11, то есть (11,3)=1.

**Пример 3.21.** 312 mod 26=?. Согласно теореме Эйлера 312 mod 26=1, где n=26, (26)=(213)=(2)(13)=112=12.

**7.4 Эллиптическая криптосистема.**

**-**

**Билет №8**

**8.1 Алгоритмы шифрования: Подстановочные шифры. Алгоритм Цезаря.**

Криптографические системы, основанные на методе подстановки, можно

разделить на четыре основных класса:

1. Одноалфавитный шифр подстановки (шифр простой замены) – шифр,

при котором каждый символ открытого текста заменяется некоторым

фиксированным при данном ключе символом того же алфавита. Примером

может служить шифр Цезаря.

2. Однозвучный шифр подстановки (омофонный) похож на

одноалфавитный за исключением того, что символ открытого текста может

быть заменен одним из нескольких возможных символов. К данным шифрам

относят шифр Билла.

3. Полиграммный шифр подстановки заменяет не один символ, а целую

группу символов. Примерами таких шифров являются шифр Плейфейра, шифр

Хилла.

4. Многоалфавитный шифр подстановки состоит из нескольких шифров

простой замены. Например, шифр Виженера, одноразовый блокнот.

Одноалфавитный шифр подстановки характеризуется тем, что каждому

символу алфавита исходного текста в соответствии ставиться другой символ из шифроалфавита. Криптографическим ключом такой системы является таблица соответствия исходного алфавита алфавиту подстановки. Например, для английского алфавита существует 26! ≈ 4\*1026 различных криптографических ключей.

Примером такого шифра является **шифр Цезаря**. Так, шифр Цезаря

заменяет каждый символ алфавита исходного текста на сдвинутый относительно него символ того же алфавита на *k* позиций вправо, при этом *k*

является ключом шифра. Т.е. в алгоритме Цезаря *i*-й символ алфавита

заменяется (*i+k*)-м по модулю *n* символом, где *n* – количество букв алфавита.

Для английского языка *n* = 26, для русского – 33, для ASCII-кодов – 256. Юлий Цезарь использовал подобную систему для *k* = 3, откуда и пошло название данного шифра. Аналитически криптосистема Цезаря описывается выражением 1.1.

*Ek*(*i*) = (*i*+*k*) mod *n*. (1.1)

Например, в соответствии с приведенным выражением буква 'a'

исходного английского алфавита, имеющая номер *i* = 0, заменяется буквой 'D', имеющей номер (*i+k*) mod *n* = (0+3) mod 26 = 3, а буква 'z' (*i* = 25) заменяется буквой 'C', имеющей номер (*i+k*) mod 26 = (25+3) mod 26 = 2.

Алгоритм дешифрования имеет вид (1.2):

*Dk*(*i*) = (*i*+*n*–*k*) mod *n*.

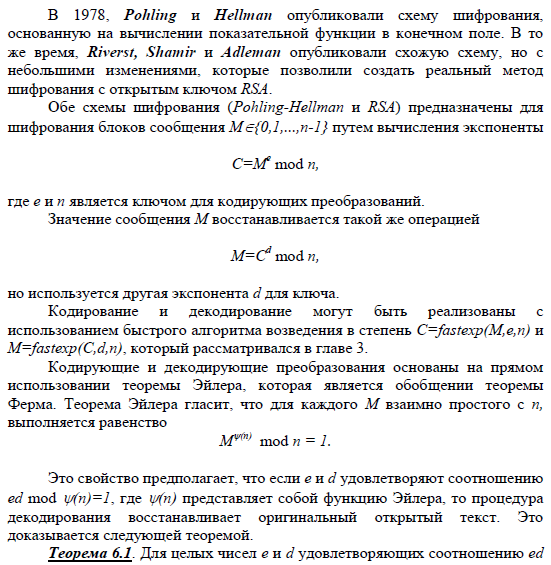
**8.2 Способы решения линейных сравнений: 3 случай (множество решений).**

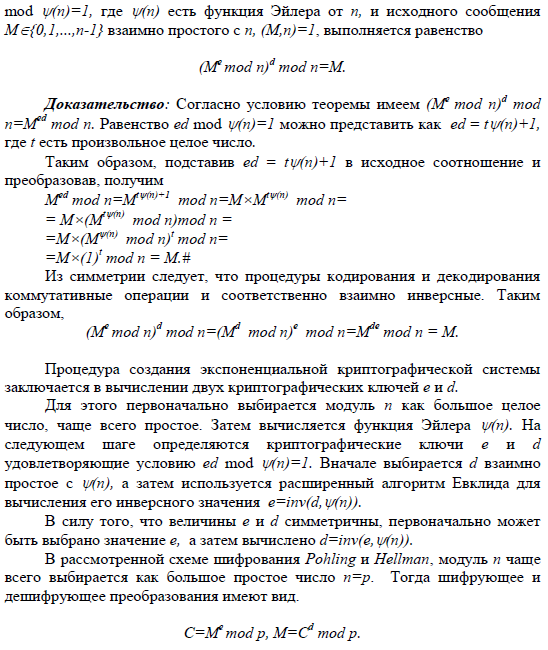
**Tеорема 3.18.** Если наибольший общий делитель d чисел a и n является делителем числа b (db), то существует d решений линейного сравнения вида ax=b **mod** n.

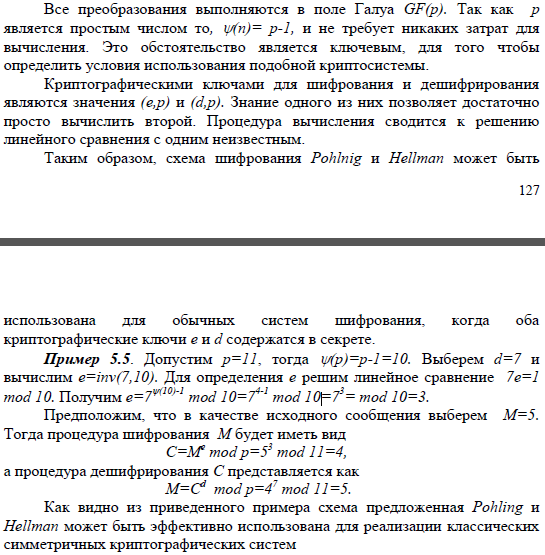
**Доказательство:** Согласно условию теоремы, d является делителем a, n и b. Тогда из линейного сравнения ax=b **mod** n получим a1dx=b1d **mod** n1d. Используя лемму 3.3, получим линейное сравнение вида a1x=b1 **mod** n1, где (a1,n1)=1. Данное сравнение a1x=b1 **mod** n1, имеет одно решение x0. Целые числа того же класса по модулю n/d будут решениями для исходного сравнения ax=b **mod** n. То есть x1=x0 **mod** n, x2=x0+n/d **mod** n, x3=x0+2n/d **mod** n,…, xd=x0 +((d-1)n)/d **mod** n.

**Пример 3.22.** Найти решения для следующего линейного сравнения 6x=4 mod 10. Принимая во внимание, что (6,10)=2 и 2 это делитель 4, получим сравнение 3x=2 mod 5. Решением последнего сравнения будет x0=ban-2 **mod** n=2×35-2 **mod** 5=2×33 **mod** 5=4. Тогда решениями сравнения 6x=4 mod 10 будут x1=x0 **mod** n=4 **mod** 10=4; x2=x0+n/d **mod** n=4+10/2 **mod** 10=9.

**8.3 Экспоненциальный шифратор (Pholing & Hellman).**

****

****

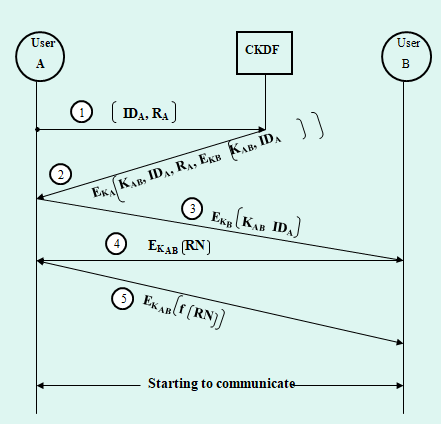
****

**8.4** **Распределение ключей (симметричные криптосистемы)**

Протокол распределения ключей для симметричных алгоритмов

Этот протокол может применяться к группе n пользователей для устранения проблемы создания и управлениявозможные пары ключей, которые были бы огромными.

Для реализации этого протокола для организации обмена данными между пользователями A и B. необходимо использовать центр централизованного распределения ключей (CKDF). Все пользователи должны иметь ключи для связи с CKDF. Пользователь A ключ KA и пользователь B - KB



Протокол распределения ключей для симметричных алгоритмов

1. Пользователь A отправляет соответствующий запрос RA вместе с идентификатором IDA A в CKDF в ясной форме.

2. CKDF отправляет пользователю A следующую криптограмму: EKA (KAB, IDA, RA, EKB (KAB, IDA)), где KAB является междоменной защитой ключа между пользователями A и B; KA и KB являются вторичными ключами связи между A и CKDF и между B и CKDF, соответственно; и EKA (KAB, IDA) является криптограммой KAB, IDA, полученной по секретному ключу B Ks.

3. Получив криптограмму, A расшифрует ее и проверит, идентичны ли идентификатор IDA и запрос RA оригиналу. Если это так, A пересылает криптограмму EKB (KAB, IDA) пользователю B.

4. B расшифровывает полученную криптограмму под ключом KB и восстанавливает четкую форму как KAB, так и IDA. В результате пользователи A и B могут иметь один и тот же ключ KAB, он инициирует процедуру аутентификации, чтобы исключить любую возможность ложной криптограммы , Таким образом, B направляет криптограмму EKAB (RN), то есть шифрование случайного числа RN с KAB, на A.

5. Дешифрует полученную криптограмму для получения RN и модифицирует ее некоторым предопределенным способом для получения f (RN) = (RN) ', который зависит от времени. Последовательность f (RN) зашифровывается под KAB и возвращает EKAB ((RN) ') пользователю B.

6. Если A подтверждает получение законного RN и использование KAB с рукопожатием, A может начать передачу сообщения B.

**Билет №9**

**9.1 Алгоритмы шифрования: Алгоритм децимации. Аффинное преобразование.**

Существуют более сложные методы подстановки. Шифраторы,

основанные на умножении номера каждого символа исходного текста на

значение ключа *k* (**метод децимации**), описываются следующим отношением

1.3:

*Ek*(*i*) = (*i*\**k*) mod *n*, (1.3)

где *n* и *k* должны быть взаимно простыми числами, т.е. у них не должно быть общих делителей кроме 1.

Комбинацией двух приведенных выше методов шифрования является

**афинное преобразование**, при котором уже используются два ключа:

*Ek*(*i*) = (*i*\**k*1*+k*2) mod *n*. (1.4)

Рассмотренные выше шифры простой замены легко взламываются с

помощью криптографических атак, основанных на анализе частот появления

символов в шифротексте. Так в естественном языке частота встречи букв

разная и некоторые из них встречаются чаще, чем другие, и это является

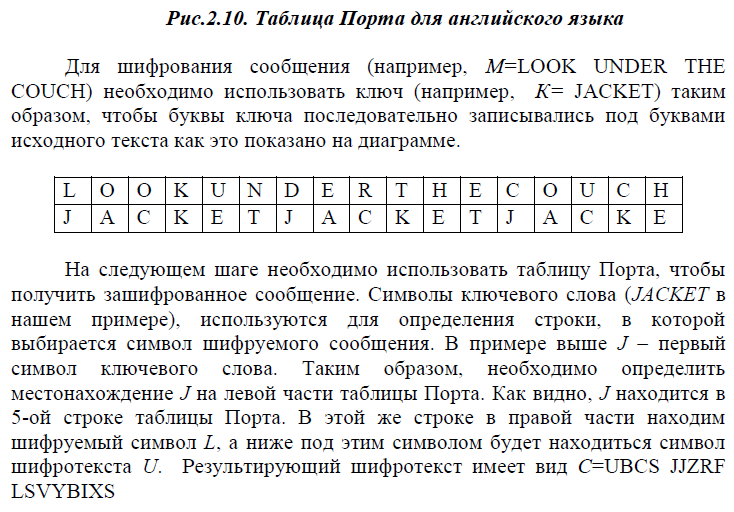
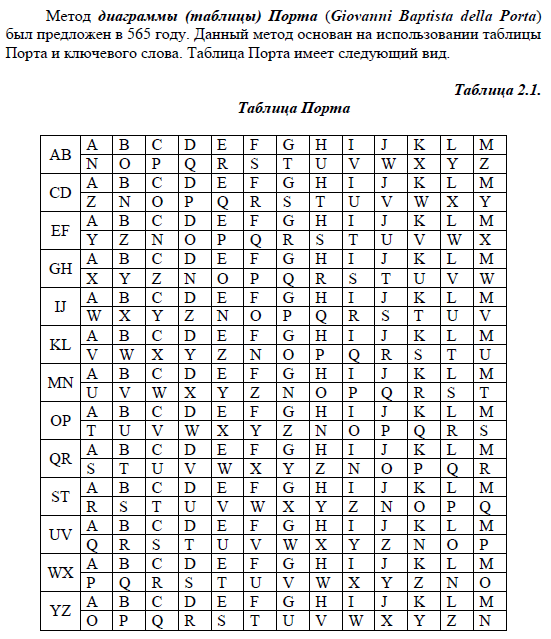
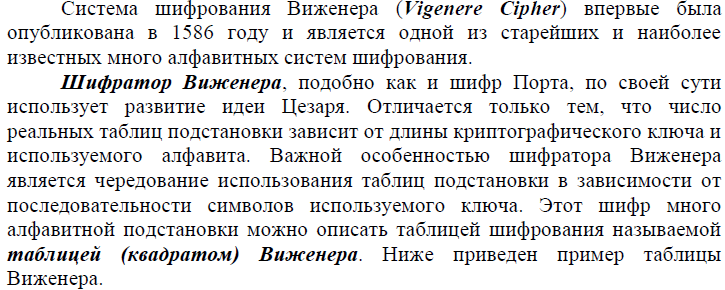
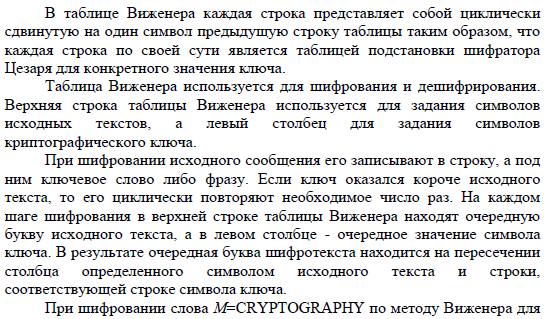
8 ключом для криптоаналитика. Проанализировав частоту встречи символов в шифротексте, можно сделать вывод о соответствии им символов исходного

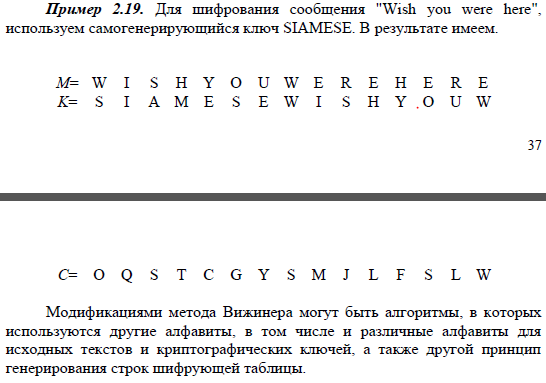
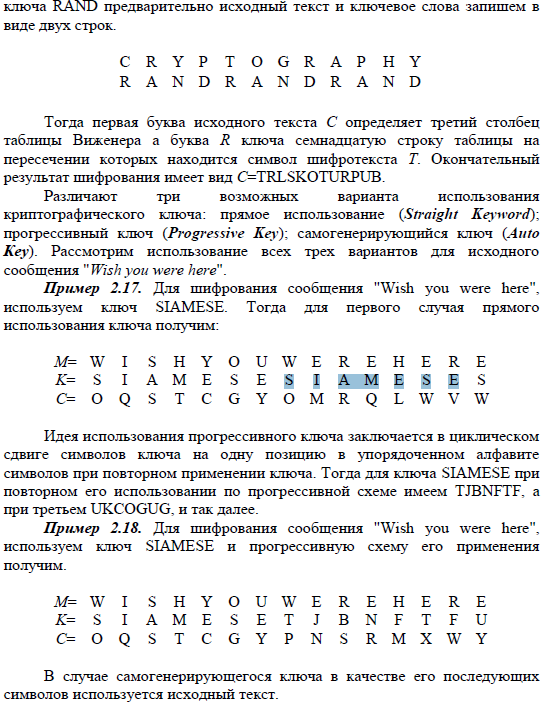
алфавита.

**9.2 Симметричные криптосистемы. Lucifer. DES.**

Теория.pdf – 64 стр.

**9.3 Шифратор на базе таблиц Порта. Шифратор Виженера**.

**Вижинер** 



**9.4 Алгоритм обмена ключами по открытому каналу с использованием эллиптических групп.**

*Эллиптическая открытая система распределения ключей (public key distribution algorithm).* Стойкость основана на сложн. вычисл. дискретного логарифма над эллиптич. группой EМ(а,b). Рассм. пару взаимно инверсных преобр-ний:Q=kG;k=logGQ (mod M) над группой EМ(а,b), где k – простое целое число, а G – генерирующая точка эллиптической группы EМ(а,b).

**Algo**:

1.Пользователь A случайным образом выбирает целое число nA, такое, что *nA*<*M*, которое является аналогом секретного ключа пользователя *A*. Значение числа *nA*должен знать только пользователь *A*. *A* генерирует свой открытый параметр (ключ) как результат скалярного произведения *nA* на *G*, то есть выполняет вычисление точки *PA=nAG*, которая является точкой эллиптической группы *EМ*(*а*,*b*)*.*

2.Пользователь *B* повторяет аналогичные действия и в результате получает свой секретный параметр *nB*и соответствующий ему открытый параметр *PB*.

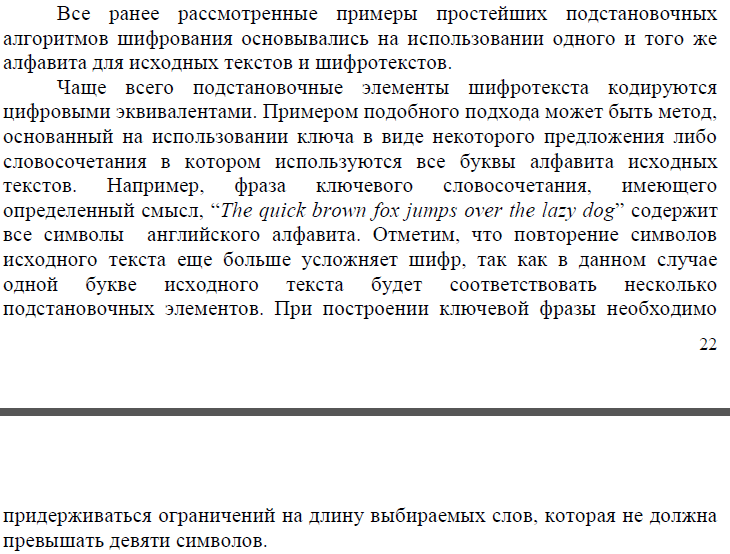
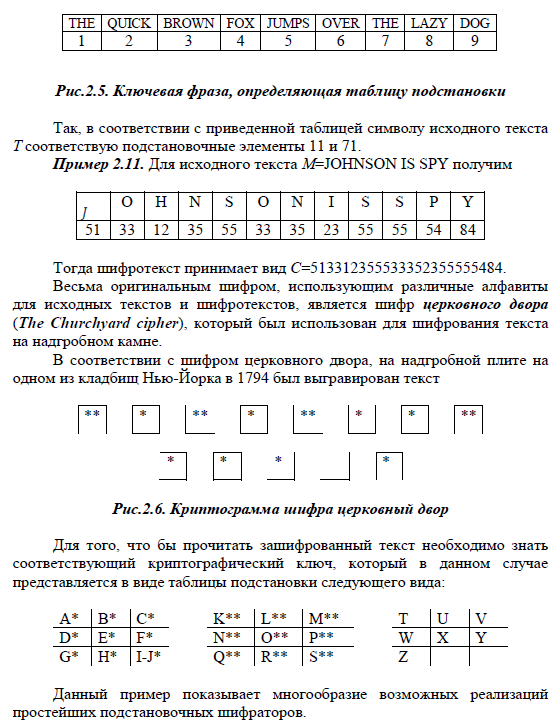
3.По открытому каналу юзер *A* высылает в адрес юзера *B* свой открытый параметр *PA*, а юзер *B*–свой параметр *PB* в адрес юзера *A*.

4.Пользователь *A* получает значение секретного ключа *K* как рез-т скалярного произведения *K=nAPB*. Аналогично пользователь *B* генерирует значение общего с пользователем *A* секретного ключа *K* как рез-т вычисления *K=nBPA*

В рез-те оба юзера получат одно и то же знач. ключа *K*из-за того,что для скалярного произвед-я целого числа на точку элл. гр. выполняется след. рав-во *nAPB*= *nA*(*nBG*)=*nB*(*nAG*)*=nBPA*.

**Билет №10**

**10.1 Шифраторы, использующие различные алфавиты. Шифр церковного двора.**

** **

**10.2 CPT схема. (Графическая стеганография).**

Схема CPT - это метод, предложенный Y.Y.Chen, H.K.Pan, Y.C.Tseng для безопасного скрытия данных для двухцветных изображений. Схема CPT является блочной. Двоичное изображение контейнера F регулярно разбивается на небольшие пиксельные блоки Fi размером m × n (для простоты мы предполагаем, что размер F кратен m × n). Эта схема способна скрыть до тех пор, пока не будет вычитано не более 2 битов в блоке r ≤ log2 (mn + 1)  бит секретного сообщения в каждом блоке хоста. Секретный ключ состоит из двух компонентов:

  K: случайная выбранная двоичная матрица размера m × n.

  W: весовая матрица, которая является целочисленной матрицей размера m × n. W удовлетворяет условию, что {[W] i, j | i = 1, ..., m; J = 1, ..., п} = {1,2, ..., 2r-1}. Заметим, что поскольку 2r ≤ 2^|log2 (mn + 1)| ≤ 2^log2 (mn + 1) ≤ mn + 1, тривиально оштрафовать матрицу, которая может служить весовой матрицей.

В самом деле, количество вариантов для W можно рассчитать как: C2r-1mn × (2r-1)! × (2r-1) mn- (2r-1) Например, если m = n = 8 и r = 5, есть C3164 × 31! × 3133 возможных W. Это число должно быть достаточно большим, чтобы предотвратить атаку грубой силы.

Скрытие данных достигается путем изменения некоторых бит F. Ниже (следующий слайд) мы показываем, как скрыть бит-поток b1b2, ..., br в хост-блок m × n, скажем Fi.

CPT Схема скрытия алгоритма.

S1. Вычислить FiK, где  означает бит-мудрый Exclusive-Or из двух равных размерных двоичных матриц. S2. Вычислить SUM ((FiK) W), где  означает парное умножение двух матриц равного размера, а SUM означает сумму всех элементов в матрице

S3. Из матрицы FiK вычислим для каждого w = 1, ... 2r-1 следующее множество: Sw = {(j, k) | ([W] j, k = w [FiK] j, k = 0 ([W] j, k = 2r-w [FiK] j, k = 1)}. Интуитивно, Sw - это множество, содержащее каждый индекс матрицы (j, k) такой, что если мы дополняем [ Fi] j, k, мы можем увеличить сумму на шаге S2 на w. На самом деле есть две возможности для этого:

(i) если [W] j, k = w и [FiK] j, k = 0, то, дополняя [Fi] j, k увеличит вес на w, а

(ii) если [W] j, k = 2r-w и [FiK] j, k = 1, то дополнение [Fi] j, k уменьшит вес на 2r-w или равномерно увеличит сумму на w (по mod 2r). Кроме того, определим Sw = S для любого w = w 'mod 2r.

S4. Определите разницу в весе d = (b1b2, ..., br) -SUM ((FiK) W) mod 2r. Если d = 0, нет необходимости менять Fi. В противном случае мы запускаем следующую программу для преобразования Fi в F'i.

Программа для преобразования Fi в F'i.

(hd mod 2r) + (- (h-1) d mod 2r) = d

a) Случайно выберем h {1, ..., 2r-1} такие, что Shd и S- (h-1) d. Индексы hd и - (h-1) d используются mod 2r. б) Случайно подберем a (j, k) Shd и дополним бит [Fi] j, k.

c) Случайно подберем a (j, k) S- (h-1) d и дополним бит [Fi] j, k.

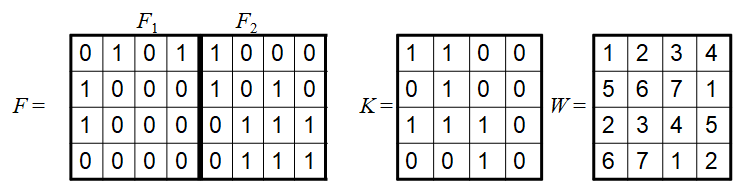
Подводя итог, приведенные выше шаги обеспечивают следующий инвариант



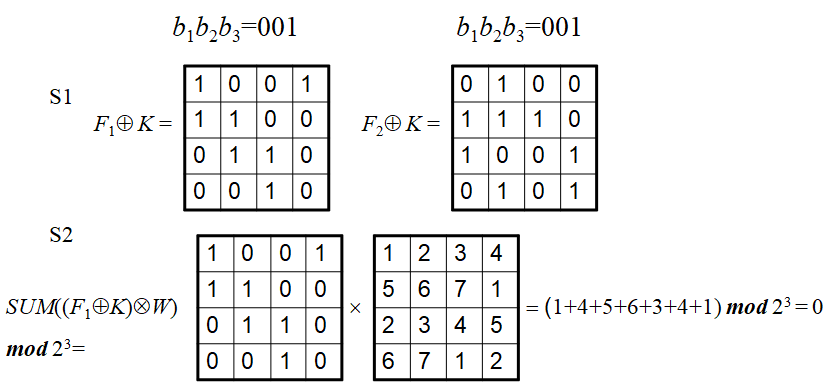
На предыдущем шаге S4 мы выбираем два непустых множества Shd и S- (h-1) d. Так как эти множества указывают места, где мы можем дополнить Fi, чтобы увеличить вес на hd и - (h-1) d соответственно. Общий эффект - увеличение веса на d. Доказано, что шаг S4 всегда будет успешным. Однако в вышеупомянутой программе есть логические недостатки, которые мы умышленно упустили для удобства презентации. Во-первых, множество S0 (и аналогично S2r и т. Д.) Еще не определено. Как и другие Sw, мы можем рассматривать S0 как набор индексов, таких, что дополнение этих местоположений в Fi приведет к увеличению веса на 0. Так как это может быть достигнуто путем изменения ничего на Fi, мы всегда можем считать S0 непустым и всякий раз, когда заявлении

Пример алгоритма скрытия схемы CPT

Например, пусть изображение хоста F, секретный ключ K и весовая матрица W показаны ниже



Во-первых, F разбивается на два блока 44, F1 и F2. Мы можем скрыть r ≤ log2 (4 × 4 + 1) 



S3. S1 = {(2,4)}; S2 = {(1,2), (3,1), (4,4), (2,2)}; S3 = {(1,3), (2,1)}; S4 = {(1,4), (3,3)}; S5 = {(3,4), (3,2)}; S6 = {(4,1)}; S7 = {(2,3), (4,2), (1,1), (4,3)}.

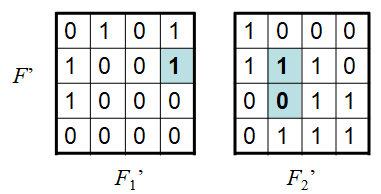
S4. Разница в весе d = (b1b2b3) -SUM ((F1K) W) = 001-0 = 1-0 = 1. Если d0, необходимо изменить F1. Мы запускаем следующую программу для преобразования F1 в F1 '.

a) Для любого h есть возможность использовать один Sw из всех возможных.

б) Случайно мы собираем a (j, k) = (2,4) и дополним бит [F1] 2,4 до нового значения 1.

c) Из-за [W] 2,4 имеет значение 1, что точно равно d = 1, мы пока не должны делать никаких изменений для F1.

Для F2, поскольку SUM ((F2K) W) mod 23 = (2 + 5 + 6 + 7 + 2 + 5 + 7 + 2) mod 23 = 4. Разница в весе d = 1- 4 = - 3 = 5. Таким образом, замена [F2] 2,2 и [F2] 3,2 увеличит вес на [W] 2,2 = -6 = 2 и [W] 3,2 = 3, соответственно. Результирующий F с модифицированными битами показан ниже.



**CPT схема** – метод для защищенного шифрования двухцветных изображений. Эта схема основана на основе блоков. Двоичный контейнер изображения F разбивается на мелкие блоки пикселей размера m x n. Секретный ключ имеет 2 компонента: 1.K: случайным образом выбирается двоичная матрица размера M × N 2.W: вес целочисленной матрицы m x n. Шифрование данных достигается путем изменения некоторых битов матрицы – мой максимум.

**10.3 Криптосистема типа «рюкзак» как криптосистема с секретным ключом.**

Криптографической односторонней функцией называется функция, отвечающая двум условиям:

1.Значение f(x) легко вычисляется на основании значения x.

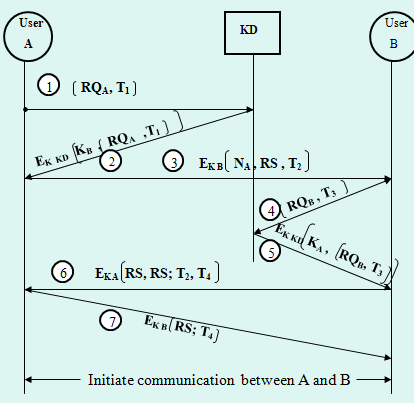
2.Определение x из f(x), вероятно, будет трудновычислимым. Одним из первых примеров односторонних функций является задача о рюкзаке (knapsack problem), которая может быть сформулирована следующим образом. Пусть задан набор K={k1,k2,k3,...,kn}, из n различных положительных чисел, а также еще одно положительное число C. Задача- нахождение таких ki , если возможно, сумма которых равна C.В простейшем случае C указывает размер рюкзака, а каждое из чисел ki- размер предмета, который может быть упакован в рюкзак. Задача- нахожд. такого набора предметов, чтобы рюкзак был полностью заполнен. + конспект.

**10.4 Протокол распределения ключей для асимметричных алгоритмов.**

Протокол распределения ключей для асимметричных алгоритмов

В асимметричных криптосистемах открытый ключ может быть отправлен через небезопасный канал, не подвергая опасности секретный ключ. Но любой пользователь, получающий ключ, должен проверить подлинность ключа.

Используя технику временного тиснения, давайте разработаем протокол распределения ключей между двумя пользователями A и B, принадлежащими к одному и тому же ключевому каталогу. Пусть KA - открытый ключ пользователя A и KB, который для пользователя B. Они зарегистрированы в Key Directory (KD). Предположим также, что открытый ключ KKD KD известен всем его пользователям. Протокол для установления распределения ключей между А и В следующий:

****

1. Пользователь A инициирует процедуру получения открытого ключа B, KB, отправив запрос RQA вместе с текущим временем T1 в ясной форме в KD.

2. Затем KD находит KB на основе запроса RQA и передает KB вместе с копией (RQA, T1) в A в виде криптограммы EKKD [KB, (RQA, T1)], зашифрованной под открытым ключа KD.

3. Знание KKD A может расшифровать криптограмму и извлечь аутентичный открытый ключ KB, копию запроса RQA и время T1. Пара (RQA, T1) затем сравнивается с оригиналом. Теперь, имея открытый ключ KB, A посылает B криптограмму, где NA является именем A, RS - случайная последовательность, выбранная A, а T2 - текущее время.

4. Получив криптограмму, B воссоздает четкую форму сообщений с использованием секретного ключа KS '. Поскольку B знает имя NA, B может запросить аутентичный открытый ключ A из KD, отправив соответствующий запрос RQB и текущее время T3.

5. В ответ KD передает на B криптограмму EKKD [KA, (RQB, T3)], зашифрованную под открытым ключом KD.

6. После расшифровки криптограммы B получает аутентичный открытый ключ A, но пару (RQB, T3) следует сравнить с оригиналом. Затем B создает криптограмму EKA (RS, RS ', T2, T4), зашифрованную под открытым ключом A, где случайная последовательность RS получается от пользователя A, случайная последовательность RS' является самогенерируемой, а T4 является текущей время. Впоследствии эта криптограмма отправляется А.

7. Дешифрует криптограмму и сравнивает (RS, T2) с оригиналом. Если эти последовательности совпадают, A уверен, что пользователь B является аутентичным. Наконец, A ретранслирует случайную последовательность RS 'в виде криптограммы EKB (RS', T4) в B.

8. B сравнить оригинал RS 'для подтверждения подлинности A. Затем запустите

**Билет №11**

**11.1 DES: получение итерационных ключей (+)**

**11.2 Модификация криптосистемы типа «рюкзак» до криптосистемы с открытым ключом.**

1. Генерируется секретный ключ Ks={k1\*,..,kn\*}, где ki\* > k1\* + .. + k(i-1)\*

2. Выбирается случайное секретное m > k1\* + … + kn\*

3. Выбирается случайное секретное w < m, (w,m) = 1

4. Вычисляется секретный параметр v = w^-1, vw = 1 mod m

5. Генерируется открытый ключ Kp = w\*Ks mod m, который используется для шифрования C = M \* Kp

Криптограмму расшифровать можно только ключом Ks. Для этого нужно преобразовать C в C\* = vC mod m и расшифровывать по принципу криптосистемы “рюкзак” II.

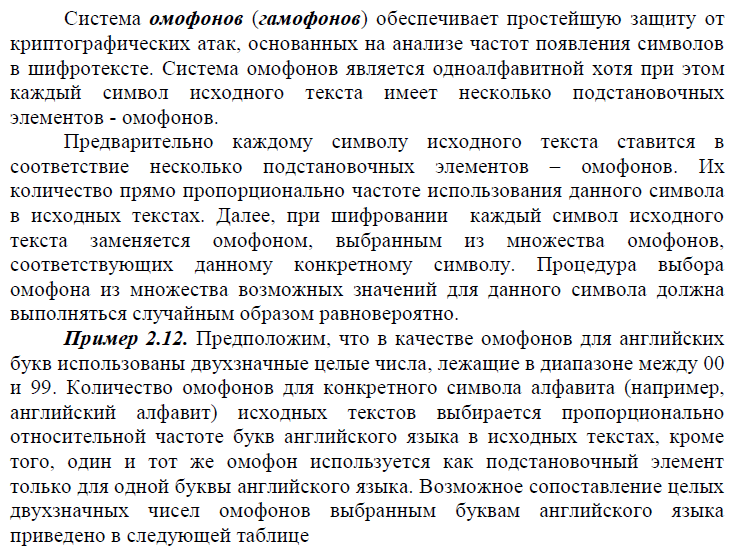
**11.3 Квантовая криптография принцип неопределенности Гейзенберга. Квантовая теорема неклонируемости.**

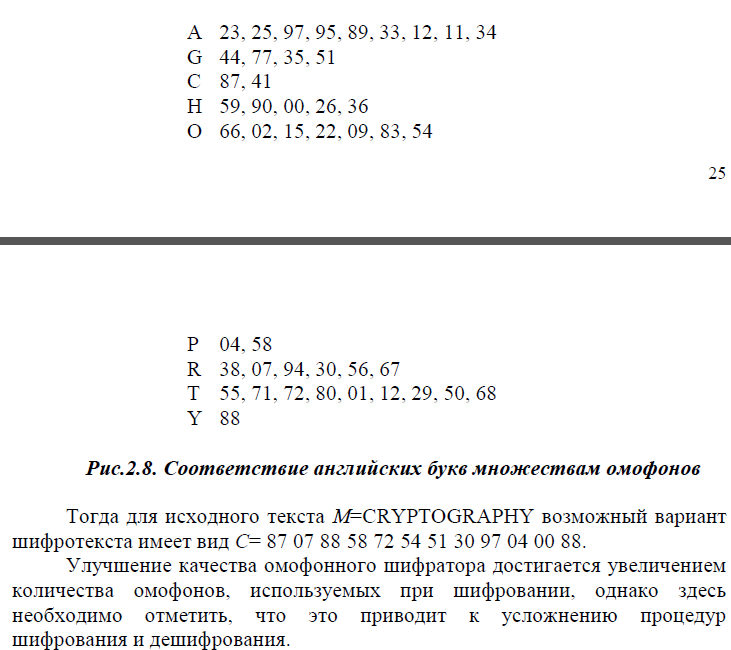
**Квантовая криптография** — метод защиты коммуникаций, основанный на принципах квантовой физики. В отличие от традиционной криптографии, которая использует математические методы, чтобы обеспечить секретность информации, квантовая криптография сосредоточена на физике, рассматривая случаи, когда информация переносится с помощью объектов квантовой механики. Процесс отправки и приёма информации всегда выполняется физическими средствами, например, при помощи электронов в электрическом токе, или фотонов в линиях волоконно-оптической связи.

**Принцип неопределённости Гейзенберга**:Невозможно одновременно получить координаты и импульс частицы, невозможно измерить один параметр фотона, не исказив другой.

**Квантовая теорема неклонируемости**:Неизвестное квантовое состояние нельзя клонировать.

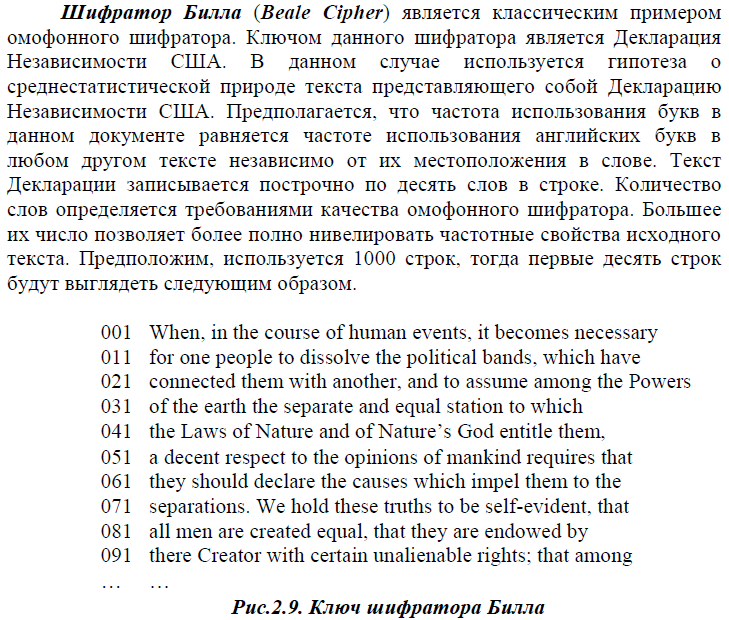
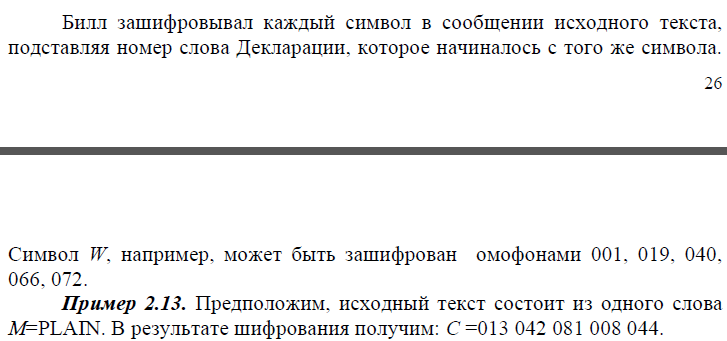
**11.4 Гомофонные шифраторы (Омофонные шифраторы).**



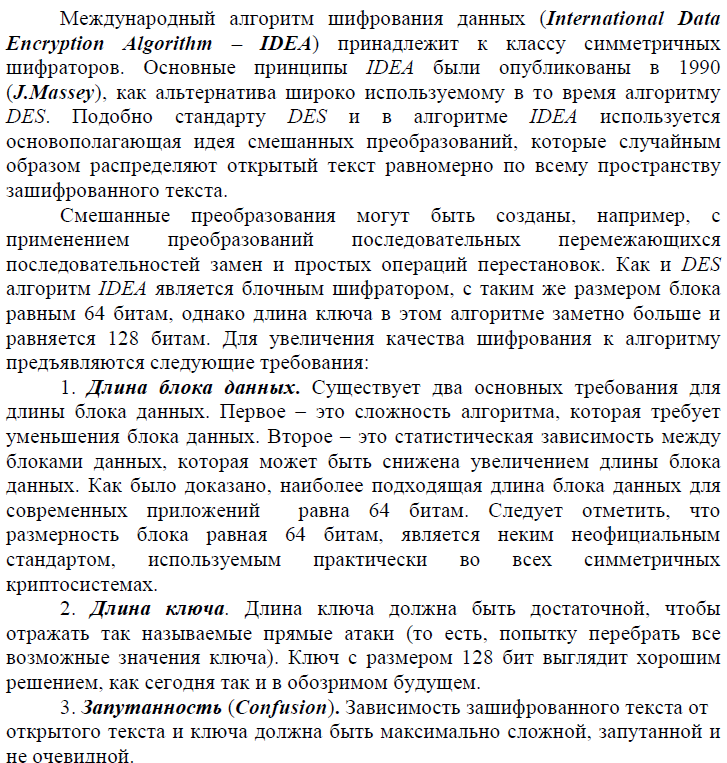


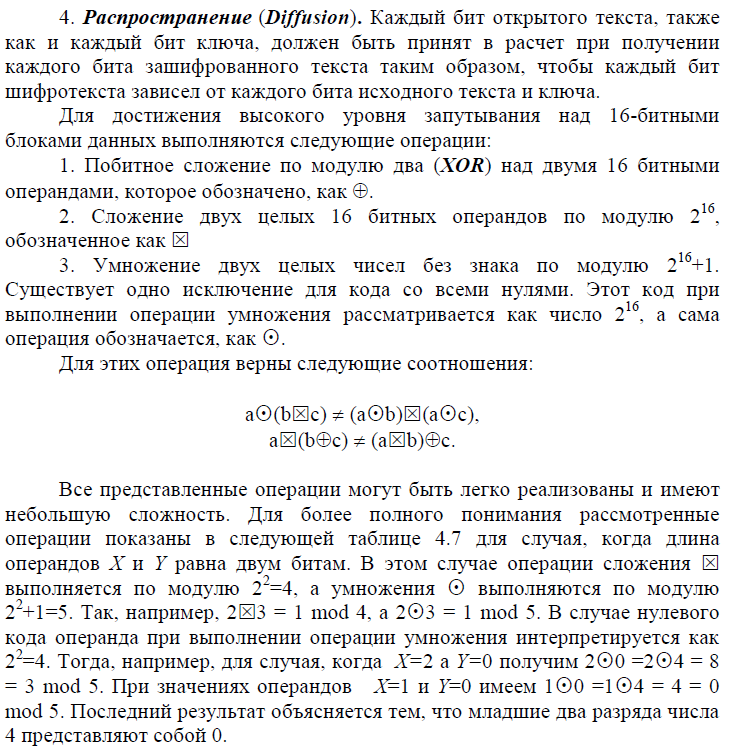
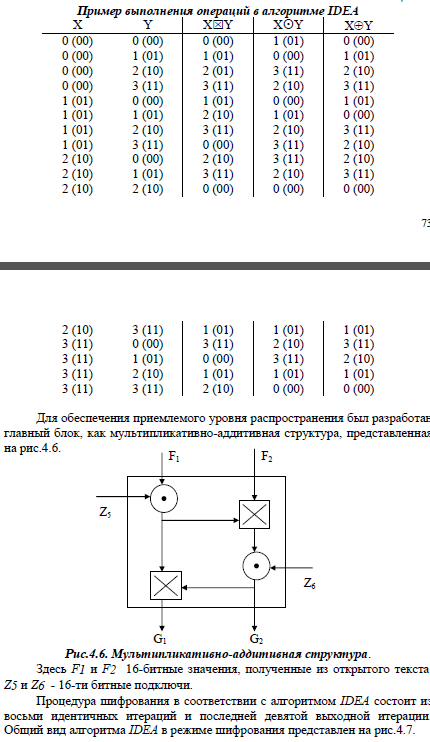
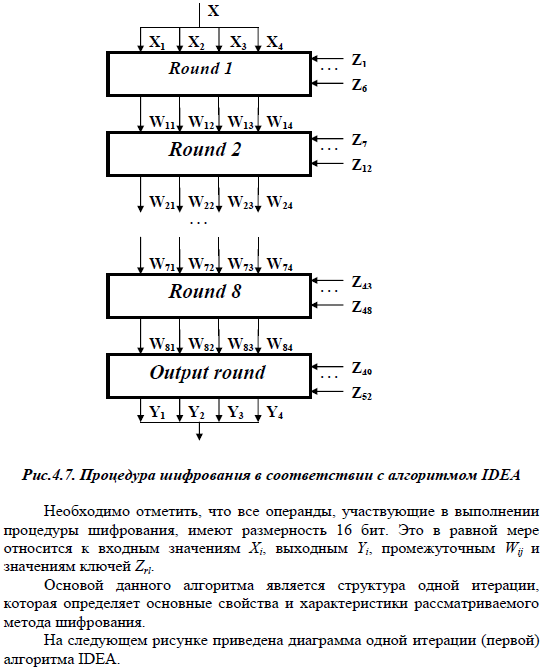
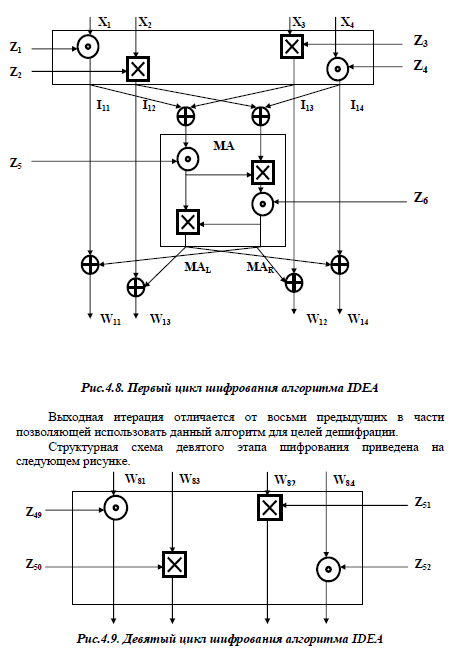
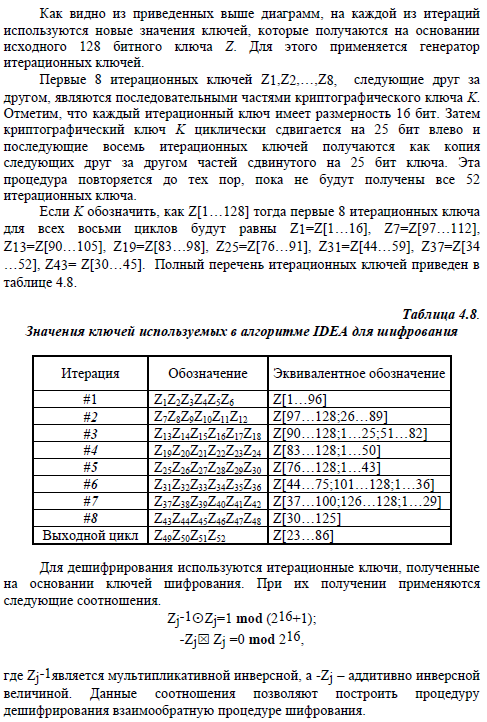
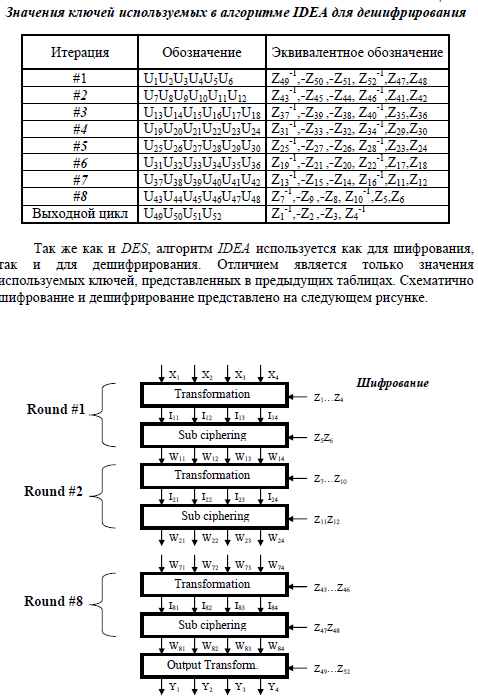
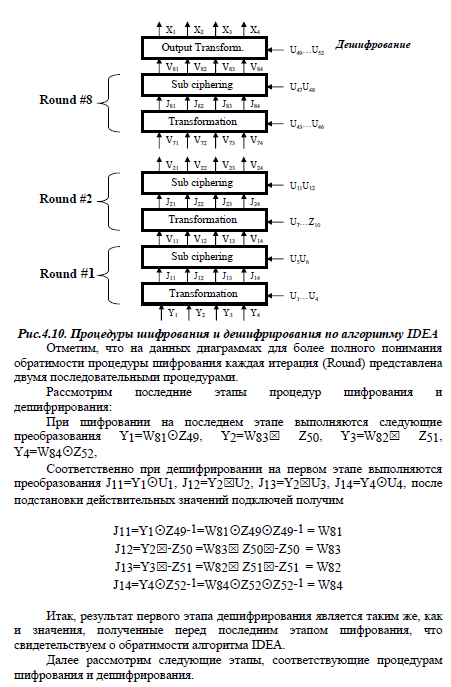
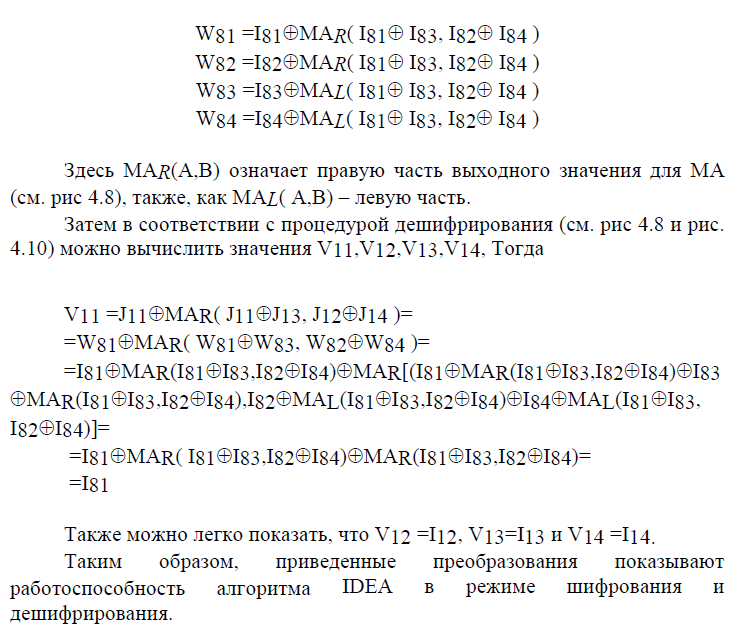
**Билет №12**

**12.1 Шифратор Билла (Beale).**

** **

**12.2 Международный алгоритм шифрования данных (IDEA).**

****

**       **

**12.3 Электронная цифровая подпись на основе RSA.**

Первой и наиболее известной во всем мире конкретной системой

электронной цифровой подписи стала система RSA, математическая схема

которой была разработана в 1977г. в Массачусетском технологическом

институте США.

Для формирования подписи по алгоритму RSA сначала необходимо

вычислить пару ключей: секретный ключ и открытый ключ, как это делается

для криптосистемы RSA:

1. Выбираются два случайных простых числа *p* и *q* таких, что *p* ≈ *q*.

2. Вычисляется их произведение *r* = *p*\**q*.

3. Вычисляется функция Эйлера для *r φ*(*r*) = (*p*–1)\*(*q*–1).

4. Выбирается открытая экспонента *e* такая, что 1<*e*<*φ*(*r*) и (*e*, *φ*(*r*)) = 1.

5. Вычисляется секретная экспонента *d*, удовлетворяющая условию (*e*\**d*)

mod *φ*(*r*) = 1.

Пару значений *Ko*=(*e*, *r*), которая является открытым ключом, автор

передает партнерам по переписке для проверки его цифровых подписей.

Значение *Kc* = (*d*, *r*) сохраняется автором как секретный ключ подписи.

Если отправителю необходимо подписать сообщение *М* перед его

отправкой, он сжимает сообщение *М* с помощью хеш-функции *h* в целое число

*m*: *m* = = *h*(*М*). Затем вычисляет цифровую подпись *S* под электронным

документом *М* на основе хеш-образа *m* и секретного значения *d*:

*S* = *md* mod *r*. (3.3)

Пара (*М*, *S*) передается получателю как электронный документ *М*,

подписанный цифровой подписью *S*, причем подпись *S* сформирована

обладателем секретного ключа (*d*, *r*).

После приема пары (*М*’, *S*) получатель вычисляет хеш-образ сообщения

*М*’ двумя различными способами. Прежде всего, он восстанавливает хеш-образ

25

*m*, применяя криптографическое преобразование подписи *S* с использованием

открытого ключа (*e*, *r*):

*m* = *Se* mod *r*. (3.4)

Кроме того, он находит результат хеширования *m*’ принятого сообщения

*М*’ с помощью такой же хеш-функции *h*: *m*’ = *h*(*М*).

Если вычисленные значения совпадают, т. е. *h*(*М*)= *Se* mod *r*, то

получатель признает пару (*М*’, *S*) подлинной.

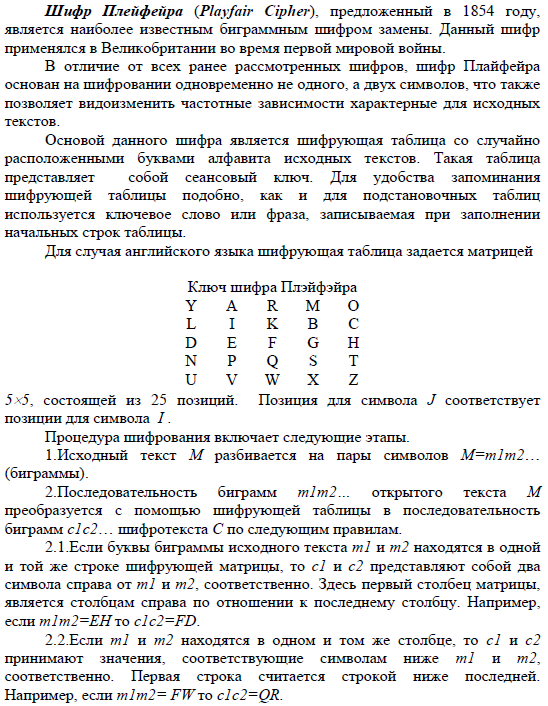
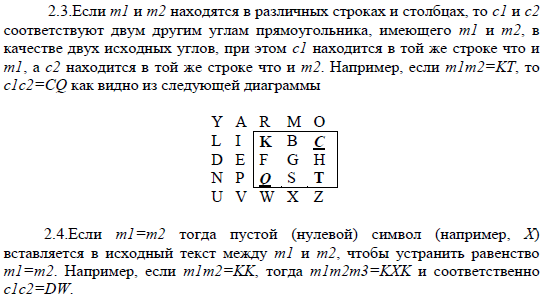
**12.4 Протокол ВВ84. Пример реализации протокола**

Пример реализации протокола. Двухканальная квантовая криптография. Протокол [использует 4 квантовых состояния(|=0 и -=1, \=0 и /=1)образующих 2 базиса(+, X), например поляризационные состояния света, для формирования секретного симметричного ключа.]

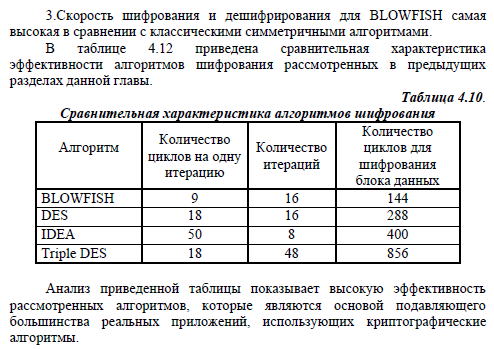
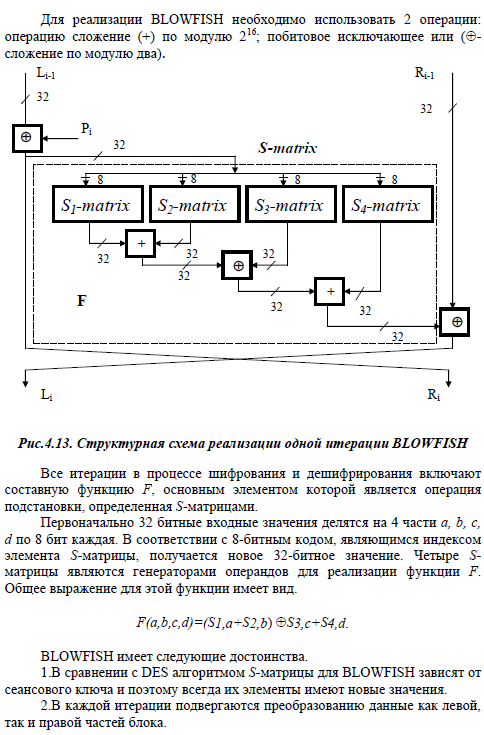
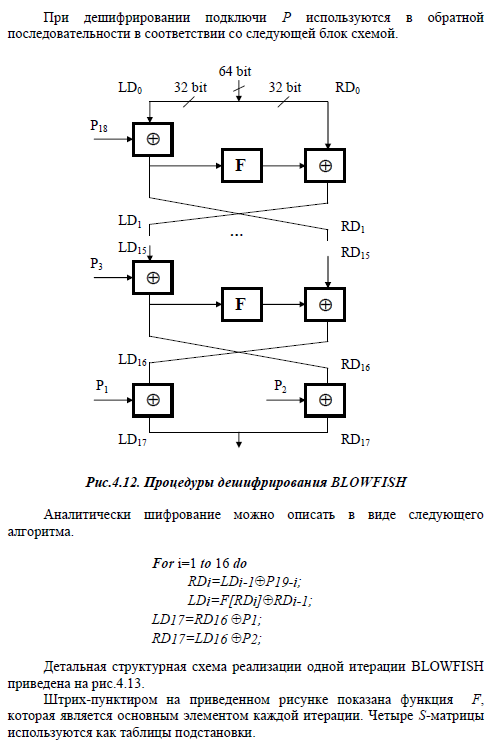
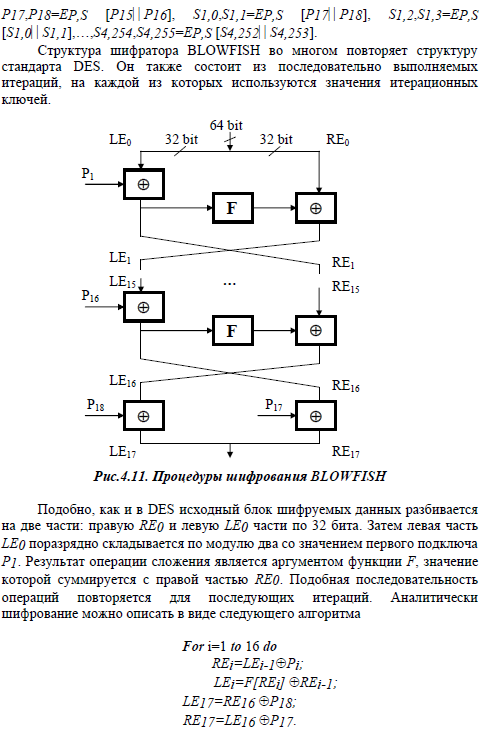
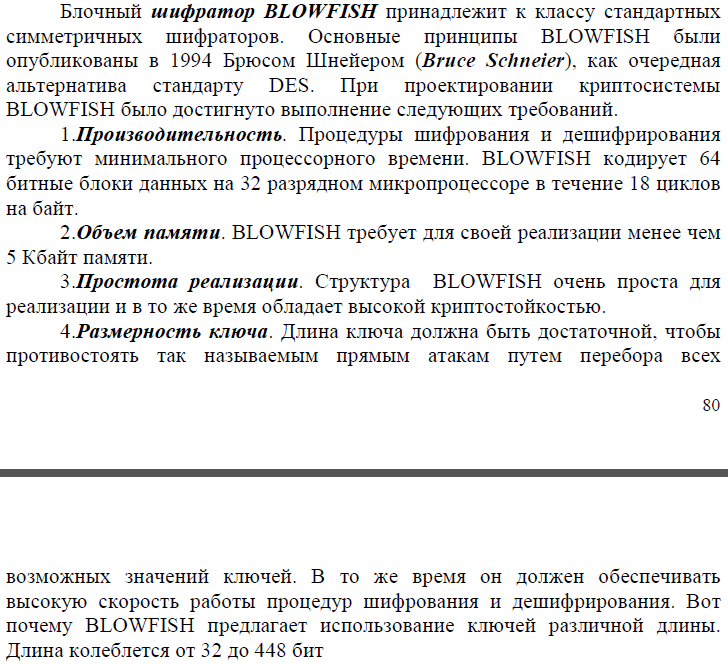
**Алгоритм**: Алиса посылает кванты в произвольно выбраных базисах. Боб измеряет эти кванты в одном из 2 базисов, выбранных случайно(независимо от А). Если базисы совподают, то значение у Боба и Алисы совпадёт. Теперь у Боба есть первичный ключ(~25% ошибок). После, по открытому каналу Боб передаёт испульзуемые им базисы(не раскрывая значений квантов), а Алиса говорит Бобу где он угадал базисы. В итоге Алиса и Боб, отбрасывая неугаданные Бобом базисы, получают просеянный ключ(обычно устраняется ~50% длины ключа). Теперь у Алисы и Боба есть один и тот же набор бит, который в дальнейшем может быть использован в симметричных криптосистемах. Если Ева перехватывает кванты, то их значения случайно меняются и кол-во ошибок в ключе Боба существенно увеличится, что легко будет обнаружить. Двухканальная квантовая криптография - [] Алиса посылает кванты по 2 каналам в противопроложных базисах. Боб принимает кванты из 2 каналов в одном случайном базисе. Если Боб не угадал базис для 1 канала, он угадал базис 2 канала и наоборот. Боб говорит Алисе свои базисы, Алиса говорит где Боб угадал, и они могут сфомировать симм. секретный ключ без потери битов, как в протоколе BB84.

**Билет №13**

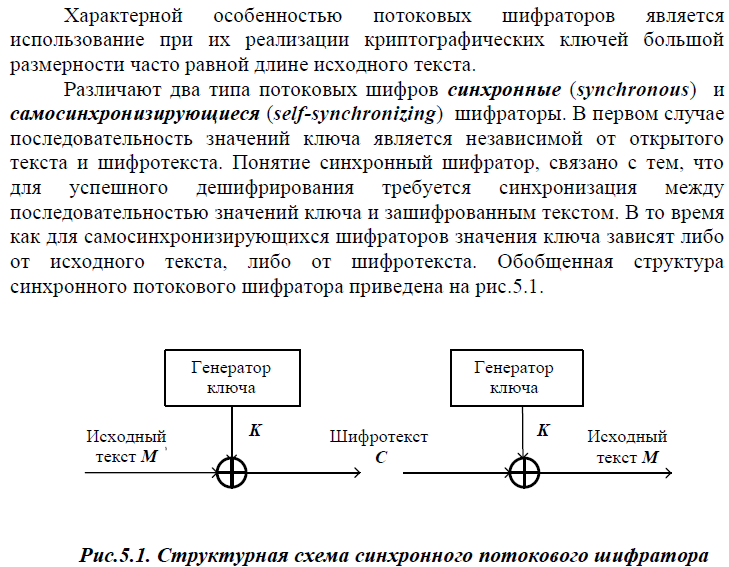
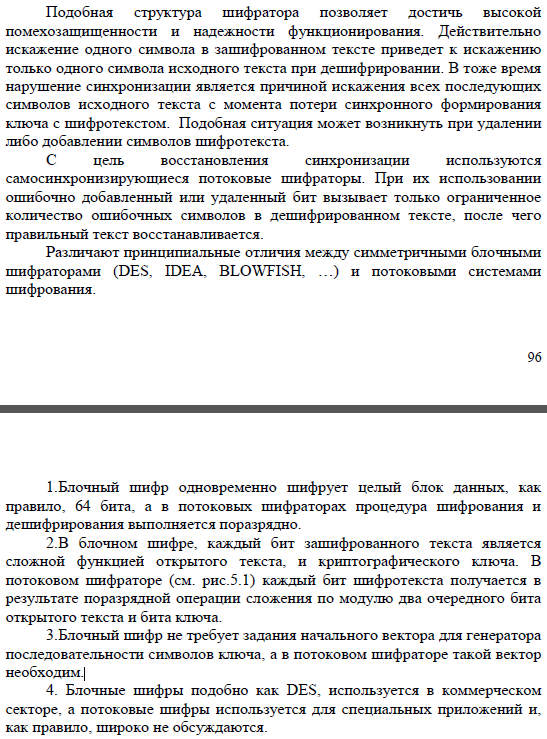
**13.1 Шифратор Playfair.**

** **

**13.2 Алгоритм BlowFish.**



**13.3 Потоковые системы шифрования.**

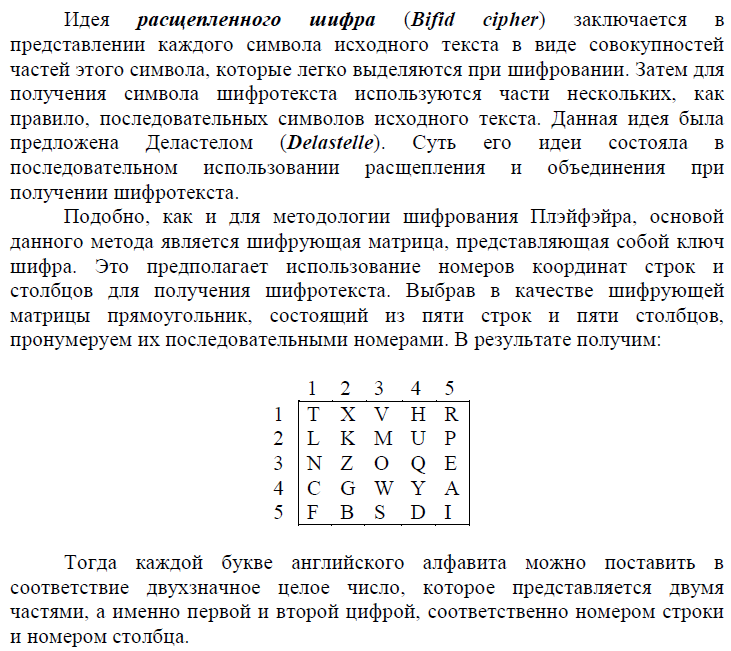
**  
**

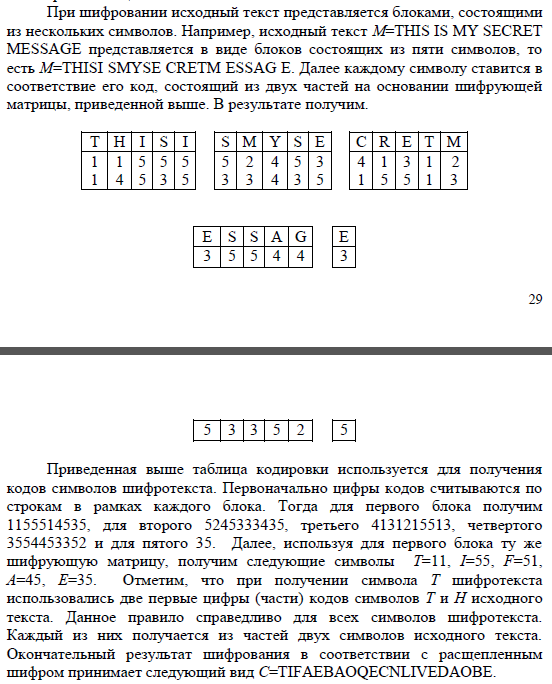
**13.4 Атаки на квантовые криптосистемы (Man in the Midle).**

При подаче мощного светового импульса на лавинный фотодиод детектора приводит к тому, что он перестает реагировать на отдельные фотоны и начинает выдавать выходной сигнал только при регистрации мощного светового импульса. Это дает возможность проведения атаки типа Intercept-Resend, позволяющей перехватить отправленный заново отправителем криптографический ключ целиком и дешифрировать информацию.

**Билет №14**

**14.1 Расщепленный шифр. Bifid шифр.**

****

****

**14.2 Процедура коррекции сырого ключа в ВВ84. Протокол Беннета.**

BB84-первый протокол квантового распределения ключа.Носителями информации являются 2х уровневые системы. Протокол использует 4 квантовые состояния, образующие 2 базиса, например, поляризационные состояния света. Сначала А посылает фотоны Б в произвольном базисе. Б измеряет принимаемые фотоны в одном из двух базисов. Если исп.одинаковые базисы, то - коррелированные результаты. Иначе- некоррелированные. Б получает строку битов с~25 % ошибок (первичный ключ). Можно провести согласование базисов: для каждого переданного состояния. Б открыто сообщает, в каком базисе проводилось измерение ку бита. А сообщает, в каких случаях её базис совпал с базисом Б. Если базисы совпали, бит оставляют, если нет, его игнорируют. Оставшийся ключ -просеянный.

**Протокол Беннета**

1. Отправитель и получатель договариваются о произвольной перестановке битов в строках, чтобы сделать положения ошибок случайными.
2. Строки делятся на блоки размера k (k выбирается так, чтобы вероятность ошибки в блоке была мала).
3. Для каждого блока отправитель и получатель вычисляют и открыто оповещают друг друга о полученных результатах. Последний бит каждого блока удаляется.
4. Для каждого блока, где четность оказалась разной, получатель и отправитель производят итерационный поиск и исправление неверных битов.
5. Чтобы исключить кратные ошибки, которые могут быть не замечены, операции пунктов 1-4 повторяются для большего значения k.
6. Для того чтобы определить, остались или нет необнаруженные ошибки, получатель и отправитель повторяют псевдослучайные проверки:

* Получатель и отправитель открыто объявляют о случайном перемешивании позиций половины бит в их строках.
* Получатель и отправитель открыто сравнивают четности. Если строки отличаются, четности должны не совпадать с вероятностью 1/2.
* Если имеет место отличие, получатель и отправитель, использует двоичный поиск и удаление неверных битов.

1. Если отличий нет, после m итераций получатель и отправитель получают идентичные строки с вероятностью ошибки 2-m.

**14.3 Цифровая подпись. Цифровая подпись, основанная на симметричных алгоритмах. (ЭЦП Diffie & Lamport).**

Пусть у [Алисы](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B8%D1%81%D0%B0_%D0%B8_%D0%91%D0%BE%D0%B1) есть 256-битная [криптографическая хеш-функция](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) и [криптографически стойкий генератор псевдослучайных чисел](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8_%D1%81%D1%82%D0%BE%D0%B9%D0%BA%D0%B8%D0%B9_%D0%B3%D0%B5%D0%BD%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_%D0%BF%D1%81%D0%B5%D0%B2%D0%B4%D0%BE%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB). Она хочет создать и использовать пару ключей Лэмпорта — [секретный ключ](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87) и соответствующий ему [открытый ключ](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87).

**Создание пары ключей**

Чтобы создать секретный ключ, Алиса использует генератор случайных чисел для получения 256 пар случайных чисел (всего 2×256 чисел). Каждое число состоит из 256 бит, то есть общий размер равен 2×256×256 бит = 16 [КиБ](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%B1%D0%B8%D0%B1%D0%B0%D0%B9%D1%82). Эти числа будут секретным ключом Алисы, и она будет хранить их в секретном месте, чтобы использовать в дальнейшем.

Чтобы создать открытый ключ, Алиса хеширует каждое из 512 чисел секретного ключа, таким образом получая 512 хешей по 256 бит каждый. Эти 512 хешей составляют открытый ключ Алисы, который она публикует.

**Подпись сообщения**

Теперь Алиса хочет подписать сообщение. Для начала она хеширует сообщение и получает 256-битный хеш. Затем, для каждого бита в этом хеше, она берёт соответствующее число из секретного ключа. Если, например, первый бит в хеше сообщения равен нулю, она берёт *первое* число из первой пары секретного ключа. Если же первый бит равен единице, она использует *второе* число из первой пары. И так далее. В итоге получается 256 случайных чисел, размер которых составляет 256×256 бит = 8 КиБ. Эти числа и составляют подпись, которую Алиса отправляет вместе с сообщением.

Стоит обратить внимание, что после того как Алиса использовала свой секретный ключ, он никогда не должен быть использован снова. Остальные 256 чисел, которые она не использовала в подписи, Алиса никогда не должна публиковать или использовать. Предпочтительно, чтобы она их удалила, так как иначе кто-то может получить к ним доступ и сгенерировать с их помощью поддельную подпись.

**Проверка подписи**

Боб хочет проверить подпись, которым Алиса подписала сообщение. Он также хеширует сообщение и получает 256-битный хеш. Затем для каждого бита в этом хеше он выбирает число из открытого ключа Алисы. Эти числа выбираются по такому же принципу, по какому Алиса выбирала числа для составления подписи. То есть, если первый бит хеша сообщения равен нулю, Боб выбирает первое число из первой пары в отрытом ключе. И так далее.

Затем Боб хеширует каждое из 256 чисел из подписи Алисы и получает 256 хешей. Если эти 256 хешей в точности соответствуют 256 хешам, которые он только что получил из открытого ключа Алисы, Боб считает подпись подлинной. Если не соответствуют — то фальшивой.

Стоит обратить внимание, что до того, как Алиса опубликует подпись к сообщению, никто не знает 2×256 случайных чисел в секретном ключе. Таким образом, никто не может создать правильный набор из 256 чисел для подписи. После того, как Алиса опубликует подпись, никто всё ещё не знает остальные 256 чисел, и, таким, образом, не может создать подпись для сообщений, имеющих иной хеш.[[2]](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C_%D0%9B%D1%8D%D0%BC%D0%BF%D0%BE%D1%80%D1%82%D0%B0#cite_note-_1573d68cb3035a01-2)

**14.4 Физическая криптография. Физически неклонируемые односторонние функции. PUF типа Arbitr.**

**Физическая криптография** - это другой и очень недавний подход к криптографии и безопасности, основанный на присущей сложности наномасштабных электронных и фотонных систем.

Р. Паппу, Б. Рехт, Дж. Тейлор, Н. Гершенфельд, Физические односторонние функции, Наука, вып. 297, стр. 2026-2030, 20 сентября 2002 года.

**Основная категория** - физическая криптография - это физическая функция Unclonable (PUF), которая сначала была вызвана как физическая односторонняя функция (POWF), физические случайные функции (PRF). В то время как два последних термина были впервые введены, сегодня в основном используется выражение Physical Uncloneable Function или PUF. Понятие физической односторонней функции было определено Р. Паппу. Он определенно принадлежит к достоинству быть первой попыткой формализации в этой области. Одно из широко используемых теперь определений PUF было предложено Tuyls et al.

**Определение 1**. Физические незащищенные функции состоят из неотъемлемо неконфигурируемых физических систем. Они наследуют свою непримиримость от того факта, что они состоят из множества случайных компонентов, которые присутствуют в производственном процессе и не могут контролироваться. Когда к системе применяется стимул, он реагирует с ответом. Такая пара стимулов (вызов) C и ответ R называется парой «вызов-ответ» (CRP). В частности, PUF рассматривается как функция, которая отображает проблемы для ответов.

В соответствии с этим определением физические неуязвимые функции или PUF были введены в качестве объектов ответа на вызов, которые неразрывно встроены в физическую систему, возможно, в кремниевую интегральную схему. Физическая система, явно или неявно, создается таким образом, что она содержит неуправляемые случайные элементы.

U.Ruhrmair сделал более общее определение PUF как систем с высоким уровнем информации (SHIC).

**Определение 2**. Физические неклонируемые функции представляют собой сложные неупорядоченные физические системы с чрезвычайно большим количеством структурной информации, которые удовлетворяют следующим условиям:

1. Информационное содержимое системы может быть надежно и многократно извлечено путем измерения с помощью различного вызова Ci и получения полученного ответа (ответа) Ri.

2. Количество возможных проблем C настолько велико, что значения всех соответствующих ответов R не могут быть определены для всех возможных проблем C в течение ограниченного времени.

3. Из-за высокого информационного содержания в системе также невозможно моделировать, вычислить, изучать, моделировать или каким-либо иным образом предсказывать пару с запросом-ответом (Cj, Rj) на основе известной пары (Cl, Rk).

4. Не обязательно трудно физически воспроизводить или клонировать PUF как физические системы.

Физическая неуглотимая функция (PUF) [Pappu, Gassend et al 2002]

Безопасность PUF основана на

-проволочные задержки

-задержки ворот

-квантовые механические колебания

Характеристики PUF

-уникальность

-надежность

-непредсказуемость

Предположения PUF

-Невозможно точно моделировать PUF

-Вероятность возникновения столкновения PUF-пары постоянна

-Физическое вмешательство изменяет PUF

-Из-за случайных вариаций процесса никакие две интегральные схемы даже с одинаковыми макетами не идентичны

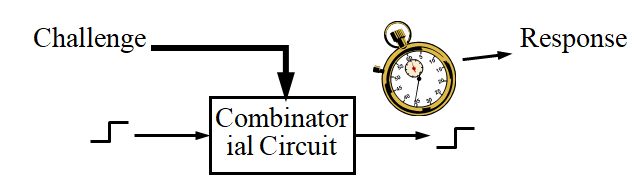
-Вариация присуща процессу изготовления

-Трудно удалить или предсказать

Относительные изменения возрастают по мере продвижения процесса изготовления

Концепция кремния PUF на основе задержки

Генерировать секретные ключи из уникальных характеристик задержки каждого процессора

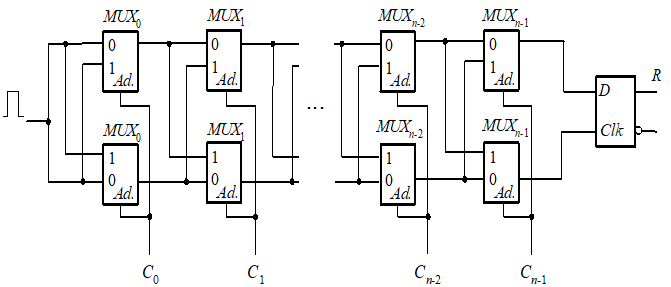


**Реализация PUF для кремниевых интегральных схем**

Первые PUF, встроенные в кремниевые устройства, которые были предложены, основаны на измерениях задержки цифровых путей. Из-за вариаций процесса два идентично разработанных цифровых пути на двух микросхемах или в двух разных местах на одной и той же ИС не будут иметь точно такой же задержки от их ввода до их выхода. Было предложено очень хорошо известный метод измерения вариаций случайных задержек, называемых Arbiter PUF.

**Арбитры PUF**

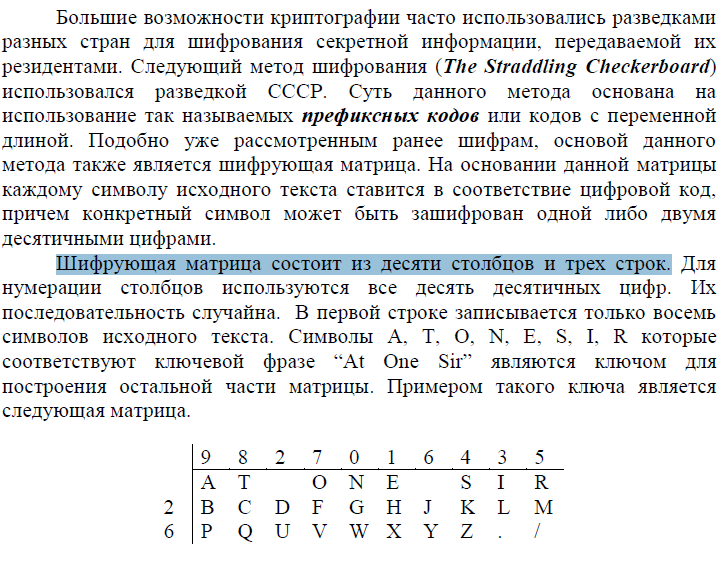
Согласно Арбитру PUF, состояние гонки вводится путем подачи двух симметричных путей в арбитр. Два одновременных импульса на входах путей, как правило, не будут поступать на выходы одновременно из-за небольших случайных колебаний задержки. Арбитр решает, какой из двух путей был самым быстрым.

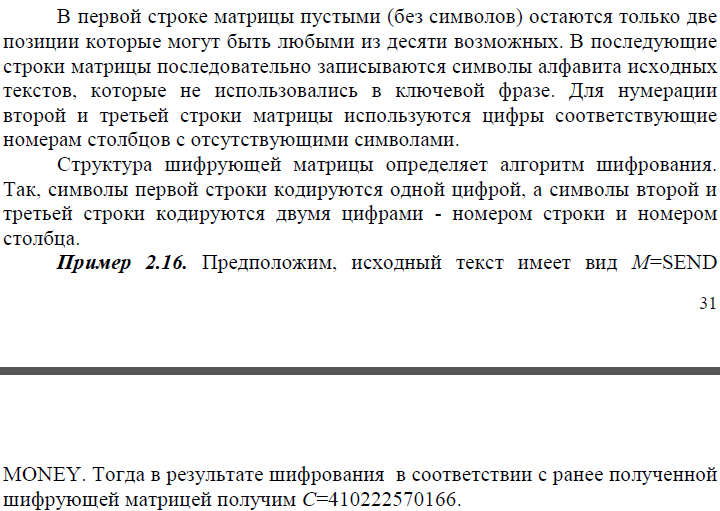


Контрольные биты C определяют пути задержки путем управления MUX. Чтобы оценить реакцию R для конкретной задачи C, восходящий сигнал подается на входы обоих трактов в одно и то же время, сигналы проходят через два пути задержки и арбитр (D-защелка) в конце двух путей решает, какой сигнал работает быстрее.

**Билет №15**

**15.1 Шифры, использующие коды переменной длины.**

****

****

**15.2 Генераторы криптографического ключа для потоковых криптосистем. Генераторы М-последовательностей. Достоинства и недостатки.**

Теория – 97 стр.

**15.3 Цифровая подпись, основанная на симметричных алгоритмах. (ЭЦП Matyas & Meyer)**

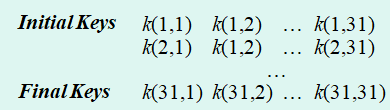
1. Фаза подготовки

Цифровая подпись Matyas и Meyer представляет собой список из 31 криптографических ключей, которые выбираются из матрицы 31х31 секретных ключей

Строка 1 матрицы содержит 31 начальный секретный ключ, обозначаемый k (1,1), k (1,2), ..., k (1,31), которые создаются через стандартный генератор псевдослучайных чисел. Например, начальные ключи для n-го сообщения могут быть получены с помощью алгоритма DES с использованием одного секретного ключа K через отношение

к (1, к) = EK (31 (п-1) + J),

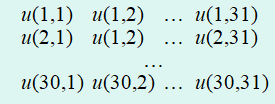
где j - номер столбца, и n номер сообщения.



В дополнение к ключевой матрице следующая матрица 30 × 31 несекретных кодовых слов создается через генератор псевдослучайных чисел по соотношению

u (I, J) = Еu (312 (n-1) +31 (I-1) + J),

где i - номер строки, j - номер столбца, n номер сообщения, U - несекретный ключ семени.



Затем ключи в строках с 2 по 31 получают с помощью повторяющихся шагов шифрования с использованием начальных ключей

k (i + 1, j) = Ek (i, j) (u (i, j)), i = 1,2, ..., 30, j = 1,2, ..., 31.

Ключ в строке 31 матрицы называется заключительными ключами. Эти заключительные ключи, которые заранее подготовлены отправителем и отправляются получателю, представляют собой шаблон проверки. Несекретные ключи a1, a2, ..., a31 могут быть установлены универсальной константой, которые записываются в общий реестр.

2. Генерация подписи

Первым шагом в формировании подписи является получение сжатого кодирования сообщения M, которое должно быть подписано. Для иллюстрации предположим, что функция Хэша имеет значение H (M) = (H1, H2), где H1 и H2 являются 64-битными величинами. Значения H1 и H2 используются вместе с 31 несекретными ключами a1, a2, ..., a31 для создания 31 уникальных кодовых слов b1, b2, ..., b31 следующим образом: Ea1 (H1) Ea1 (H2) = b1, Ea2 (H1) Ea2 (H2) = b2, ..., Ea31 (H1) Ea31 (H2) = b31. 31 значения b теперь сортируются в числовую последовательность

Тогда сигнатура представляет собой последовательность ключей k (5,1), k (20,2), k (11,3), ..., k (7,31), что гарантирует, что один и только один ключ выбран из каждого строки и столбца.

3. Подтверждение подписей

Приемник, за исключением одного дополнительного шага, проверяет подпись, повторяя те же шаги, которые выполняет отправитель. Во-первых, получается сжатое кодирование сообщения. Затем вычисляются 31 b-значения и сортируются в числовом порядке. Это позволяет повторно вставить 31 ключ в вектор подписи в пустую матрицу 3131. Затем приемник использует каждый ключ подписи для шифрования стандартного несекретного кодового слова, чтобы сформировать каждый нижний ключ в том же столбце матрицы, включая заключительные ключи, которые составляют шаблон проверки. Если заключительные ключи (строка 31) ключевой матрицы равны шаблону проверки, ранее полученному от отправителя, то сообщение и подпись принимаются как действительные; в противном случае сообщение и подпись отклоняются.

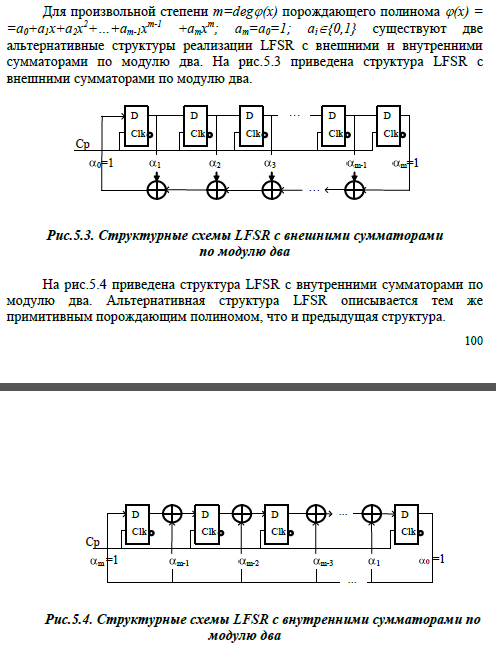
**Билет №16**

**16.1 Современная роторная машина**

Теория – 38 стр.

**16.2 Регистры сдвига с линейной обратной связью двух типов.**

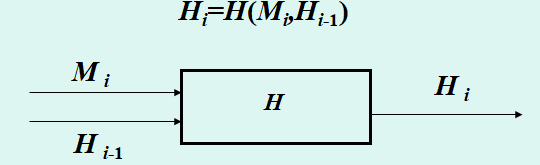
Регистр сдвига с линейной обратной связью (LFSR) часто используется в качестве криптографического ключа для кодера устройств поточного шифрования. Одной из основных причин этого является то, что LFSR легко получить и это сравнительно недорого. Кодер, который генерирует ключ-битовый поток должен быть детерминированным, так что ключ-битовый поток может быть воспроизведен для дешифровки в соответствии с примитивным многочленом: *P(x)=a0+a1x+a2x 2+…+am-1x m-1 +amx m ; am=a0=1; aiÎ{0,1}* Степень многочлена m задает длину сдвигового регистра. Ненулевые коэффициенты am определяют разряды регистра, которые будут участвовать в формировании вдвигаемого в первый разряд значения. Через bk (bkÎ {0,1}) обозначены текущие значения разрядов LFSR (k = 1, …, m). Новое значение для всех разрядов, кроме первого, берется из ближайшего младшего разряда. «XOR» используется для сложения значений из разрядов, которые отмечены ненулевыми коэффициентами ak характеристического многочлена. Полученная сумма вдвигается в первый разряд LFSR. Очередной бит ключа Ki для потоковой криптосистемы равен значению старшего разряда LFSR bm.Пример P(x)=1+x+x 3



**16.3 Однонаправленные хэш-функции. Основные свойства.**

Функция Hash H (M) имеет сообщение M, которое следует зашифровать как аргумент. Длина сообщения должна быть произвольной, а размер хэш-функции фиксируется m битами. Важно подчеркнуть, что функция H (M) является односторонней функцией, это означает, что восстановление открытого текста, основанного на значении функции, невозможно.

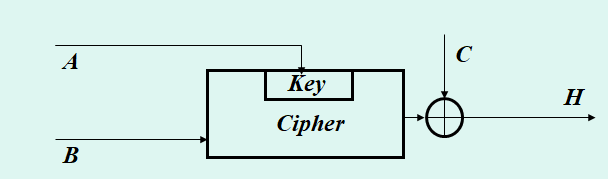
Большинство хэш-функций вычисляется на основе следующего рекуррентного соотношения



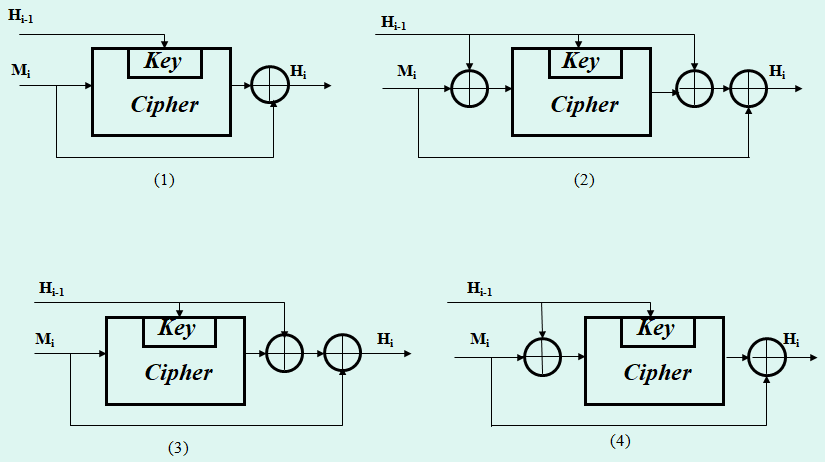
где Mi - последовательный блок сообщения.

В случае, когда длина хеш-значения Hi и блок сообщений равны, может использоваться следующая диаграмма для расчета.

В случае, когда длина хеш-значения Hi и блок сообщений равны, может использоваться следующая диаграмма для расчета.



H0-IV - начальный вектор; Hi = EA (B) C, где A, B и C могут принимать следующее значение Mi, Hi-1 и (MiHi-1) или быть постоянными значениями. где Mi - последовательный блок сообщения.

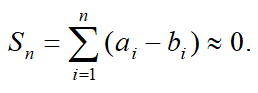


Хэш-функция предназначена для сжатия подписываемого документа М до нескольких десятков или сотен бит. Хэф- функция h() принимает в качестве аргумента сообщение М произвольн. длины и возвращает хэш-значение h(M)=H фиксиров. длины. Обычно хэшированная инфа явл. сжатым двоичным представлением основного сообщения. Значение h(M) сложным образом зависит от документа M и не позволяет восстановить сам документ. Хэш-ф-ция должна удовлетв. ряду условий: 1)должна быть чувствительна к всевозможн. изменениям в тексте М 2)должна обладать св-вом необратимости, т.е. задача подбора М должна быть вычислительно неразрешима; 3)вероятность совпадения хэш-ф-ций 2ух различных документов должна быть ничтожно мала. Одноправлен. ф-ция f() образует выходное зн.. длиной n при задании 2ух входн. знач. длиной n: блок исходного текста Mi и хэш-значение Hi-1 предыдущ. блока текста. Hi= f(Mi, Hi-1). Хэш-значение, вычисляемое при вводе последн. блока текста, станов. хэш-знач. Всего сообщения М. В рез-те однонаправл. хэш- ф-ция всегда формирует выход фикс. длины n. Примером может служить однонаправлен. хэш-ф-ции на основе симметричн. блочн. алгоритмов.

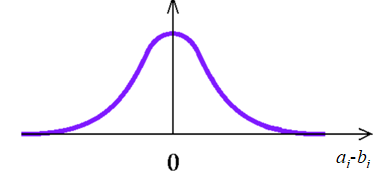
**16.4 Графическая стеганография Patchwork метод.**

Лоскутный метод (W.Bender, 1996),

Метод патч-метода основан на особенностях графических изображений, которые для достаточно большого числа n пар пикселей A и B со значениями a и b, параметр

****

имеет значение, очень близкое к нулю. Распределение S = a - b показано ниже



Метод Patchwork состоит из изменения яркости случайного выбранного набора пар пикселей (ai, bi), таким образом, что значение S'n для этого множества будет достаточно большим, сравнивается с нулем.

Локальный алгоритм внедрения водяных знаков:

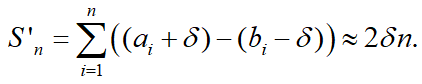
1. На основе генератора случайных чисел должна быть выбрана пара (A, B) пикселя с яркостью (ai, bi);

2. Яркость ai возрастает по значению ;

3. Яркость bi уменьшается по значению ;

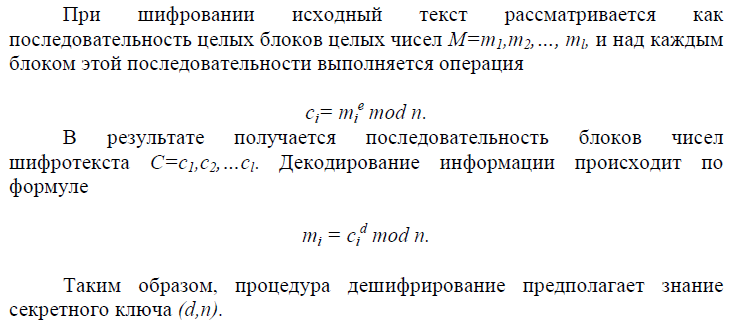
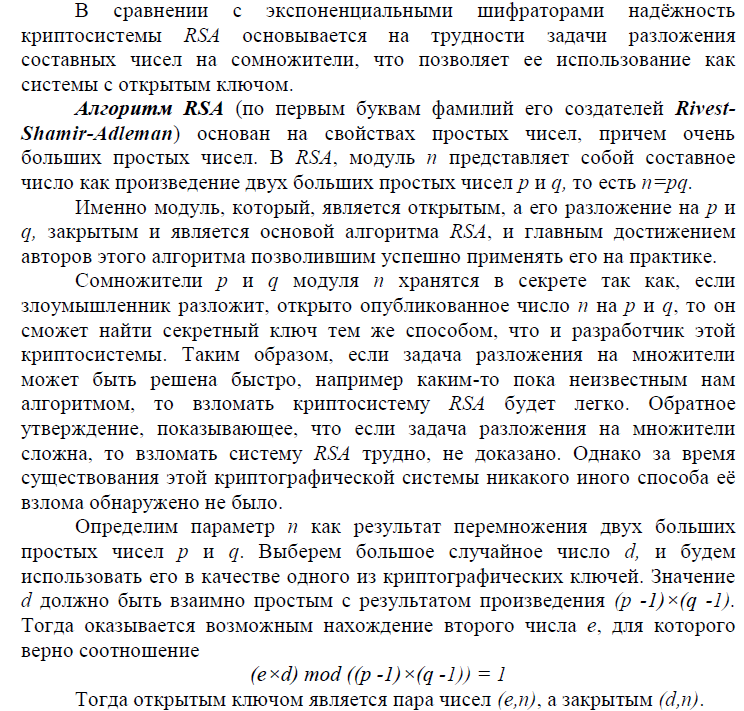
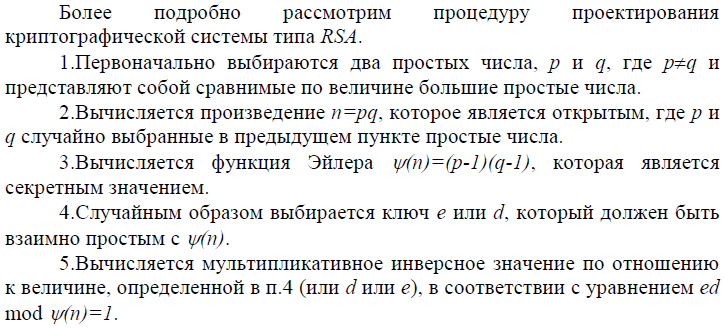
Шаги 1 - 3 повторяются n раз.

Тогда значение S'n определяется как



**Билет №17**

**17.1 Криптосистема RSA**

** **

**17.2** **Свойства М-последовательностей**

1.Сущ-ет ψ(2m -1)/m примитивных полиномов степени m, где ψ есть функция Эйлера.

2.Для зад. полинома ϕ(x) сущ-ет инверсный полином ϕ(x) -1=x mϕ(x -1 ).

3.Период М-послед-сти зависит от степени m примитивного порождающего полинома и равняется L=2m -1. Причём для заданного полинома ϕ(x) сущ-ет L=2m -1 различных М-послед-стей, каждая из кот. отличается фазовым сдвигом.

4.М-послед-сть - псевдослучайная послед-сть, в кот. вероятность появления 0 и 1( ≈ 0.5.) определяется: p(ak=1)=2m-1 /(2m -1)=1/2+1/(2m+1 -2), p(ak=0)=(2m-1 -1)/(2m -1)=1/2-1/(2m+1 -2)

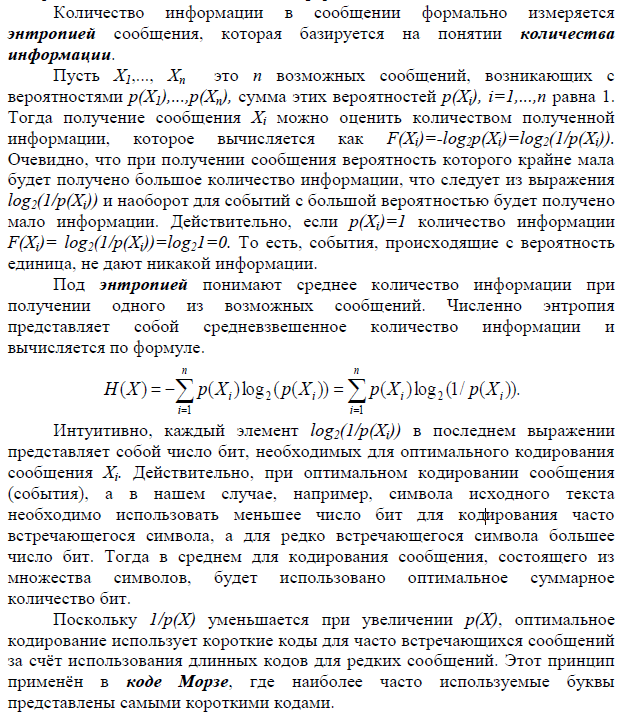
5.Ф-ция автокорреляции М-послед-сти макс. близка к автокорреляц. ф-ции идеальной случайной послед-сти. Оригинальная М-послед-сть яв-ся идентичной в 2m-1 -1 позициях со сдвинутой своей копией на любое число тактов, и будет отличаться в 2m-1 позициях.

6.Св-во сдвига и сложения.Для любого s(1≤s< L) сущ. r≠s(1≤r<L), что {ak}xor{ak-s}={ak-r}.

**7.**Среди LМ-послед-стей, получ. на осн. примитивного полинома ϕ(x) сущ-ет единств. М-послед-сть, для кот. выполн ak=a2k, k=0,1,2,…. Такая М-послед-сть -*характеристика*.

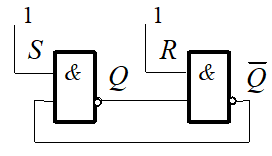
**8.**ДецимацияМ-послед-сти {ai} по индексу q, (q=1,2,3,…)означает порождение др.послед-сти {bj} как q-тые элементы первоначальной М-послед-сти {ai}, т.е. bj=qai. Если (L,q)=1, где L=2m-1, то период {bj} будет равен L=2m-1, и децимация называется нормальной.

**17.3** **Элементы математических основ криптографии. Количество информации. Энтропия.**

****

**17.4** **PUF на базе статических запоминающих устройств.**

Статическая память произвольного доступа или SRAM - очень распространенный тип энергозависимой памяти в интегральных схемах. Типичная SRAM-ячейка состоит из шести транзисторов в CMOS-технологии. Фактический элемент хранения, содержащий четыре транзистора, реализует два инвертора с перекрестной связью. Скрещиваемый элемент хранения является бистабильным, то есть имеет два стабильных состояния и, следовательно, может хранить один бит информации, находясь в одном из двух состояний. Когда ячейка включена, она быстро сходится в одно из двух устойчивых состояний. Схема с поперечной связью сконструирована таким образом, что она обеспечивает цикл обратной связи для хранения требуемого значения бита в цикле. Примером такой схемы является простая защелка R-S на основе двух вентилей 2NAND, как показано ниже.



Как было показано ранее, количество случайности, присутствующее в значениях включения питания триггеров, ограничено. Это связано с сильным преимуществом большинства триггеров, которые начинаются в «0», а не с вероятностью более или менее пятидесяти пятидесяти. Экспериментальные исследования были проведены по трем различным FPGA Virtex-II Pro. Значения включения 4096 триггеров (32 столбца × 128 строк) были измерены после 101 последовательного включения питания. Результат показывает, что более 90% измеренных триггеров всегда включаются как «0», а чуть меньше 10%, которые всегда начинаются как «1». Менее 1% будет запускаться как «0» в некоторых случаях и «1» в другое время. Эти результаты позволяют сделать вывод о том, что PUF на основе SRAM не является надежным, особенно для реализации FPGA. Для преодоления этой проблемы были предложены концепции бабочки PUF, которые ведут себя аналогично ячейке SRAM на этапе запуска.

**Битлет №18**

**18.1 Роторные машины. Структура роторной машины(+)**

**18.2 Пример потоковой криптосистемы на базе М-последовательностей.**

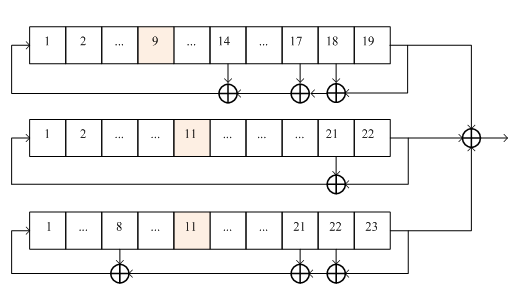
A5 / 1 - это потоковый шифр, используемый для обеспечения конфиденциальности в режиме эфирного вещания в стандарте сотового телефона GSM. Первоначально он хранился в секрете, но стал публичным знанием через утечки и обратную инженерию.

Передача GSM организована как последовательность всплесков. В типичном канале и в одном направлении один пакет отправляется каждые 4,615 миллисекунды и содержит 114 бит, доступных для информации. A5 / 1 используется для создания для каждого пакета 114-битовой последовательности ключевого потока, который XORed с 114 бит перед модуляцией. A5 / 1 инициализируется с использованием 64-битного ключа вместе с общеизвестным 22-битным номером кадра.

A5 / 1 основан на комбинации трех линейных регистров сдвига обратной связи (LFSR) с нерегулярным синхронизацией. Три сдвиговых регистра указаны следующим образом:



Регистры синхронизируются в режиме остановки / прохода с использованием правила большинства. Каждый регистр имеет связанный бит синхронизации. На каждом цикле проверяется бит синхронизации всех трех регистров и определяется бит большинства. Регистр синхронизируется, если бит синхронизации совпадает с младшим битом. Следовательно, на каждом шаге синхронизируются два или три регистра, и каждый регистр останавливается с вероятностью 3/4.



**18.3 Скорость языка**

**Скорость (частота) языка (r)** - среднее число бит информации в каждом символе сообщения. Пусть для заданного языка определено множество сообщений длиной N символов, тогда скорость языка для сообщений X длиной N определяется как

**r=H(X)/N**

Простейший способ определения частоты языка (абсолютной частоты R), основывается на предположении, что все буквы алфавита языка имеют одинаковую вероятность появления во всех возможных сообщениях так же, как и всевозможные последовательности букв алфавита в сообщениях равновероятны. Если алфавит языка содержит L букв, тогда абсолютная частота может быть получена так: **R=log2L**

**18.4 Слепая подпись**

**Слепая подпись** (Blind Signature) — разновидность электронной цифровой подписи, особенностью которой является то, что подписывающая сторона не может точно знать содержимое подписываемого документа. Понятие слепой подписи придумано Дэвидом Чаумом, им же предложена первая реализация через алгоритм RSA.

**Полностью слепая подпись**: Дана ситуация: Боб — нотариус. Алисе нужно, чтобы он подписал документ не имея никакого представления о его содержании. Боб только заверяет, что документ нотариально засвидетельствован в указанное время. Тогда они действуют по следующему алгоритму: Алиса умножает документ на rand число - маскирующий множитель - и отсылает Бобу.Боб подписывает документ и отсылает обратно. Алиса удаляет маскирующий множитель и получает свой документ с подписью. Это работает, только если функции подписи и умножения коммутативны.

**Слепая подпись:** 1)Боб готовит n документов на каждом из которых написано некоторое уникальное слово (чем больше n, тем меньше у Боба шансов смошенничать). 2) Боб маскирует каждый документ уникальным маскирующим множителем и отправляет их Алисе. 3)Алиса получает все документы и случайным образом выбирает n-1 из них. 4)Алиса просит Боба выслать маскирующие множители для выбранных документов. 5) Боб делает это. 6)Алиса вскрывает n-1 документов и убеждается что они корректны. 7)Алиса подписывает оставшийся документ и отсылает Бобу. Теперь у Боба есть подписанный Алисой документ с уникальным словом, которое Алиса не знает.

**Билет №19**

**19.1 Введение в теорию чисел. Теорема Евклида. Теорема 2. Теорема 3.**

Все множество натуральных чисел состоит их двух подмножеств: действительных чисел и целых чисел. Наиболее часто используется подмножество целых чисел. Целые бывают простыми и составными.

**Теорема 1.(Евклида)**

Существует бесконечное множество простых чисел. Доказательство: Предположим, что это множество конечно и состоит из простых чисел p1, p2, p3,…,pk, тогда получили противоречие, заключающееся в том, что число не делится ни на одно простое число p1, p2, p3,…,pk, тогда как оно делится на 1 и на самого себя, а это значит, что это число простое.

**Теорема 3.2.**

Для сколь угодно большого положительного целого числа k>1, на числовой оси существует k последовательно идущих друг за другом составных чисел. **Доказательство:** Число (k+1)!=2×3×4 × … ×(k+1) делится на любое из следующих чисел 2,3,4,…,(k+1). Тогда числа, следующие последовательно в числовом ряду целых чисел (k+1)!+2, (k+1)!+3, (k+1)!+4,…, (k+1)!+(k+1), являются составными числами вследствие того факта,что первое число делится, по крайней мере, на 2, второе на 3 и т. д.

**Теорема 3.3.**

Отношение количества простых чисел (x) находящихся в интервале от 2 до x к величине равной x/ln(x) стремиться к единице при x стремящемся к бесконечности, то есть

**19.2 Недостаток синхронных потоковых криптосистем. Процедура их взлома.**

Зачастую синхронные потоковые системы легко взламываются.В данном случае под взломом понимается получение третей стороной долгосрочного ключа.Сама структура однозначно описывается порождающим полиномом (x)= a0+a1 x+a2x2 +…+am-1xm-1; am =a0 =1; {0,1},и для его взлома необходимо получить 2m бит шифротекста и столько же бит зашифрованного текста .непосредственно взлом будет состоять в нахождении решения системы из m линейных уравнений с неизвестными, в которой будут коэффициенты порождающего полинома, а коэффициенты значениями 2m бит последовательности ключа K.

**19.3 SHA хэш-алгоритм.**

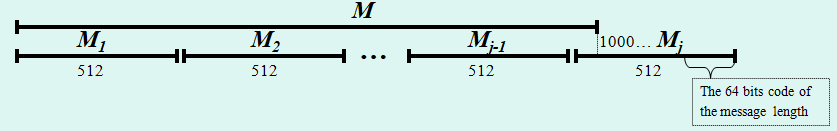
Secure Hash Algorithm (SHA) был разработан в рамках разработки Secure Hash Standard в 1992 году. Этот безопасный алгоритм хеширования является неотъемлемой частью алгоритма цифровой подписи (DSA).

Размер сообщения M, подлежащего шифрованию, должен быть меньше 2^64.

Хэш-функция H (M) называется Message Digest (MD) и имеет размер m = 160 бит.

Определение H (M) в соответствии с алгоритмом Secure Hash выполняет следующие шаги:

1. Сообщение M делится на блоки Mi со стандартным размером - 512 бит.



Их первоначальная величина принимает пять переменных.

A=0X67452301

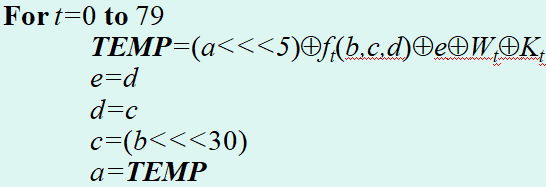
B=0XEFCDAB89

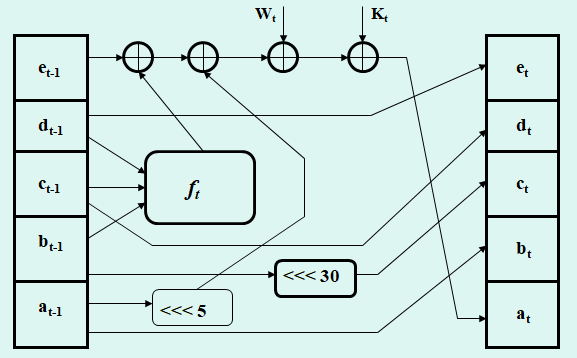
C=0X98BADCFE

D=0X10325476

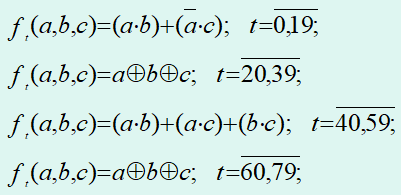
E=0XC3D2E1F0

3. Основной цикл рассчитан на выполнение 512-битового блока Mi, начиная с M1, затем M2, ... и до Mj. Для каждого блока Mi должна быть выполнена стандартная процедура. Пять 32-битных переменных A, B, C, D и E копируются во внутренние переменные a, b, c, d и e. Основной цикл можно охарактеризовать как:



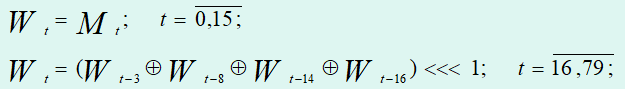


Нелинейные булевы функции ft (a, b, c):



Существует четыре постоянных значения Kt = 0X5A827999, Kt = 0X6ED9EBA1, Kt = 0X8F1BBCDC, Kt = 0XCA62C1D6 для тех же значений t, что и в случае булевых функций.

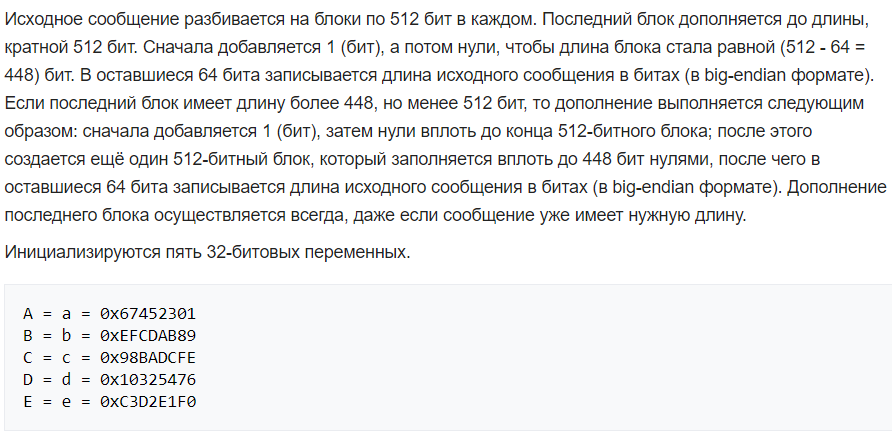
Блок Mi, состоящий из шестнадцати 32-битного слова (M0, M1, ..., M15), преобразуется в восемьдесят 32-битных слов (W0, W1, ..., W79) в соответствии с соотношениями

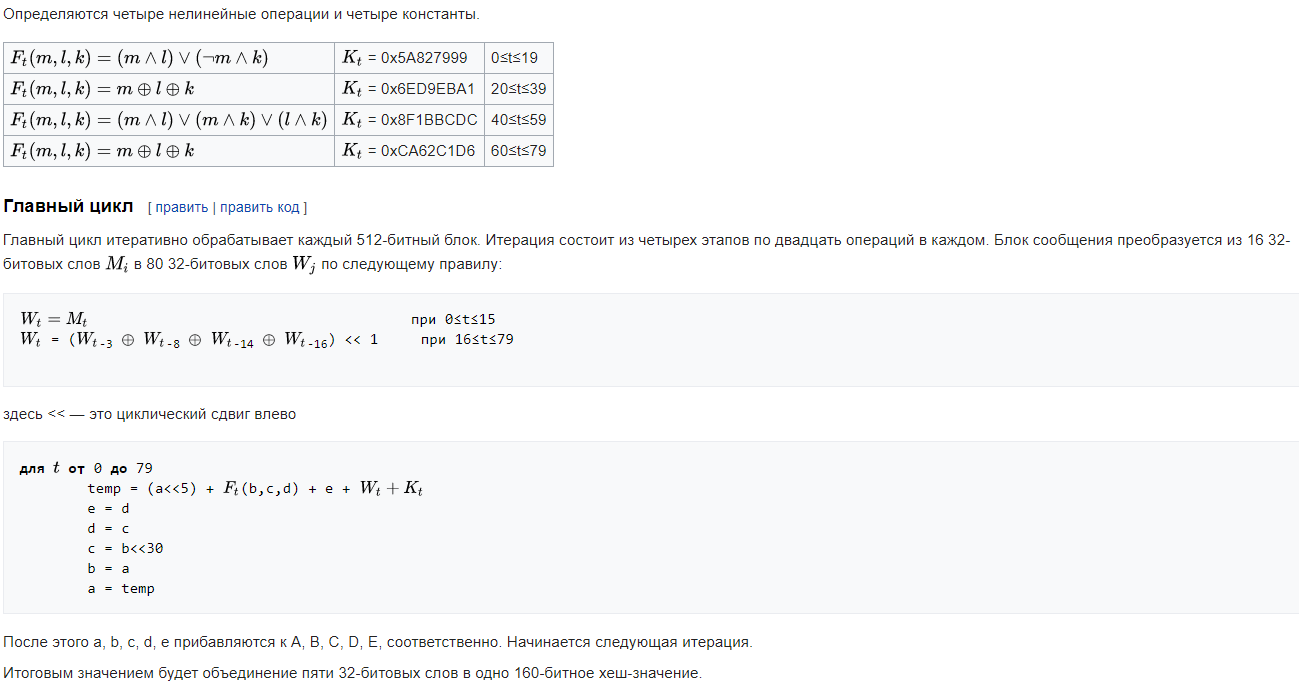


где <<< s, - циклический сдвиг влево на s-позициях.

4. В конце основного цикла значения переменных a, b, c, d и e добавляются к значениям соответствующих переменных A, B, C, D и E.

Конечные значения генерируются как конкатенация переменных A, B, C, D и E.





**19.4 Обфускация (Obfuscation). Лексическая обфускация.**

Безопасность через неясность уже давно рассматривается с презрением в сообществах безопасности и криптографии. Однако существуют ситуации, когда более высокие уровни защиты, чем достижимые через неясность в настоящее время, не достижимы. Насколько нам известно, не существует каких-либо методов предотвращения атак путем обратной инженерии, чем то, что обеспечивается затенением цели кода.

Учитывая набор запутывающих преобразований T = {T1, ..., Tn} и программу P, состоящую из объектов исходного кода (классов, методов, операторов и т. Д.) {S1, ..., Sk} find

новая программа P \* = {..., So \* = Ti (So), ...} такая, что:

P \* имеет такое же наблюдаемое поведение, как P, т. Е. Преобразования сохраняют семантику.

Безвестность P \* максимизируется, т. Е. Понимание и обратное проектирование P \* будет строго более трудоемким, чем понимание и обратное проектирование P.

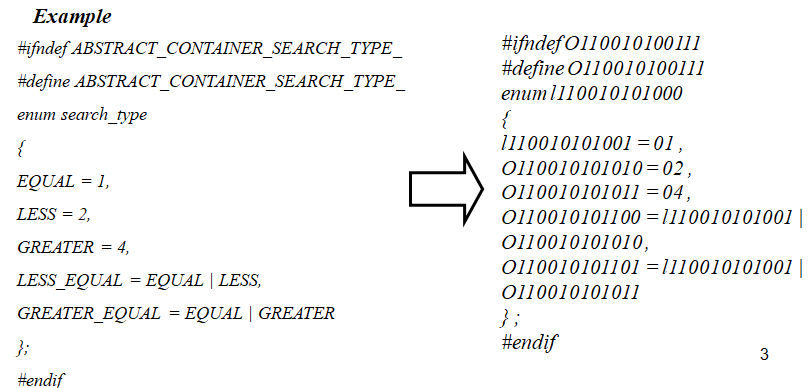
Устойчивость каждого преобразования Ti (Sj) максимизируется, т. Е. Будет сложно построить автоматический инструмент для отмены преобразований или выполнение такого инструмента будет чрезвычайно трудоемким.

Стебель каждого преобразования Ti (Sj) максимизируется, т. Е. Статистические свойства Sj \* аналогичны статистическим свойствам Sj

Стоимость (время выполнения / пробел времени) P \* минимизируется.

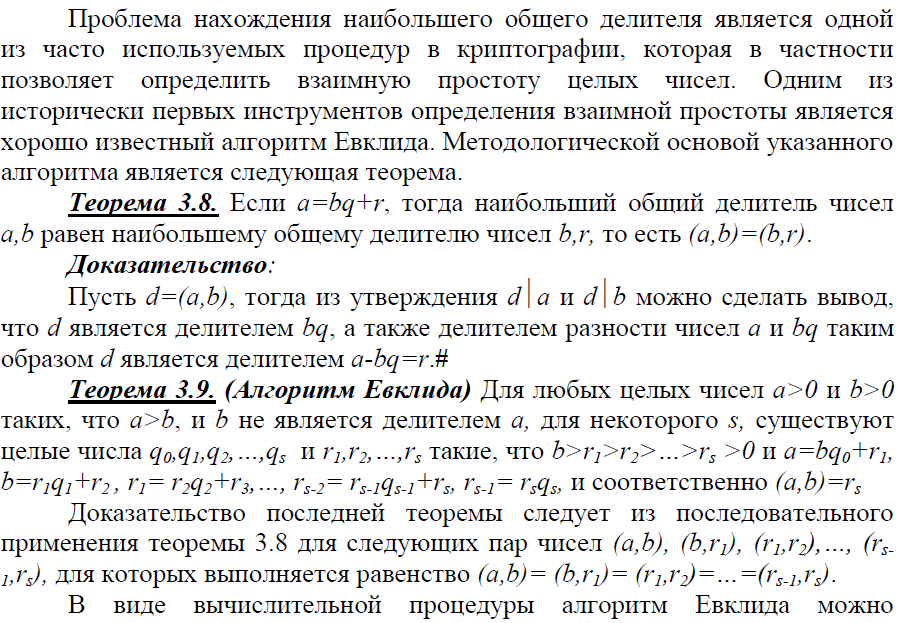
Лексические преобразования

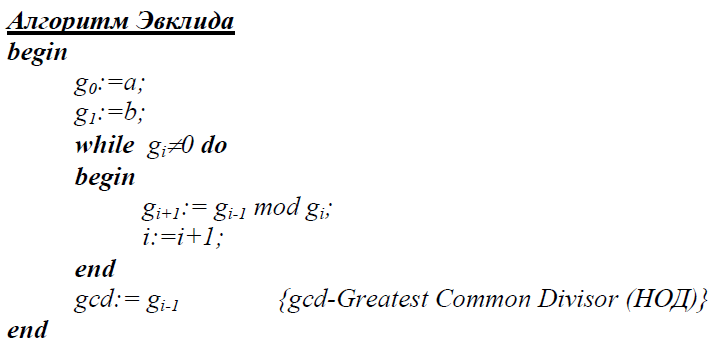
Существует ряд инструментов обфускации Java, большинство из которых модифицируют только лексическую структуру программы. Как правило, они не более чем скремблируют идентификаторы. Такие лексические преобразования, несомненно, будут раздражать инженера-реверсора и, следовательно, предотвратят некоторое проникновение интеллектуальной собственности в программное обеспечение.



**Билет №20**

**20.1 Алгоритм Евклида.**

****

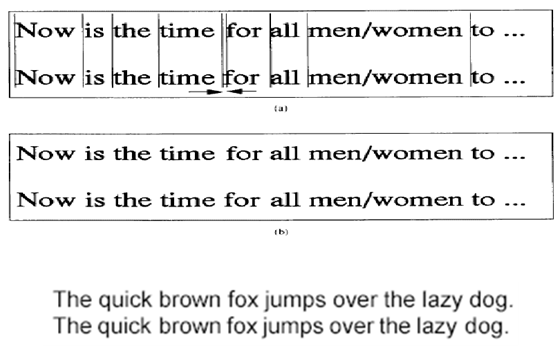
****

**20.2 Текстовая стеганография.**

Текстовое использование стеганографии можно описать по методике Brassil et al. Брассиль и другие обеспокоены незаконным распространением документов, в которые входят люди, которых вы не платите отчисления оригинальному автору, и не используйте документы, обладающие авторским правом. Brassil разработал метод маркировки печатных документов с уникальным кодовым словом и не видимым большинству. Это в основном форма водяных знаков. Кодовые слова могут быть двоичным числом, которое встроено в документ, изменяя определенные текстовые функции. Существует три конкретных метода, которые он разработал; кодирование с линейным сдвигом, кодирование с сдвигом слов и кодирование объектов.

Кодирование с линейным сдвигом - это метод, который перемещает вертикальную текстовую строку, делая документ уникальным. Например, можно взять исходный документ и переместить каждую вторую строку 1/300 дюйма вверх или вниз. Этот метод более заметен, чем другие.

Кодирование Word-shift вставляет кодовое слово в документ, перемещая горизонтальные местоположения слов в текстовых строках, сохраняя при этом естественный внешний вид. Этот метод применим только к документам с переменным расстоянием между смежными словами. Одним из способов использования этой техники является использование каждой текстовой строки и определение самых больших и малых пробелов между словами. Для кода наибольший интервал уменьшается на определенную величину, а наименьший - на ту же сумму. Это поддерживает длину линии и мало видимых изменений.



Встроенное сообщение

****

**20.3 Иерархия криптографических ключей. Методы генерирования ключей.**

Любая информация об используемых ключах должна быть защищена, в частности храниться в зашифрованном виде.

Необходимость в хранении и передаче ключей, зашифрованных с помощью других ключей, приводит к концепции *иерархии ключей*. В стандарте ISO 8532 (Banking-Key Management) подробно изложен **метод главных/сеансовых ключей**(master/session keys). Суть метода состоит в том, что вводится иерархия ключей: **главный ключ** (ГК), **ключ шифрования ключей** (КК), **ключ шифрования данных** (КД).

Иерархия ключей может быть:

· двухуровневой (КК/КД),

· трехуровневой (ГК/КК/КД).

Самым нижним уровнем являются *рабочие или сеансовыe КД*, которые используются для шифрования данных, персональных идентификационных номеров (PIN) и аутентификации сообщений. Когда эти ключи надо зашифровать с целью защиты при передаче или хранении, используют ключи следующего уровня – *ключи шифрования ключей*. Ключи шифрования ключей никогда не должны использоваться как сеансовые (рабочие) КД, и наоборот.

Такое разделение функций необходимо для обеспечения максимальной безопасности. Фактически стандарт устанавливает, что различные типы рабочих ключей (например, для шифрования данных, для аутентификации и т.д.) должны всегда шифроваться с помощью различных версий ключей шифрования ключей.

В частности, ключи шифрования ключей, используемые для пересылки ключей между двумя узлами сети, известны также как *ключи обмена между узлами сети* (cross domain keys). Обычно в канале используются два ключа для обмена между узлами сети, по одному в каждом направлении. Поэтому каждый узел сети будет иметь *ключ отправления*для обмена с узлами сети и *ключ получения* для каждого канала, поддерживаемого другим узлом сети.

На верхнем уровне иерархии ключей располагается *главный ключ, мастер-ключ*. Этот ключ применяют для шифрования КК, когда требуется сохранить их на диске. Обычно в каждом компьютере используется только один мастер-ключ.

Мастер-ключ распространяется между участниками обмена неэлектронным способом – при личном контакте, чтобы исключить его перехват и/или компрометацию. Раскрытие противником значения мастер-ключа полностью уничтожает защиту компьютера.

Значение мастер-ключа фиксируется на длительное время (до нескольких недель или месяцев). Поэтому генерация и хранение мастер-ключей являются критическими вопросами криптографической защиты. На практике мастер-ключ компьютера создается истинно случайным выбором из всех возможных значений ключей. Мастер-ключ помещают в защищенный по считыванию и записи и от механических воздействий блок криптографической системы таким образом, чтобы раскрыть значение этого ключа было невозможно. Однако все же должен существовать способ проверки, является ли значение ключа правильным.

**Главный ключ** — высший ключ в иерархии, который не защищается криптографически. Его защита осуществляется с помощью физических или электронных средств.

**Ключи для шифрования ключей** — закрытые или открытые ключи, используемые для засекречивания перед передачей или при хранении других шифровальных ключей. Эти ключи сами могут быть зашифрованы с помощью других ключей.

**Ключи для шифрования данных** — используются для защиты данных пользователей. Ключи более высоких уровней используются для защиты ключей или данных на более низких уровнях, что уменьшает ущерб при раскрытии ключей и объём необходимой информации, нуждающейся в физической защите.

Методы генерирования дублируют другие билеты по темам:

1. Генераторы криптографического ключа

2. Генераторы М - последовательностей

3. Генераторы нелинейных последовательностей

4. Комбинированные генераторы ключа

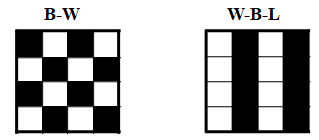
5. Синхронные потоковые шифраторы

6. Самосинхронизирующиеся потоковые криптосистемы

**20.4 ABCDE стеганография.**

ABCDE Stegangraphy - данные, основанные на блочной сложности. Встраивание можно рассматривать как расширение стеганографии BPCS, основанное на трех измерениях сложности: мера сложности черно-белой границы (то же, что и для BPCS); нерегулярность по длине; пограничный шум.

Для примера шаблона шахматной доски (B-W) мера α = 1, то есть максимум α. Однако этот блок имеет регулярный периодический характер и не может считаться сложным.



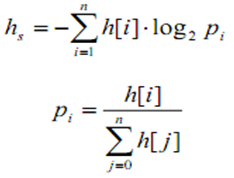
Возможным обходным решением здесь является введение верхнего предельного порога αU и принятие только блоков, которые имеют α, принадлежит интервалу αL≤α≤αU. К сожалению, даже если мы примем этот новый критерий, мы можем столкнуться с такими блоками, как W-B-L, для которых α = 0,5

Неравномерность длины пробега - определяется на основе гистограммы длины пробега как черно-белых пикселей вдоль строки, так и столбца. Предположим, что у нас есть бинарный пиксель, показанный ниже



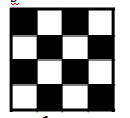
Он состоит из: пробега одного черного пикселя; пробег двух белых пикселей; пробег двух черных пикселей и пробег трех белых пикселей. Здесь мы находим, что h [1] = 1, h [2] = 2, h [3] = 1, где h [i] - частота прогонов i пикселей в черном или белом.

Теперь вводятся следующие hs для измерения неравномерности бинарной последовательности пикселей:

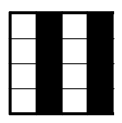
****

где n - самая длинная длина пути

Для области W-B у нас есть восемь из одного белого и восьми черных прогонов, поэтому h [1] = 16, и нет никаких прогонов с большим количеством пикселей. Вот почему p1 = 16/16 = 1, а hs = 0.



Для области W-B-L мы имеем два из четырех белых и двух из четырех черных прогонов, поэтому h [4] = 4, и нет никаких прогонов с большим количеством пикселей. Вот почему p4 = 4/4 = 1, а hs = 0.



Если последовательность содержит пробеги только одной длины, она будет показывать регулярную периодичность. С другой стороны, если последовательность содержит пробеги различной длины, она не будет иметь никакой отдельной периодичности, а hs станет большой.

Измерение сложности шумности границы вычисляется на основе различий между соседними бинарными последовательностями пикселей в блоке. Пусть размер блока равен n × n (n> 1). Пусть ri и cj - i-я строка и j-й столбец блока соответственно. Уровень шума  блока определяется следующим образом:



Где Px (r) = {g (r0r1), ..., g (rn-2rn-1)}; Px (c) = {g (c0c1), ..., g (cn-2cn-1)}, символ  обозначает бит-мудрый эксклюзив - или, g (x) - число единиц в двоичном последовательность x и

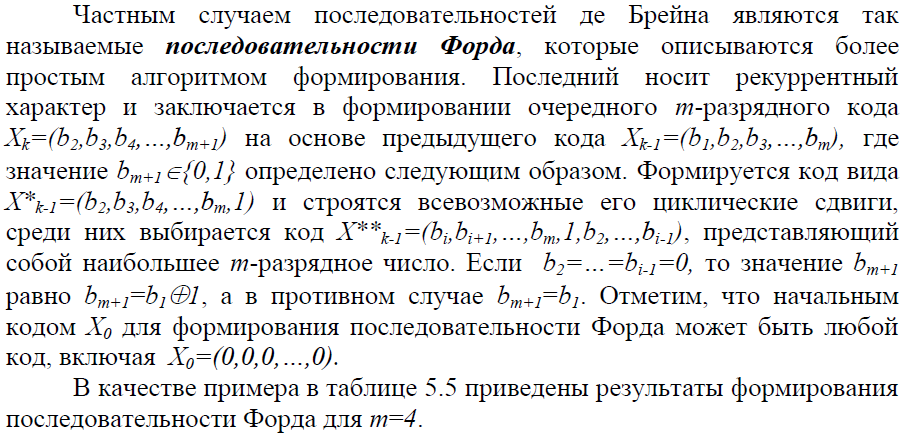


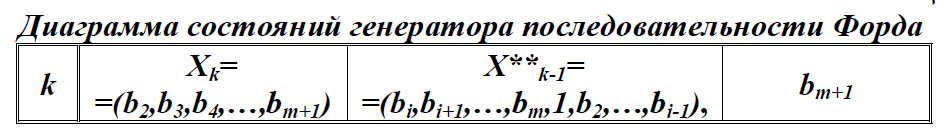
Х = {х0, ... хт-1}; Var (X) - дисперсия X; и Aver (X) - среднее значение X.

Определение шумности границы  аналогично определению нерегулярности длины пробега . Количество черно-белых границ пикселей подсчитывается для каждой пары соседних строк и столбцов. Они представлены множеством Px (r) для строк и Px (c) для столбцов в приведенном выше определении. Затем вычисляется взвешенное среднее значение Ef против этих множеств.

**Билет №21**

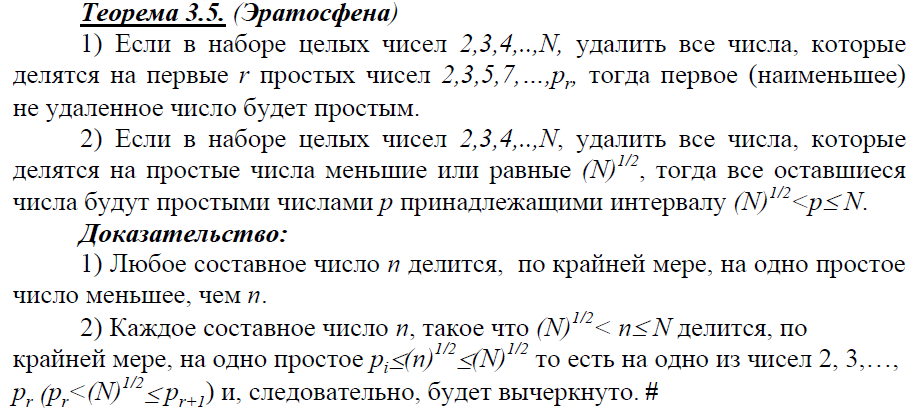
**21.1 Последовательность Форда**

****

****

****

**21.2 Генерирование простых чисел. Решето Эратосфена.**

****

**21.3 Отличия между потоковыми и симметричными криптосистемами.**

**Шифрования** – алгоритмы шифрования, для которых для зашифров. И расшифровыв. Используется один и тот же ключ, или ключ расщифровывания легко вычисляется из ключа зашифровыв. И наоборот. Симметричное шифрование бывает двух видов : блочное и потоковое.

**Блочное** - в этом случае информация разбивается на блоки фиксированной длины(например 64 или 128 битов), после чего эти блоки поочерёдно шифруются. Причём существ. различные вариации одного и того же алгоритма, т.е шифрование «со сцеплениеи или без него»(независим. шифрование блоков или зашифрованный блок зависит от предыдущ. зашифрованного блока).

**Поточное** - в тех случах, когда инфу нельзя разбить на блоки, скажем некий поток данных, каждый символ которых должен быть зашифрован и отправле куда-либо не дожидаясь остальных данных, достаточных для формирования блока. Некоторые классификации считают, что потоковое шифрование- шифров. Блоков единичной длины. Основная идея потоковых криптосистем заключается в шифровании исходного текста M с помощью криптографического ключа K, длина которого равна длине текста. Каждый бит шифротекста Ci является функцией соответствующих битов исходного текста и ключевого потока. (Пример : LFSR, генератор Геффе.)

**21.4 Вычисление мультипликативной инверсной величины**

EUCLIDEX(a; b)

d0:=a; d1:=b;

x0:=1; x1:=0;

y0:=0; y1:=1;

while d1>1 do

begin

q:=d0 div d1;

d2:=d0 mod d1;

x2:=x0–q\*x1;

y2:=y0–q\*y1;

d0:=d1; d1:=d2;

x0:=x1; x1:=x2;

y0:=y1; y1:=y2;

end

return (x1; y1; d1)

Расширенный алгоритм Евклида позволяет вычислить числа *x*1 и *y*1, для которых выполняется равенство *x*1\**a*+*y*1\**b* = *d*1, где *d*1 = НОД(*a*, *b*). Если *a* и *b* взаимно простые, и *a*>*b*, то *y*1 является мультипликативным инверсным по модулю *а* для *b*, т. е. *y*1\**b* mod *a* = 1. Используя данный алгоритм, можно вычислим *d*, положив *a*, равным *φ*(*r*), и *b*, равным *e* = 7. Если значение *y*1 получилось отрицательным, то для получения корректного значения *d* необходимо добавить к *y*1 значение *a* (или *φ*(*r*)).

**Билет № 22**

**22.1 Генератор последовательностей de Bruin.(+)**

**22.2** **Стеганография (Общая информация**).

Стеганография - это искусство и наука о сокрытии сообщений внутри других сообщений. На греческом языке стеганография означает скрытое или секретное письмо. (Стегано = покрытая графа = запись).

Криптография - далекий кузен стеганографии. Криптография скремблирует сообщение, поэтому его нельзя понять, но есть знание о существовании сообщения. Хотя целью стеганографии является скрытие существования скрытого сообщения.

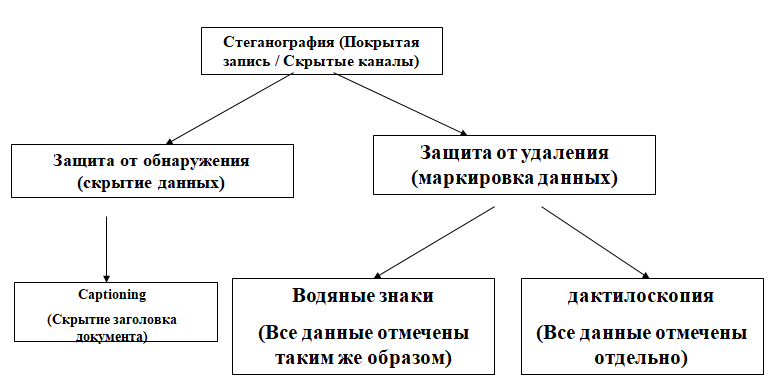
Цифровые водяные знаки и отпечатки пальцев аудио- и видеоиндустрии начинают распознавать возможные виды использования стеганографии.

Данные обложки (обложка, сосуд или контейнер) - это термин, используемый для описания оригинального, невинного сообщения, данных, звука, неподвижного изображения и т. Д. Встроенные данные - это информация, скрытая в данных обложки. Данные stego - это данные, содержащие как сигнал обложки, так и встроенную информацию.

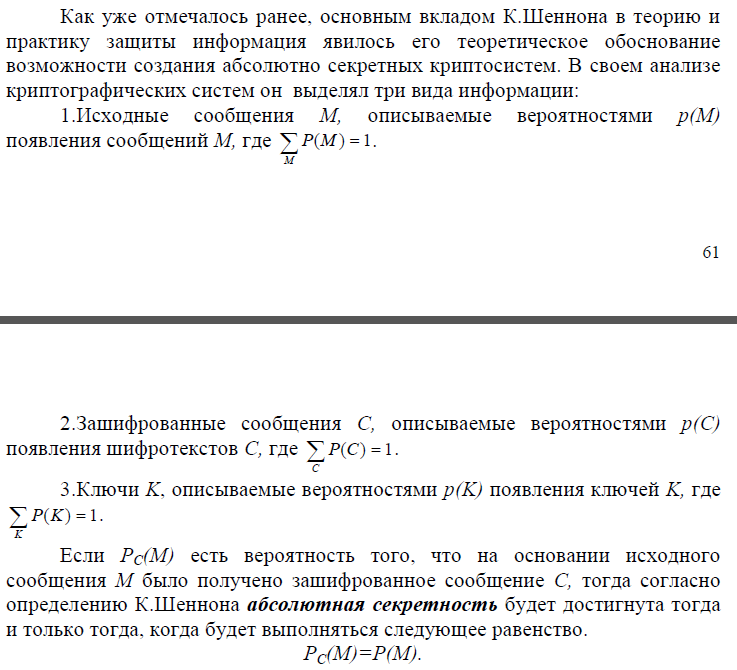
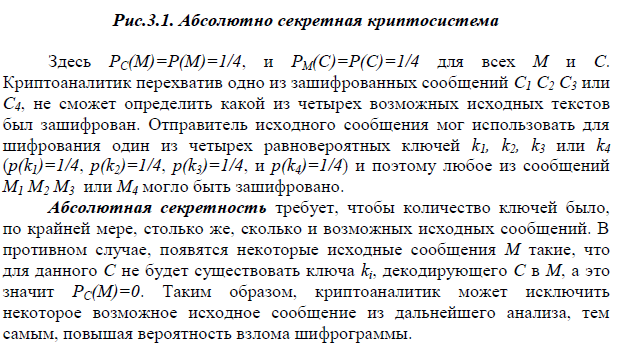
Стеганалист - это тот, кто применяет стеганализ в попытке обнаружить существование скрытой информации. Стеганалист - это тип кибер-криминалистика. Несколько других терминов, которые следует учитывать, - это способность и надежность.

Емкость относится к количеству информации, которая может быть скрыта перед искажением, или количеству информации, которая может быть введена в аудио с помощью подслушивающих устройств для обнаружения.

Надежность - это величина модификации, которую может поддерживать защитная среда до того, как скрытая информация будет уничтожена.



**22.3** **Абсолютная секретность.**

**22.4** **Управление ключами. Деление ключей.**

**Управление ключами** состоит из процедур, обеспечивающих: включение пользователей в систему;выработку, распределение и введение в аппаратуру ключей; контроль использования ключей; смену и уничтожение ключей; архивирование, хранение и восстановление ключей. Управление ключами -основа для обеспечения конфиденциальности обмена информацией, идентификации и целостности данных. Важное свойство- сведение сложных проблем обеспечения безопасности многочисленных ключей к проблеме обеспечения безопасности нескольких ключей.. В случае использования ключей для обеспечения безопасности хранимой информации субъектом может быть единственный пользователь, который осуществляет работу с данными в последовательные промежутки времени. Управление ключами в сетях связи включает, по крайней мере, двух субъектов — отправителя и получателя сообщения.

Разделение ключей по уровням:

-Главный ключ-высший ключ в иерархии,не защищается криптографически. Его защита осуществл. с помощью физич. или электронных средств.

-Ключи для шифрования ключей-закрытые или открытые ключи, использ. для засекречивания перед передачей или при хранении шифровальных ключей.Эти ключи могут быть зашифрованы с помощью других ключей.

-Ключи для шифр данных-используются для защиты данных пользователей

**Билет №23**

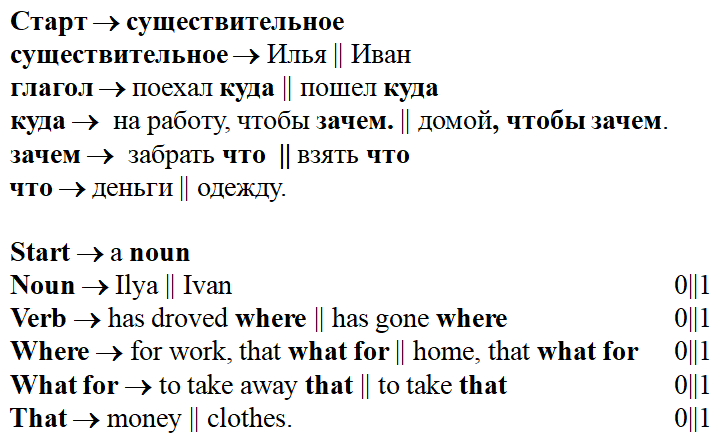
**23.1 Стеганография, основанная на формальных грамматиках (Mimicry).**

Другая широко распространенная лексическая стеганография в мимике. В соответствии с этим подходом создается реальный текст, который описан в «Свободной грамматике контекста» (CFG).

Грамматика CFG - является одним из формальных методов описания любого языка и состоит из слов фраз и правил.

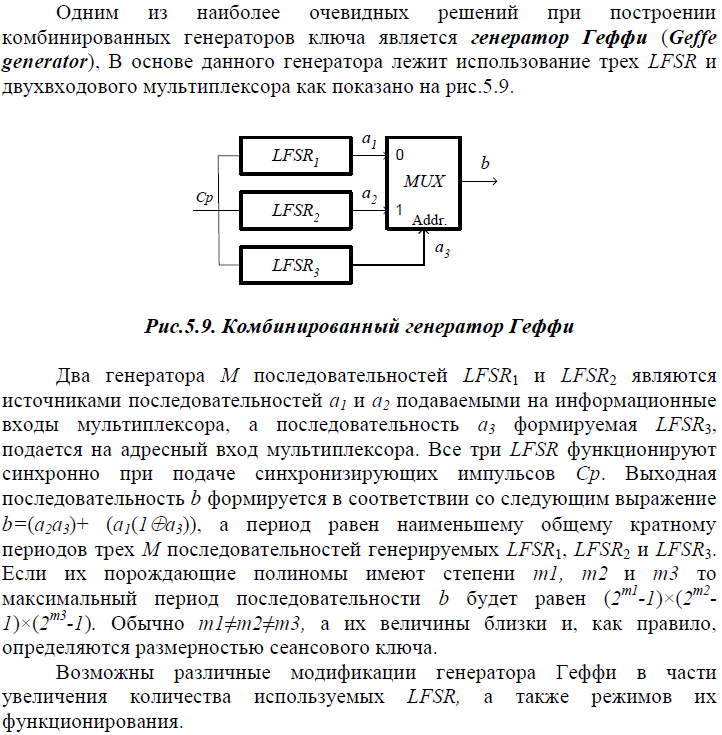
Мимикрия проектирует двоичное дерево на основе конкретного CFG с узлами решения и строит текст в соответствии со встроенными битами сообщения.

Пример: встроенное сообщение - 10101, а двоичное дерево - как показано ниже.

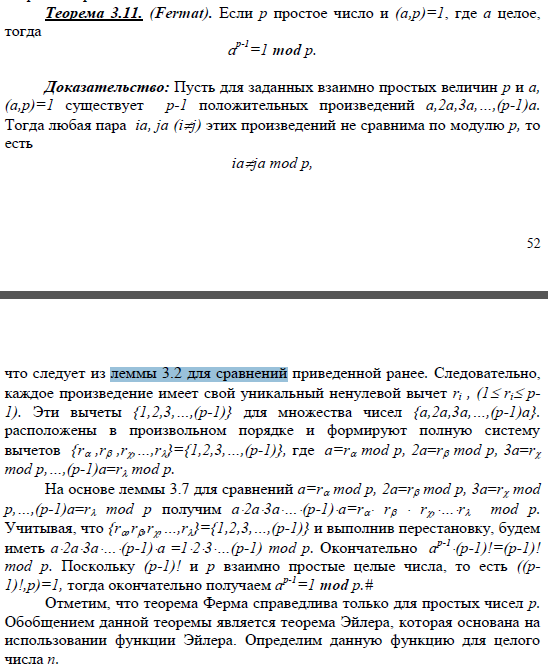
****

****

**23.2 Комбинированные генераторы ключа. Генератор Geffe**

****

**23.3 Теорема Ферма.**

****

**23.4 DES: функция f, реализующая подстановку(+)**