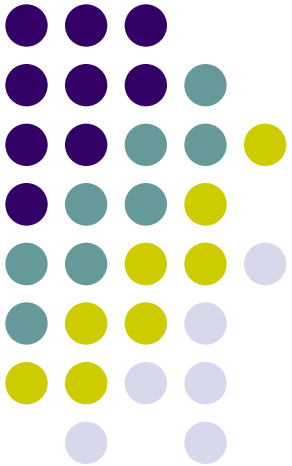
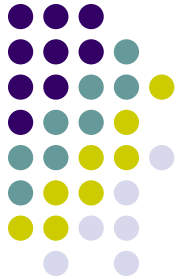
Криптографическая защита бизнес-информации

Губенко Инна Михайловна,
доц., к.ф.-м.н.
img0504@yandex.ru

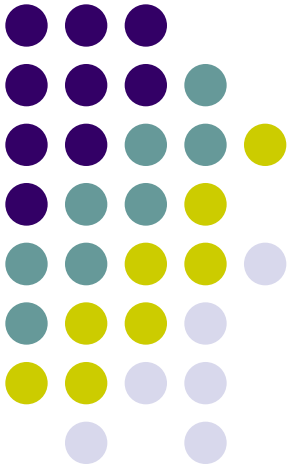


“Криптография...” Как много в этом звуке…..

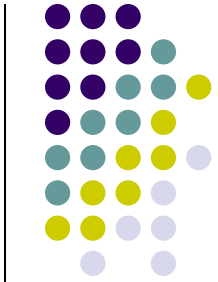


- Основные понятия
- Математические основы
- Симметричное шифрование и криптография с открытым ключом
- Цифровая подпись
- Хэш-функции
- Генераторы псевдослучайных последовательностей
- Вопросы управления ключами
- И о Космосе

I. ОСНОВНЫЕ ПОНЯТИЯ



Сообщения и шифрование



- Само по себе сообщение называется **открытым текстом (clear,или клер)**
- Изменение вида сообщения, чтобы скрыть его суть называется **шифрованием (enchipher [ISO 7498-2], encrypt)**
- Процесс преобразования шифртекста в открытый – **дешифрование (dechipher [ISO 7498-2], decrypt)**
- Искусство и наука безопасных сообщений – **криптография** – воплощается **криптографами.**
- Искусство и наука взламывания шифра – **криптоаналитиками,** использующими **криптоанализ**
- Отрасль математики, включающая **криптографию** и **криптоанализ**, назевается **криптологией**, а ее специалисты – **криптологи**.

Криптосистема: алгоритм, открытый текст, шифртекст, ключи

Симметричные криптоалгоритмы

Шифрование и дешифрирование с ключом

$$E_K(M)=C$$

$$D_K(C)=M$$

$$D_K(E_K(M))=M$$

2 категории - потоковые и блочные шифры

Криптосистема: алгоритм, открытый текст, шифртекст, ключи

Асимметричные криптоалгоритмы (или криптоалгоритмы с открытым ключом)

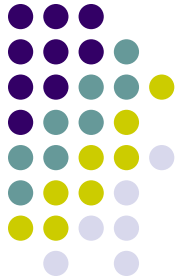
Шифрование и дешифрирование с двумя различными ключами

Ключ шифрования – **открытый** ключ,
для дешифрования - **закрытый**

$$E_K(M)=C$$

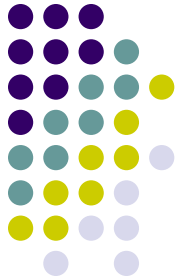
$$D_K(C)=M$$

Для чего шифровать???



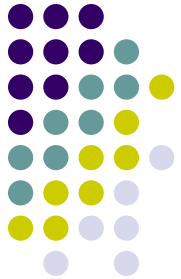
1. Обеспечение конфиденциальности
2. Проверка подлинности (получатель может проверить источник, злоумышленник нет)
3. Проверка целостности (получатель может проверить, было ли изменено сообщение, злоумышленник не знает, было ли сообщение ложным)
4. Неотрицание авторства.

Криптоанализ



- **Смысл** криптографии – **сохранение** открытого текста, ключа (либо и того, и другого) от злоумышленника
- Предполагается, что **злоумышленник** полностью **контролирует линии связи** между отправителем и получателем
- Раскрытие ключа некриптологическими (а криптоаналитическими) способами называется **компрометацией**
- Попытка криптоанализа называется **вскрытием**
- **Основное предположение криптоанализа (А. Керкхофс, XIX в) : безопасность полностью определяется *ключом*,т.е. криптоаналитик полностью располагает криптоалгоритмом и его реализацией.**

Некоторые способы криптоаналитического вскрытия



1. **Вскрытие с использованием только шифротекста**

   Дано: $C_1=E_k(P_1), C_2=E_k(P_2), \ldots C_i=E_k(P_i)$

   Получить: Либо $P_1, P_2, \ldots P_i$; k; либо алгоритм, как получать P_{i+1} из $C_{i+1}=E_k(P_{i+1})$

2. **Вскрытие с использованием открытого текста**

   Дано: $P_1, C_1=E_k(P_1), P_2, C_2=E_k(P_2), \ldots P_i, C_i=E_k(P_i)$

   Получить: Либо k; либо алгоритм, как получать P_{i+1} из $C_{i+1}=E_k(P_{i+1})$

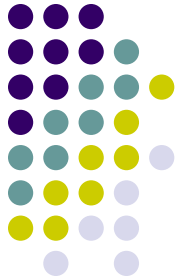
3. **Вскрытие с использованием выбранного открытого текста**

   Дано: $P_1, C_1=E_k(P_1), P_2, C_2=E_k(P_2), \ldots P_i, C_i=E_k(P_i)$
   где криптоаналитик может выбирать $P_1, P_2, \ldots P_i$

   Получить: Либо k; либо алгоритм, как получать P_{i+1} из $C_{i+1}=E_k(P_{i+1})$

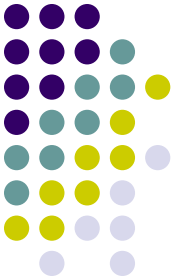
4. **Адаптивное вскрытие с использованием открытого текста**

Данные “скорее всего” в безопасности, если ……



1. Стоимость взлома алгоритма больше, чем зашифрованные данных
2. Время взлома алгоритма больше, чем время, в течение которого зашифрованные данные должны храниться в секрете
3. Если объем данных, зашифрованных одним ключом, меньше, чем объем данных, необходимый для взлома алгоритма

Категории вскрытия алгоритмов по Ларсу Кнудсену:



1. **Полное вскрытие**. Криптоаналитик получил ключ, K, такой, что $D_K(C) = P$.
2. **Глобальная дедукция**. Криптоаналитик получил альтернативный алгоритм, A, эквивалентный $D_K(C)$ без знания K.
3. **Местная (или локальная) дедукция**. Криптоаналитик получил открытый текст для перехваченного шифротекста.
4. **Информационная дедукция**. Криптоаналитик получил некоторую информацию о ключе или открытом тексте. Такой информацией могут быть несколько бит ключа, сведения о форме открытого текста и так далее.

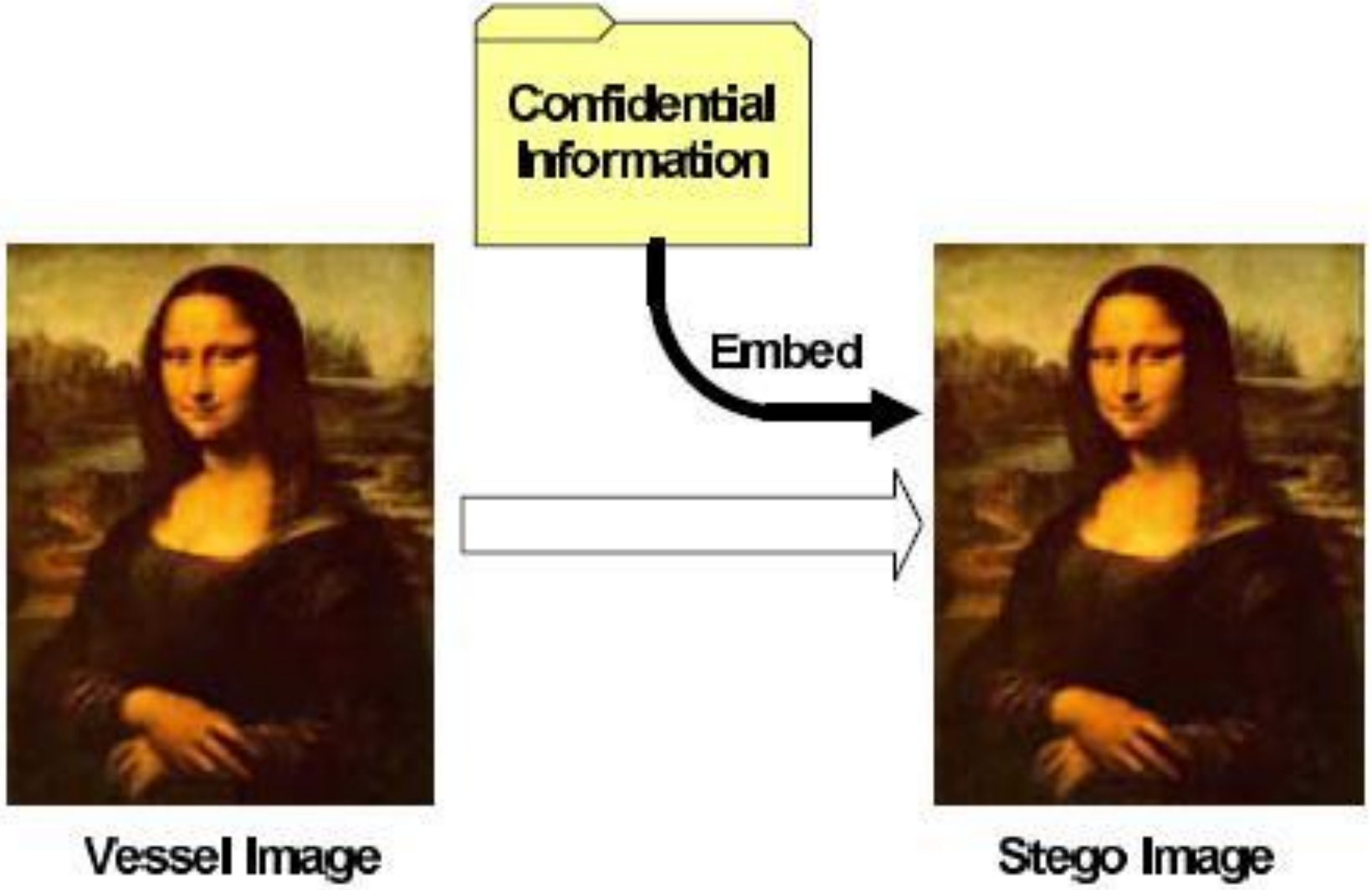
Вычислительно безопасные криптоалгоритмы: критерии

1. **Сложность данных**. Объем данных, используемых на входе операции вскрытия .
2. **Сложность обработки**. Время, нужное для проведения вскрытия . Часто называется **коэффициентом работы**.
3. **Требования к памяти**. Объем памяти, необходимый для вскрытия .

Оценки среднего времени для аппаратного вскрытия грубой силой

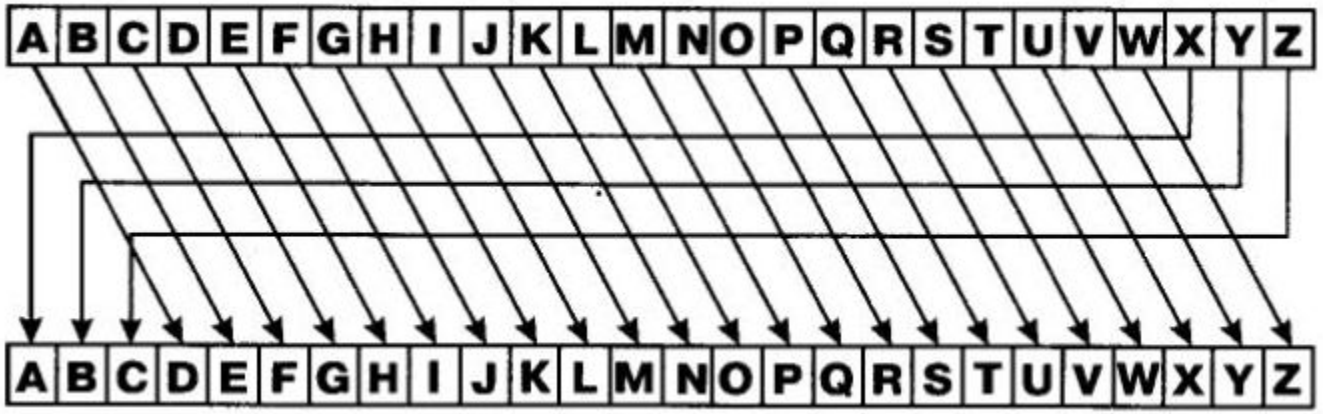
Длина ключей в битах					
40	56	64	80	112	128
0.02 микросекунды	1 миллисекунда	0.3 секунды	6 часов	10^6 лет	10^{11} лет

Стеганография



Из истории…..

- **Шифр Цезаря** (симметричный, подстановка)



Ключ: 3

Открытый текст:

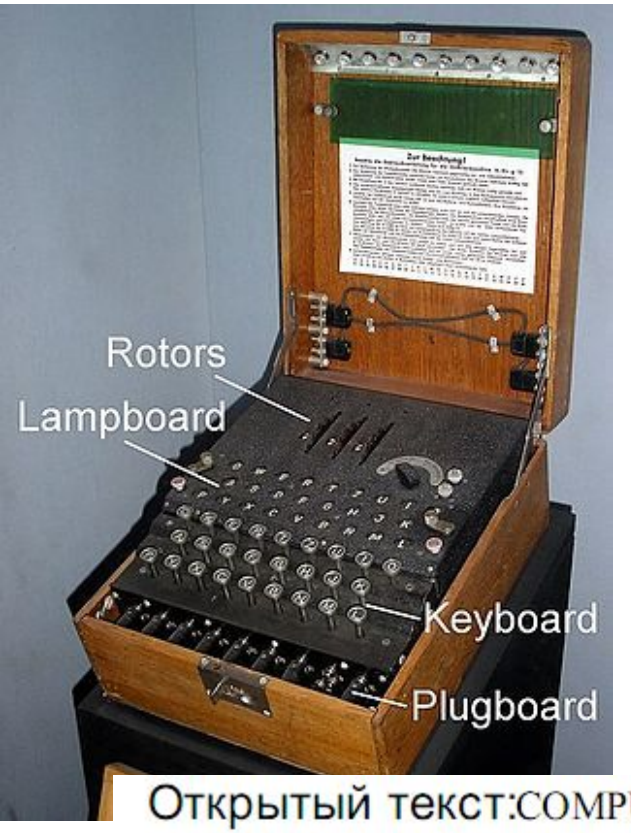
P = HELLO CAESAR CIPHER

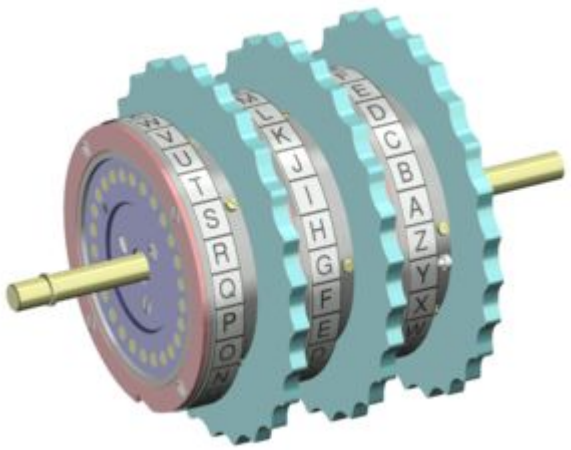
Зашифрованный текст:

C = KHOOR FDHVDU FLSKHU

Из истории…..

- **Роторные машины (1920-е гг.)**



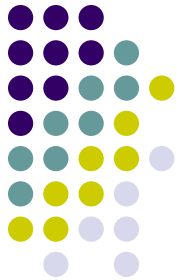


Открытый текст:COMPUTER GRAPHICS MAY BE SLOW BUT AT LEAST IT'S EXPENSIVE

COMPUTERGR
APHICSMAYB
ESLOWBUTAT
LEASTITSEX
PENSIVE

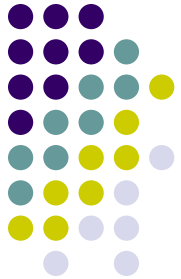
Шифротекст:CAELP OPSEE MHLAN PIOSS UCWTI TCBIV EMUTE RATSG YAERB TX

-Идеальный способ шифрования?
- Одноразовый блокнот!
-O_o



- Это большая **неповторяющаяся последовательность** символов **ключа**, распределенных случайно, написанных на бумаге и приклеенных к листу блокнота.
- *Отправитель* использует **каждый символ ключа** блокнота для **шифрования одного символа** открытого текста, затем уничтожает страницу блокнота (часть ленты)
- *Получатель* использует такой же блокнот, дешифрируя каждый символ, затем уничтожает страницу блокнота (часть ленты).
- **Новое сообщение – новые символы ключа - новые страницы блокнота**
- **ПРЕДПОЛОЖЕНИЕ:** злоумышленник не имеет доступа к блокноту

Одноразовый блокнот: пример



1. Исходное сообщение ONETIMEPAD
2. Ключ TBFRGFARFM
3. Шифрование IPKLPSFHGQ

МЕТОД: СЛОЖЕНИЕ ПО МОДУЛЮ 26 СИМВОЛА ОТКРЫТОГО ТЕКСТА И КЛЮЧА

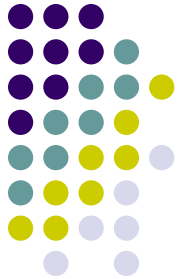
$$Q + T \bmod 26 = I$$

$$N + B \bmod 26 = P$$

$$E + F \bmod 26 = K$$

и т.д.

Компьютерные алгоритмы, используемые чаще всего



1. **Data Encryption System (DES)** – симметричный алгоритм шифрования. Американский и международный стандарт
2. **Rivest-Shamir-Adleman (RSA)** – алгоритм шифрования с открытым ключом. Применяется для шифрования и цифровой подписи
3. **Digital Signature Algorithm (DSA)** - алгоритм шифрования с открытым ключом. Применяется только для цифровой подписи

А без этого никуда…….

Операция XOR (оно же ^ , “исключающее или”)

$$0 \oplus 0 = 0$$

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

$$1 \oplus 1 = 0$$

$$a \oplus a = 0$$

$$a \oplus b \oplus b = a$$

В операциях шифрования:

$$P \oplus K = C$$

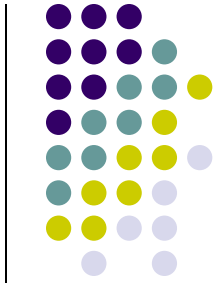
$$C \oplus K = P$$

О больших числах

Табл. 1-1. Большие числа

Физический аналог	Число
Вероятность быть убитым молнией (в течение дня)	1 из 9 миллиардов (2^{33})
Вероятность выиграть главный приз в государственной лотерее США	1 из 4000000 (2^{22})
Вероятность выиграть главный приз в государственной лотерее США и быть убитым молнией в течение того же дня	1 из2^{61}
Вероятность утонуть (в США в течение года)	1 из 59000 (2^{16})
Вероятность погибнуть в автокатастрофе (в США в году)	1 из 6100 (2^{13})
Вероятность погибнуть в автокатастрофе (в США в течение времени жизни)	1 из 88 (2^{7})
Время до следующего оледенения	14000 (2^{14}) лет
Время до превращения Солнца в сверхновую звезду	10^{9} (2^{30}) лет
Возраст планеты	10^{9} (2^{30}) лет
Возраст Вселенной	10^{10} (2^{34}) лет
Число атомов планеты	10^{51} (2^{170})
Число атомов Солнца	10^{57} (2^{190})
Число атомов галактики	10^{67} (2^{223})
Число атомов Вселенной	10^{77} (2^{265})
Объем Вселенной	10^{84} (2^{280}) см ^{3}

О больших числах : немного о Вселенной



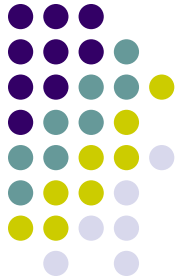
Если Вселенная конечна:

Полное время жизни вселенной	10^{11} (2^{37}) лет
	10^{18} (2^{61}) секунд

Если Вселенная бесконечна:

Время до остывания легких звезд	10^{14} (2^{47}) лет
Время до отрыва планет от звезд	10^{15} (2^{50}) лет
Время до отрыва звезд от галактик	10^{19} (2^{64}) лет
Время до разрушения орбит гравитационной радиацией	10^{20} (2^{67}) лет
Время до разрушения черных дыр процессами Хокинга	10^{64} (2^{213}) лет
Время до превращения материи в жидкость при нулевой температуре	10^{65} (2^{216}) лет
Время до превращения материи в твердое тело	$10^{10^{26}}$ лет
Время до превращения материи в черную дыру	$10^{10^{76}}$ лет

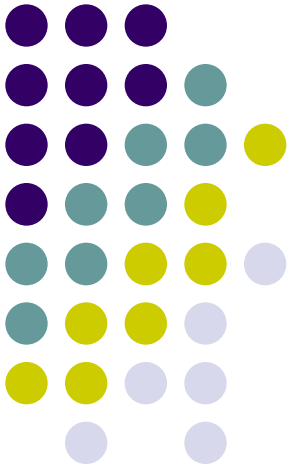
Основные направления криптологии



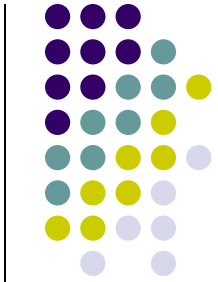
- Шифрование
- Управление ключами
- Цифровая подпись
- Криптографический протоколы
- Аппаратная реализация средств криптографической защиты информации (аппаратные СКЗИ)
- Стеганография
- Квантовая криптография
- ***+ много чего еще……***

II. Математические основы

О СВЯЗИ ТЕОРИИ СЛОЖНОСТИ И ТЕОРИИ ЭЛЕМЕНТАРНЫХ ЧИСЕЛ



Понятие вычислительной сложности



Сложность алгоритмов

Сложность алгоритма определяется вычислительными мощностями, необходимыми для его выполнения. Вычислительная сложность алгоритма часто измеряется двумя параметрами: T (**временная сложность**) и S (**пространственная сложность**, или требования к памяти). И T, и S обычно представляются в виде функций от n, где n - это размер входных данных.

Время выполнения для различных классов алгоритмов

Класс	Сложность	Количество операций для $n=10^6$	Время при 10^6 операций в секунду
Постоянные	$O(1)$	1	1 мкс
Линейные	$O(n)$	10^6	1 с
Квадратичные	$O(n^2)$	10^{12}	11.6 дня
Кубические	$O(n^3)$	10^{18}	32000 лет
Экспоненциальные	$O(2^n)$	10^{301030}	В 10^{301006} раз больше, чем время существования вселенной

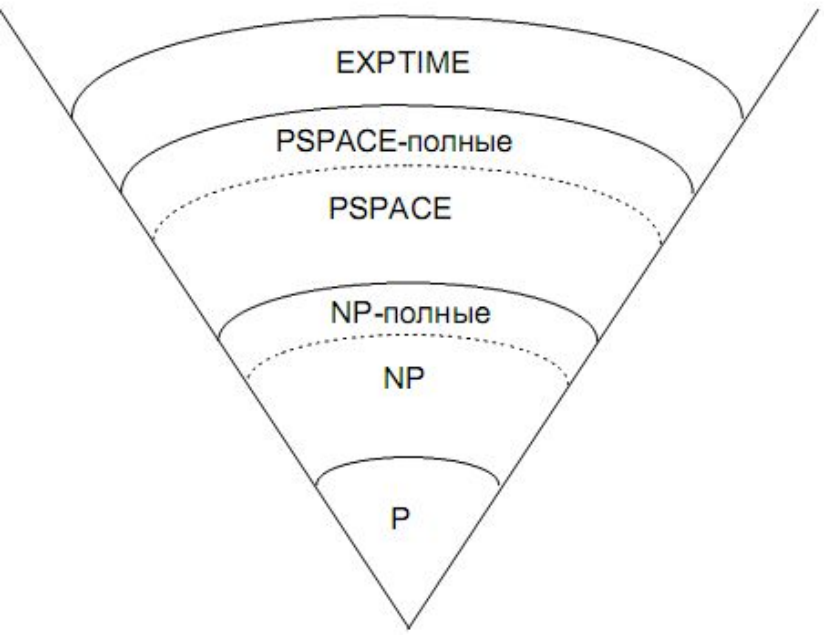
Понятие класса сложности проблемы

Сложность проблем

Теория сложности также классифицирует и сложность самих проблем, а не только сложность конкретных алгоритмов решения проблемы.

Теория рассматривает минимальное время и объем памяти, необходимые для решения самого трудн го варианта проблемы на теоретическом компьютере, известном как **машина Тьюринга**.

Машина Тьюринга представляет собой конечный автомат с бесконечной лентой памяти для чтения-записи и является реалистичной моделью вычислений.

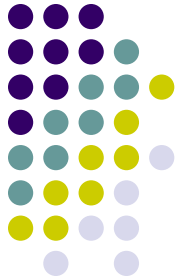


Классы сложности

Примером задач, принадлежащих классу P, являются существование **односторонних функций** - математическая функция, которая легко вычисляется для любого входного значения, но трудно найти аргумент по заданному значению функции. Речь идет об:

- **Умножение и факторизация**
- **Возведение в квадрат и извлечение кв. корня по модулю**
- **Дискретное экспоненцирование и логарифмирование**
- **Криптографические хэш-функции**

Пример



Дискретные логарифмы

В качестве другой однонаправленной функции в криптографии часто используется возведение в степень по модулю. Легко вычислить:

$a^x \bmod n$

Задачей, обратной возведению в степень по модулю, является поиск дискретного логарифма.

Найти x, для которого $a^x \equiv b \pmod{n}$.

Например:

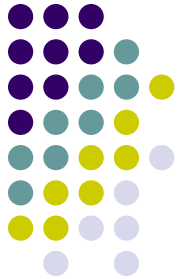
Если $3^x \equiv 15 \bmod 17$, то $x = 6$

Решения существуют не для всех дискретных логарифмов (помните, речь идет только о целочисленных р е-шениях). Легко заметить, что следующее уравнение не имеет решений

$3^x \equiv 7 \pmod{13}$

Еще сложнее решать эту задачу для 1024-битовых чисел.

Основные сведения о целых числах (Из теории чисел)



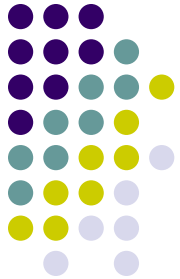
1. Неполное частное и остаток от деления

Для любой пары целых чисел x, y, где $y \neq 0$, существует единственная пара целых чисел (q, r), удовлетворяющая условию

$$x = qy + r \text{ и } 0 \leq r < |y|.$$

Число q называется *неполным частным* от деления x на y, а число r называется *остатком* от деления x на y.

Основные сведения о целых числах (Из теории чисел)



2. Наибольший общий делитель (НОД) и взаимнопростые числа

Наибольший общий делитель целых чисел x и y это *натуральное* число n, удовлетворяющее условиям

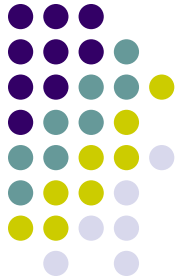
1. n есть общий делитель чисел x и y,

2. каждый общий делитель чисел x и y есть делитель числа n.

Обозначение: НОД(x, y), $\gcd(x, y)$ или просто (x, y). НОД(x, y) существует и единствен, если хотя бы одно из чисел x, y отлично от нуля. При этом НОД(x, y) есть наибольший из общих делителей чисел x и y.

Целые числа x и y называются взаимно простыми, если НОД$(x, y) = 1$

Основные сведения о целых числах (Из теории чисел)



3. Алгоритм Евклида для нахождения НОД.

Для любых натуральных чисел x и y

$$\text{НОД}(x, y) = \text{НОД}(x, r),$$

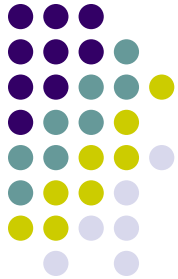
где r есть остаток от деления y на x.

Например, найдем НОД (4158,1056)

$$4158 = 3 \cdot 1056 + 990$$
$$1056 = 1 \cdot 990 + 66$$
$$990 = 15 \cdot 66 + 0.$$

$$\text{НОД}(4158, 1056) = \text{НОД}(1056, 990) = \text{НОД}(990, 66) = \text{НОД}(66, 0) = 66.$$

Основные сведения о целых числах (Из теории чисел)



4. Сравнение по модулю. Понятие вычетов

Целые числа x и y сравнимы по модулю n (n – натуральное число), если

$$x - y \text{ делится на } n,$$

т.е. существует такое целое число k, что

$$x - y = kn.$$

Обозначение:

$$x = y \pmod{n} \text{ или } x \equiv y \pmod{n}.$$

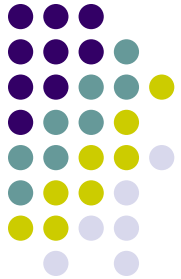
отношение сравнимости по модулю n рефлексивно, симметрично и транзитивно, т.е. есть отношение эквивалентности.

Тогда y – ***вычет*** x по модулю n.

Иногда говорят, что x ***конгруэнтно*** y по модулю n.

Множество чисел от 0 до n-1 образует то, что называется **полным множеством вычетов** по модулю n.

Основные сведения о целых числах (Из теории чисел)



5. Арифметика остатков

$$(a + b) \bmod n == ((a \bmod n) + (b \bmod n)) \bmod n$$

$$(a - b) \bmod n == ((a \bmod n) - (b \bmod n)) \bmod n$$

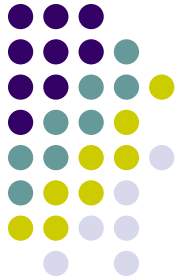
$$(a * b) \bmod n == ((a \bmod n) * (b \bmod n)) \bmod n$$

$$(a * (b+c)) \bmod n == (((a*b) \bmod n) + ((a*c) \bmod n)) \bmod n$$

6. Число Блюма

Если p и q - два простых числа, конгруэнтных 3 по модулю 4, то $n = pq$ иногда называют **целым числом Блюма.**

Основные сведения о целых числах (Из теории чисел)



7. Разложение на множители

Разложением натурального числа $n \neq 1$ в произведение простых обычно называется выражение

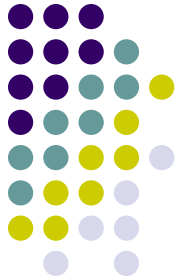
$$n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_s^{\alpha_s},$$

где $p_1, p_2, \dots, p_s$ – все различные простые делители числа n, а $\alpha_1, \alpha_2, \dots, \alpha_s$ – некоторые натуральные числа.

8. Вычисление функции Эйлера

$$\varphi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\dots\left(1 - \frac{1}{p_s}\right)$$

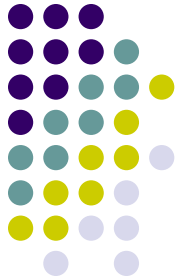
Основные сведения о целых числах (Из теории чисел)



Алгоритмы разложения на множители :

- Решето числового поля чисел (Number field sieve, **NFS**) - для чисел размером 110 и более разрядов
- Квадратичное решето (Quadratic sieve, **QS**)- для чисел размером менее 110 разрядов
- Метод эллиптической кривой (Elliptic curve method, **ECM**) – для поиска не более чем 43-разрядных множителей
- Проверка делением (Trial division) – состоит из проверки каждого простого числа, меньшего или равного квадратному корню из раскладываемого числа

Основные сведения о целых числах (Из теории чисел)



9. Малая теорема Ферма и теорема Эйлера

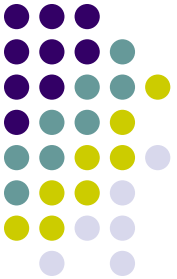
Малая теорема Ферма. Если число p простое число, а число a натуральное, взаимно простое с числом p (т.е. не делится на p), то

$$a^{p-1} = 1 \pmod{p}$$

Теорема Эйлера. Если натуральные числа n и a взаимно просты, то

$$a^{\varphi(n)} = 1 \pmod{n}$$

Основные сведения о целых числах (Из теории чисел)



10. Китайская теорема об остатках

Китайская теорема об остатках. Пусть даны попарно взаимно простые числа $m_1, m_2, \ldots, m_n$ и числа $r_1, r_2, \ldots, r_n$, удовлетворяющие условию $0 \le r_i < m_i$ для каждого i, $1 \le i \le n$. Тогда существует единственное число x, удовлетворяющее условиям

1. $x = r_i \pmod{m_i}$ для каждого i, $1 \le i \le n$,
2. $0 \le i < m_1 m_2 \ldots m_n$.

Иначе говоря, в условиях теоремы система сравнений

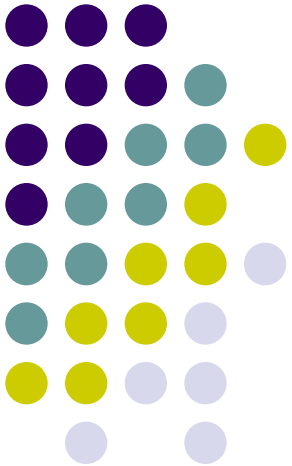
$$\begin{cases} x & = & r_1 & (\mathrm{mod}\ m_1) \\ x & = & r_2 & (\mathrm{mod}\ m_2) \\ \ldots & \ldots & \ldots\ldots & \\ x & = & r_n & (\mathrm{mod}\ m_n) \end{cases}$$

имеет единственное решение по модулю $M = m_1 m_2 \ldots m_n$.

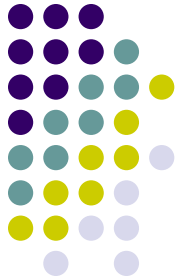
III. Про ключи

Длина ключа

Некоторые вопросы управления криптографическими ключами



Длина симметричного ключа



Безопасность симметричной криптосистемы является функцией двух факторов: надежности алгоритма и длины ключа.

Для выполнения такого вскрытия криптоаналитику требуется кусочек шифротекста и соответствующего о т-крытого текста, вскрытие грубой силой представляет собой вскрытие с известным открытым текстом . Для блоч-ного шифра криптоаналитику понадобится блок шифротекста и соответствующий открытый текст : обычно 64

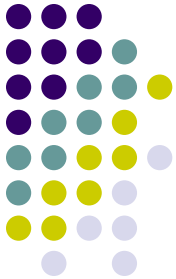
Рассчитать сложность вскрытия грубой силой нетрудно . Если используется 8-битовый ключ, то существует 2^8, или 256, возможных ключей. Следовательно, для обнаружения правильного ключа потребуется, самое бол ь-шее, 256 попыток, с 50-процентной вероятностью найти нужный ключ после половины попыток . Если длина ключа равна 56 битам, то существует 2^{56} возможных ключей. Если компьютер может проверить миллион кл ю-чей в секунду, поиск нужного ключа займет в среднем 2285 лет. Если используется 64-битовый ключ, то тому же суперкомпьютеру понадобится около 585000 лет, чтобы найти правильный ключ среди 2^{64} возможных клю-чей. Если длина ключа равна 128 битам поиск ключа займет 10^{25} лет. Возраст вселенной составляет всего 10^{10} лет, поэтому 10^{25} лет - это большое время. При 2048-битовом ключе миллион компьютеров, работая параллел ь-но и проверяя миллион ключей в секунду, потратят 10^{587} лет в поисках ключа. К этому времени вселенная давно расширится, превратившись в ничто или сожмется.

Длина открытого ключа

Разложение на множителя с помощью "квадратичного решета"

Число десятичных разря-дов в разложенном числе	Во сколько раз сложнее разложит на множители 512-битовое числ
71	>20 миллионов
80	>2 миллионов
90	250000
100	30000
120	500
129	100

Длина открытого ключа



Разложение на множители с помощью решета общего поля чисел

Количество бит	Сколько mips-лет нужно для разложения
512	30000
768	$2*10^{8}$
1024	$3*10^{11}$
1280	$1*10^{14}$
1536	$3*10^{16}$
2048	$3*10^{20}$

Разложение на множители с помощью решета специального поля чисел

Количество бит	Сколько mips-лет нужно для разложения
512	<200
768	100000
1024	$3*10^{7}$
1280	$3*10^{9}$
1536	$2*10^{11}$
2048	$4*10^{14}$

Вычислительная мощь обычно измеряется в mips-годах: годовая работа компьютера, выполняющего миллион операций в секунду (one-million-instruction-per-second, mips), или около $3*10^{13}$ операций.

Длина открытого ключа

Рекомендованные длины открытых ключей в (битах)

Год	Частное лицо	Корпорация	Правительство
1995	768	1280	1536
2000	1024	1280	1536
2005	1280	1536	2048
2010	1280	1536	2048
2015	1536	2048	2048

Длины симметричных и открытых ключей с аналогичной устойчивостью к вскрытию грубой силой

Длина симметричного ключа (в битах)	Длина открытого ключа (в битах)
56	384
64	512
80	768
112	1792
128	2304

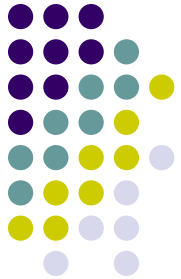
Какова же должна быть длина ключа??

1. **Сколько** стоит ваша **информация**?
2. Как **долго** она должна **храниться**?
3. Каковы **ресурсы** ваших **врагов**?

Требования к безопасности различной информации

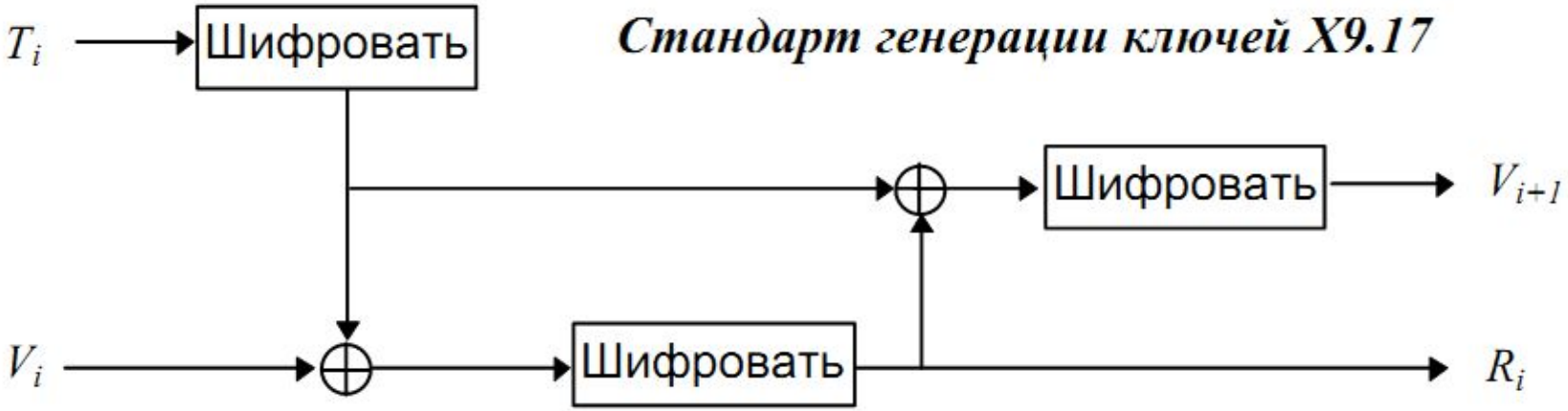
Типы трафика	Время жизни	Минимальная длина ключа (в битах)
Тактическая военная информация	минуты/часы	56-64
Объявления о продуктах, слиянии компаний, процентных ставках	дни/недели	64
Долговременны бизнес-планы	годы	64
Торговые секреты (например, рецепт кока-колы)	десятилетия	112
Секреты водородной бомбы	>40 лет	128
Личности шпионов	>50 лет	128
Личные дела	>50 лет	128
Дипломатические конфликты	>65 лет	128
Данные переписи США	100 лет	по меньшей мере 128

Некоторые вопросы управления ключами: что включает



1. Генерация ключа
2. Передача ключей
3. Использование
4. Обновление
5. Хранение
6. Резервные ключи
7. Разрушение ключей
8. Время жизни ключа

Генерация ключа



Пусть $E_K(X)$ - это X, зашифрованный DES ключом K, специальным ключом, предусмотренным для генерации секретных ключей. V_0 - это секретная 64-битовая стартовая последовательность. T - это метка времени. Для генерации случайного ключа R_i вычислим:

$$R_i = E_K(E_K(T_i) \oplus V_i)$$

Для генерации V_{i+1}, вычислим:

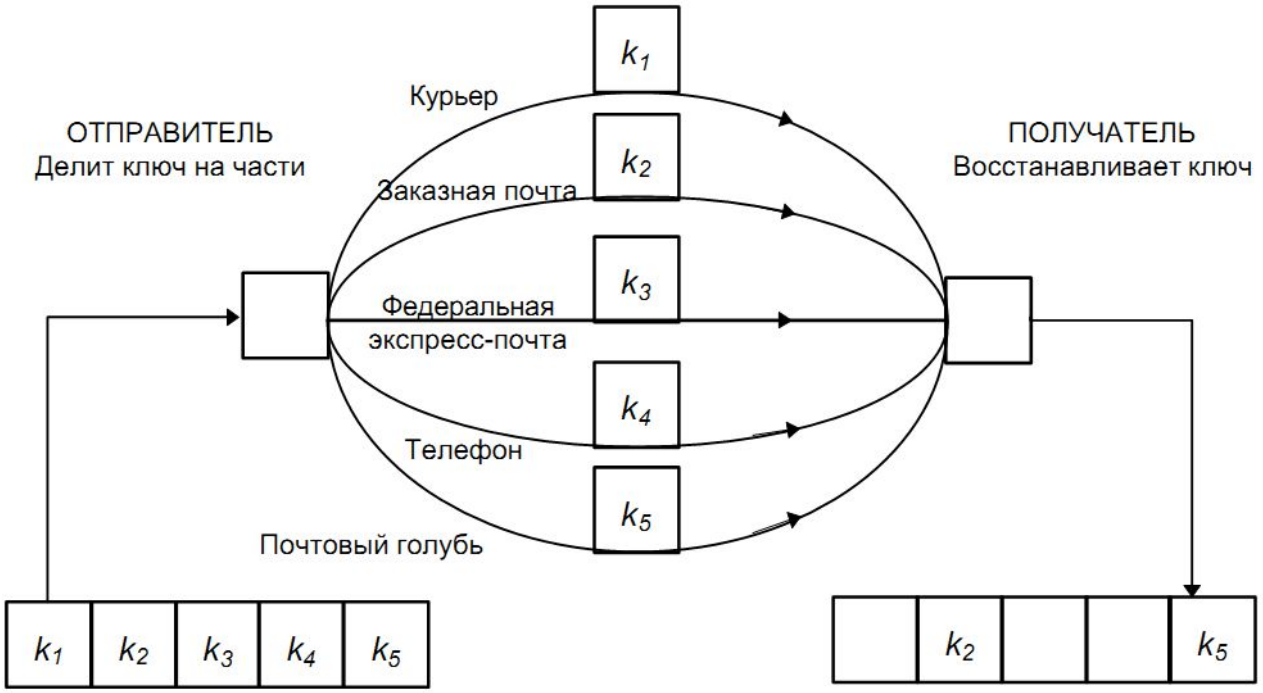
$$V_{i+1} = E_K(E_K(T_i) \oplus R_i)$$

Для превращения R_i в ключ DES, просто удалите каждый восьмой бит. Если вам нужен 64-битовый ключ, используйте ключ без изменения. Если вам нужен 128-битовый ключ, создайте пару ключей и объедините их.

Передача ключей

При использовании симметричных алгоритмов шифрования, возникает проблема передачи секретных ключей. Существует два пути:

1. При личной встрече
2. Теория разделения секрета



Проверка ключей

Как Боб узнает, получив ключ, что ключ передан Алисой, а не кем-то другим, кто выдает себя за Алису? Все просто, если Алиса передает ему ключ при личной встрече. Если Алиса посылает свой ключ через доверенного курьера, то курьеру должен доверять и Боб. Если ключ зашифрован ключом шифрования ключей, то Боб должен доверять тому, что этот ключ шифрования ключей есть только у Алисы. Если для подписи ключа Алиса использует протокол электронной подписи, Боб при проверке подписи должен доверять базе данных открытых ключей,. (Ему также придется считать, что Алиса сохранила свой ключ в безопасности.) Если Центр распределения ключей (Key Distribution Center, KDC) подписывает открытый ключ Алисы, Боб должен считать, что его копия открытого ключа KDC не была подменена.

Использование ключей

Сеансовые ключи могут быть разрешены к использованию только на определенной машине или только в определенное время. По одной из схем управления подобными ограничениями к ключу добавляется **вектор контроля** (Control Vector, CV), вектор контроля определяет для этого ключа ограничения его использования. Этот CV хэшируется, а затем для него и главного ключа выполняется операция XOR.

Результат используется как ключ шифрования для шифрования сеансового ключа . Полученный сеансовый ключ затем хранится вместе с CV. Для восстановления сеансового ключа нужно хэшировать CV и выполнить для него и главного ключа операцию XOR. Полученный результат используется для дешифрирования шифрованного сеансового ключа.

Обновление ключей

Представьте себе шифрованный канал передачи данных, для которого вы хотите менять ключи каждый день. Иногда ежедневное распределение новых ключей является нелегкой заботой . Более простое решение - генерировать новый ключ из старого, такая схема иногда называется **обновлением ключа**.

Все, что нужно - это однонаправленная функция . Если Алиса и Боб используют общий ключ и применяют к нему одну и ту же однонаправленную функцию, они получат одинаковый результат . Они могут выбрать из результата нужные им биты и создать новый ключ .

Хранение ключей

ROM-ключ

Пользователь может ввести свой ключ в систему, вставив физический носитель в считывающее устройство, встроенное в его шифрователь или подключенное к компьютерному терминалу . Хотя пользователь может использовать ключ, он не знает его и не может его скомпрометировать . Он может использовать его только тем способом и только для тех целей, которые определены вектором контроля .

Разрушение ключей

Принимая во внимание, что ключи должны регулярно меняться, старые ключи необходимо уничтожать. Старые ключи имеют определенное значение, даже если они никогда больше не используются. С их помощью враг сможет прочитать старые сообщения, зашифрованные этими ключами

Ключи должны уничтожаться надежно

Если ключ записан на бумажке, бумажку нужно разрезать и сжечь.

Если ключ - это микросхема EEPROM, то ключ необходимо переписать несколько раз.

Если ключ хра нится на диске компьютера, действительные биты соответствующего участка памяти должны быть переписаны несколько раз или диск должен быть уничтожен.

Возможная проблема состоит в том, что в компьютере ключи могут быть легко скопированы и сохранены во множестве мест.

необходимо использовать специальную программу, которая на физическом уровне искала бы на диске копию ключа даже в неиспользуемых блоках и затем стирала бы

Время жизни ключа

— Чем дольше используется ключ, тем больше вероятность его компрометации . Люди записывают ключи и теряют их. Происходят несчастные случаи. Если вы используете ключ в течение года, то вероятность его компрометации гораздо выше, чем если бы вы использовали его только один день .

— Чем дольше используется ключ, тем больше потери при компрометации ключа . Если ключ используется только для шифрования одного финансового документа на файл-сервере , то потеря ключа означает компрометацию только этого документа. Если тот же самый ключ используется для шифрования всей финансовой информации на файл-сервере, то его потеря гораздо более разрушительна.

— Чем дольше используется ключ, тем больше соблазн приложить необходимые усилия для его вскрытия - даже грубой силой. Вскрытие ключа, используемого в течение дня для связи между двумя воинскими подразделениями, позволит читать сообщения, которыми обмениваются подразделения, и создавать поддельные. Вскрытие ключа, используемого в течение года всей военной командной структурой, позволило бы взломщику в течение года читать все сообщения, циркулирующие в этой системе по всему миру, и подделывать их. В нашем мире закончившейся холодный войны какой ключ выбрали бы для вскрытия вы?

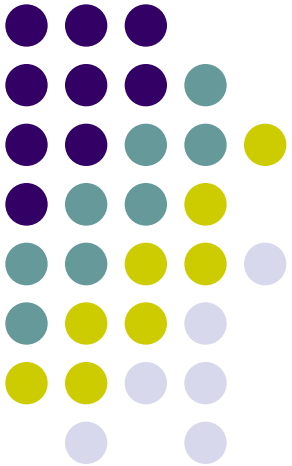
— Обычно намного легче проводить криптоанализ, имея много шифротекстов, шифрованных одним и тем же ключом.

IV. Кое-что о блочных шифрах

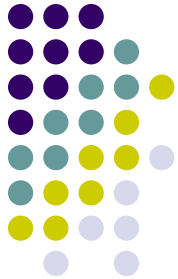
DES

IDEA

ГОСТ



Data Encryption System (DES): описание



DES представляет собой блочный шифр, он шифрует данные 64-битовыми блоками. С одного конца алг о-ритма вводится 64-битовый блок открытого текста, а с другого конца выходит 64-битовый блок шифротекста. DES является симметричным алгоритмом: для шифрования и дешифрирования используются одинаковые алг о-ритм и ключ (за исключением небольших различий в использовании ключа).

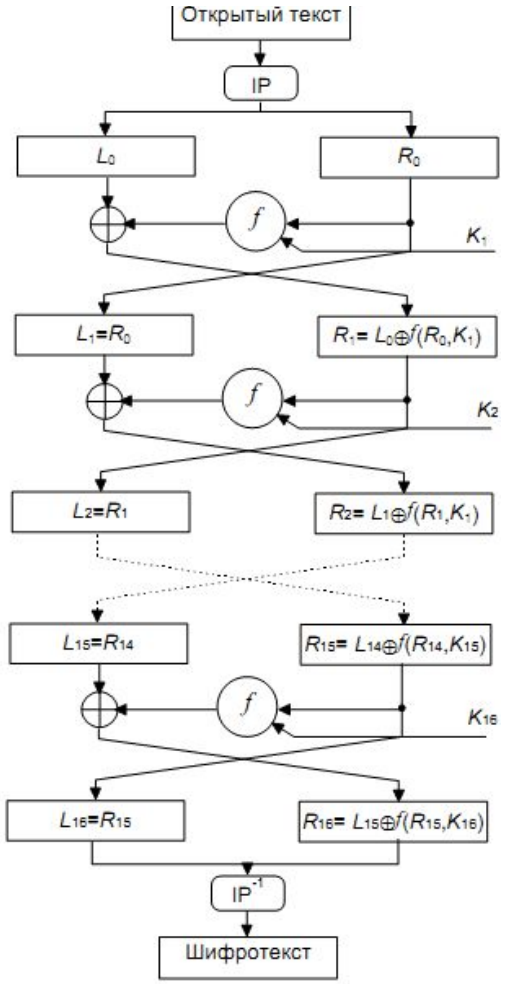
Длина ключа равна 56 битам. (Ключ обычно представляется 64-битовым числом, но каждый восьмой бит используется для проверки четности и игнорируется. Биты четности являются наименьшими значащими битами байтов ключа.) Ключ, который может быть любым 56-битовым числом, можно изменить в любой момент вр е-мени. Ряд чисел считаются слабыми ключами, но их можно легко избежать. Безопасность полностью определ я-ется ключом.

DES состоит из 16 этапов.

DES:схема алгоритма

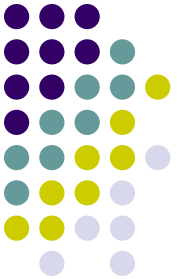
DES работает с 64-битовым блоком открытого текста. После первоначальной перестановки блок разбивается на правую и левую половины длиной по 32 бита. Затем выполняется 16 этапов одинаковых действий, называемых функцией f, в которых данные объединяются с ключом. После шестнадцатого этапа правая и левая половины объединяются и алгоритм завершается заключительной перестановкой (обратной по отношению к первоначальной).

DES.



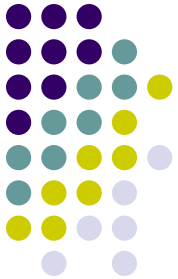
Один этап DES.

DES: дешифровка



1. Алгоритм и функция шифрования и дешифрования одни и те же
2. Отличие: ключи дешифрования используются в обратном порядке, т.е. если для шифрования применялись ключи К1,К2,…К16, то для дешифрования будут К16,К15,…К1

Стандарт шифрования IDEA



ПРОФАЙЛ:

- Блок длиной 64 бита
- Длина ключа 128 бит
- Работают с 16-битовыми подблоками

Стандарт IDEA: схема шифрования

64-битовый блок данных делится на четыре 16-битовых подблока:

X_1, X_2, X_3 и X_4. Эти четыре подблока становятся входными данными для первого этапа алгоритма. Всего в алгоритме восемь этапов. На каждом этапе четыре подблока подвергаются операциям XOR, сложениям и умножениям друг с другом и с шестью 16-битовыми подключами. Между этапами обмениваются местами второй и третий подблоки. Наконец четыре подблока объединяются с четырьмя подключами в окончательном преобразовании. На каждом этапе события происходят в следующей последовательности:

(1) Перемножаются X_1 и первый подключ.

(2) Складываются X_2 и второй подключ.

(3) Складываются X_3 и третий подключ.

(4) Перемножаются X_4 и четвертый подключ.

(5) Выполняется XOR над результатами этапов (1) и (3).

(6) Выполняется XOR над результатами этапов (2) и (4).

(7) Перемножаются результаты этапа (5) и пятый подключ.

(8) Складываются результаты этапов (6) и (7).

(9) Перемножаются результаты этапа (8) и шестой подключ.

(10) Складываются результаты этапов (7) и (9).

(11) Выполняется XOR над результатами этапов (1) и (9).

(12) Выполняется XOR над результатами этапов (3) и (9).

(13) Выполняется XOR над результатами этапов (1) и (10).

(14) Выполняется XOR над результатами этапов (4) и (10).

Выходом этапа являются четыре подблока - результаты действий (11), (12), (13) и (14).

После восьмого этапа выполняется заключительное преобразование:

(1) Перемножаются X_1 и первый подключ.

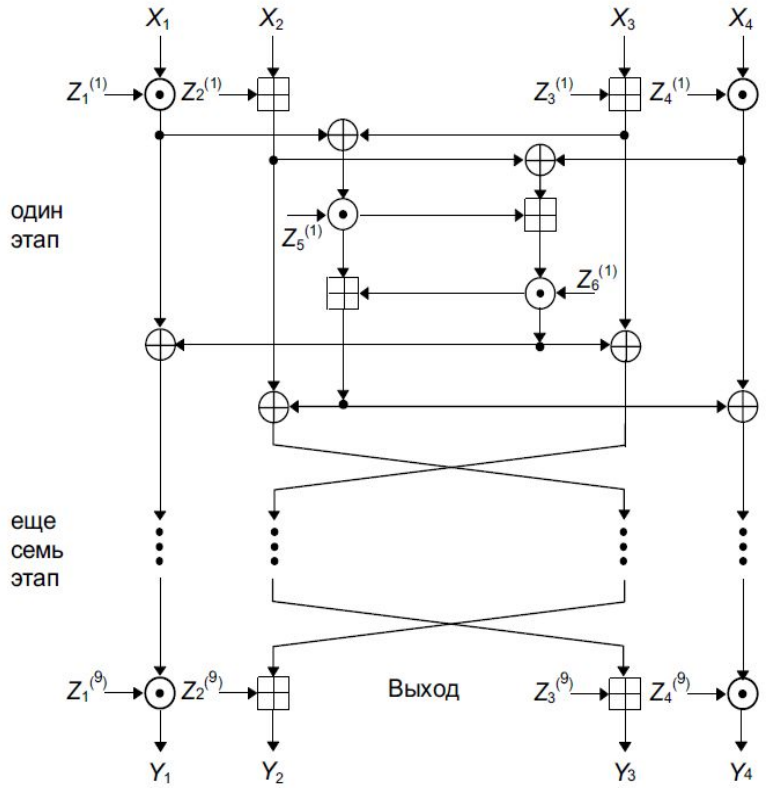
(2) Складываются X_2 и второй подключ.

(3) Складываются X_3 и третий подключ.

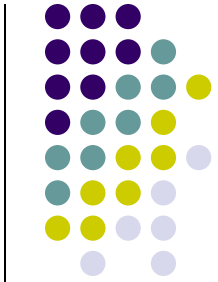
(4) Перемножаются X_4 и четвертый подключ.

Наконец четыре подблока снова соединяются, образуя шифротекст.

X_i : 16-битовый подблок открытого текста
Y_i : 16-битовый подблок шифротекста
$Z_i^{(r)}$: 16-битовый подблок ключа
$\oplus$: побитовое "исключающее или" (XOR) 16-битовых подблоков
$\boxplus$: сложение по модулю 2^{16} 16-битовых целых
$\odot$: умножение по модулю $2^{16}+1$ 16-битовых целых при условии, что нулевой подблок соответствует 2^{16}



Стандарт IDEA: подключи шифрования

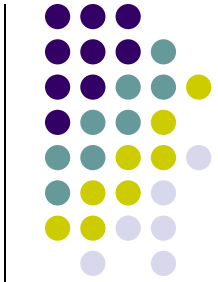


Алгоритм использует 52 из них (шесть для каждого из восьми этапов и еще четыре для заключительного преобразования). Сначала 128-битовый ключ делится на восемь 16-битовых подключей. Это первые восемь подключей алгоритма (шесть для первого этапа и два - для второго). Затем ключ циклически сдвигается налево на 25 битов и снова делится на восемь подключей. Первые четыре используются на этапе 2, а оставшиеся четыре - на этапе 3. Ключ циклически сдвигается налево на 25 битов для получения следующих восьми подключей, и так до конца алгоритма.

Подключи шифрования

Этап	Подключи шифрования					
1	$Z_1^{(1)}$	$Z_2^{(1)}$	$Z_3^{(1)}$	$Z_4^{(1)}$	$Z_5^{(1)}$	$Z_6^{(1)}$
2	$Z_1^{(2)}$	$Z_2^{(2)}$	$Z_3^{(2)}$	$Z_4^{(2)}$	$Z_5^{(2)}$	$Z_6^{(2)}$
3	$Z_1^{(3)}$	$Z_2^{(3)}$	$Z_3^{(3)}$	$Z_4^{(3)}$	$Z_5^{(3)}$	$Z_6^{(3)}$
4	$Z_1^{(4)}$	$Z_2^{(4)}$	$Z_3^{(4)}$	$Z_4^{(4)}$	$Z_5^{(4)}$	$Z_6^{(4)}$
5	$Z_1^{(5)}$	$Z_2^{(5)}$	$Z_3^{(5)}$	$Z_4^{(5)}$	$Z_5^{(5)}$	$Z_6^{(5)}$
6	$Z_1^{(6)}$	$Z_2^{(6)}$	$Z_3^{(6)}$	$Z_4^{(6)}$	$Z_5^{(6)}$	$Z_6^{(6)}$
7	$Z_1^{(7)}$	$Z_2^{(7)}$	$Z_3^{(7)}$	$Z_4^{(7)}$	$Z_5^{(7)}$	$Z_6^{(7)}$
8	$Z_1^{(8)}$	$Z_2^{(8)}$	$Z_3^{(8)}$	$Z_4^{(8)}$	$Z_5^{(8)}$	$Z_6^{(8)}$
заключительное преобразование	$Z_1^{(9)}$	$Z_2^{(9)}$	$Z_3^{(9)}$	$Z_4^{(9)}$		

Стандарт IDEA: дешифрование



- Дешифрование производится по тому же самому алгоритму
- Применяются обратные значения ключей по отношению к операциям сложения, либо умножения

IDEA: криптоанализ

Криптоанализ IDEA

Длина ключа IDEA равна 128 битам - более чем в два раза длиннее ключа DES. При условии, что наиболее эффективным является вскрытие грубой силой, для вскрытия ключа потребуется 2^{128} (10^{38}) шифрований. Создайте микросхему, которая может проверять миллиард ключей в секунду, объедините миллиард таких микросхем, и вам потребуется 10^{13} лет для решения проблемы - это больше, чем возраст вселенной. 10^{24} таких микросхем могут найти ключ за день, но во вселенной не найдется столько атомов кремния, чтобы построить такую машину.

Стандарт шифрования ГОСТ

ПРОФАЙЛ:

- Блок длиной 64 бита
- Длина ключа 256 бит
- 32 раунда шифрования
- Источник ГОСТ 28147-89

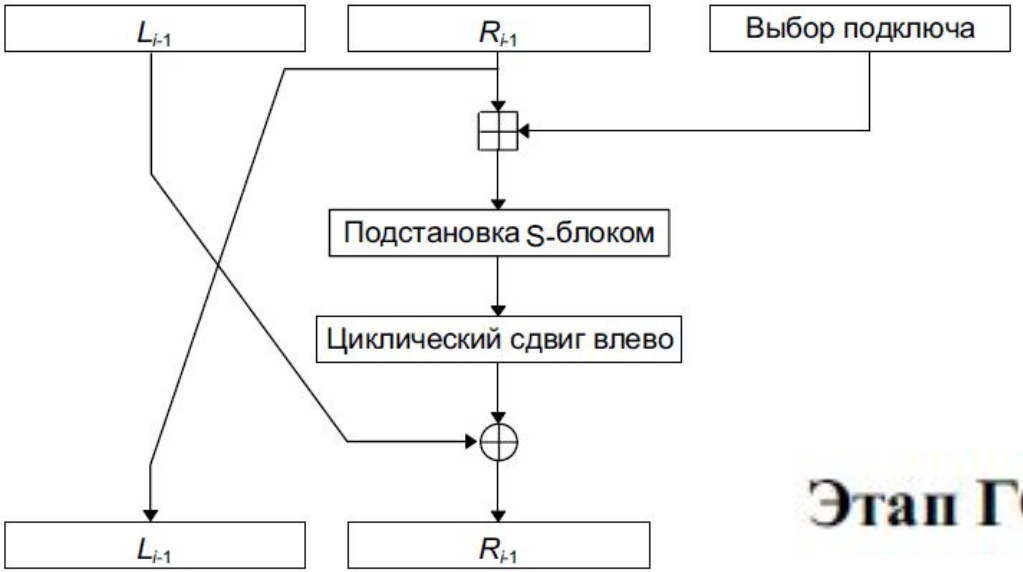
На начальном этапе:

- Текст для шифрования разбивается на левую (L) и правую половину (R) длиной по 32 бита
- На этапе i используется подключ Ki. На данном этапе:

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

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

Функция f

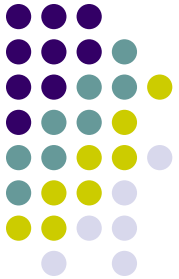


Этап ГОСТ.

Сначала правая половина и *i*-ый подключ складываются по модулю 2^{32}.
Результат разбивается на восемь 4-битовых кусочков, каждый из которых поступает на вход св о-
его S-блока. ГОСТ использует восемь различных S-блоков, первые 4 бита попадают в первый S-блок, вторые 4
4 бита - во второй S-блок, и т.д. Каждый S-блок представляет собой перестановку чисел от 0 до 15. Например,
S-блок может выглядеть как: 7, 10, 2, 4, 15, 9, 0, 3, 6, 12, 5, 13, 1, 8, 11

Выходы всех восьми S-блоков объединяются в 32-битовое слово, затем все слово циклически сдвигается влево на 11 битов. Наконец результат объединяется с помощью XOR с левой половиной, и получается новая правая половина, а правая половина становится новой левой половиной. Выполните это 32 раза

Генерация подключей и дешифрование



256-битовый ключ разбивается на восемь 32-битовых блоков: $k_1, k_2, \ldots k_8$.

На каждом этапе используется свой подключ

Использование подключей на различных этапах ГОСТ

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

Дешифрирование выполняется также, как и шифрование, но инвертируется порядок подключей k_i.

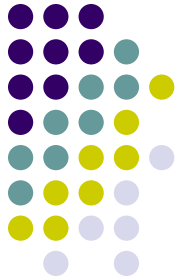
Об S-блоках

- Блоки должны выглядить только таким образом
- Производитель сам создает перестановки S-блока с помощью генератора случайных чисел

S-блоки ГОСТ

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

Криптоанализ ГОСТа (сравниваем с DES)

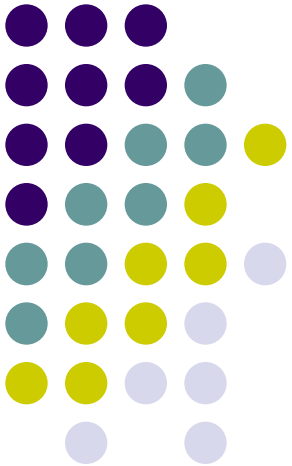


- Длина ключа 256 бит. Это много.+ еще секретные S-блоки
- 32 раунда шифрования – это тоже много.
- В ГОСТе прежде чем изменение одного вх. бита повлияет на каждый бит результата, потребуется 8 этапов. В DES лишь 5, то есть лавинный эффект слабее

IV. Немного о потоковых шифрах

А что это?...

RC4



Потоковый шифр: взгляд изнутри

Потоковые шифры преобразуют открытый текст в шифротекст по одному биту за операцию. Простейшая реализация потокового шифра показана на 3-й. **Генератор потока ключей** (иногда называемый генератором с бегущим ключом) выдает поток битов: k_1, k_2, k_3, ..., k_i. Этот поток ключей (иногда называемый бегущим ключом) и поток битов открытого текста, p_1, p_2, p_3, ..., p_i, подвергаются операции "исключающее или", и в результате получаетсяы поток битов шифротекста.

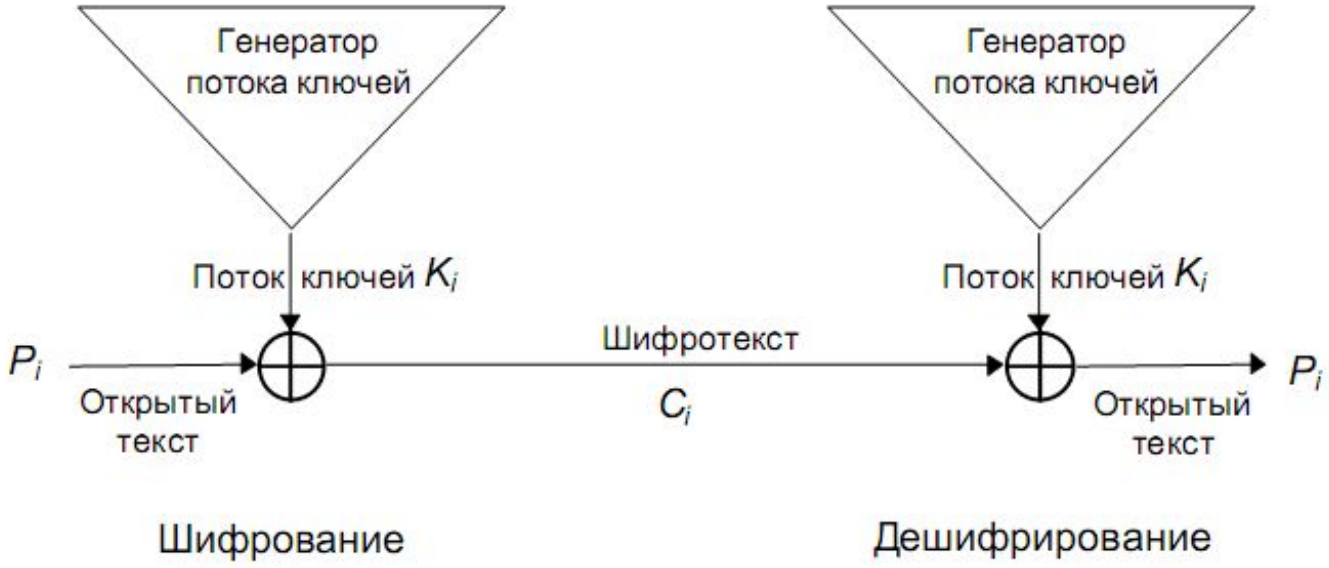
$$c_i = p_i \oplus k_i$$

При дешифрировании операция XOR выполняется над битами шифротекста и тем же самым потоком ключей для восстановления битов открытого текста.

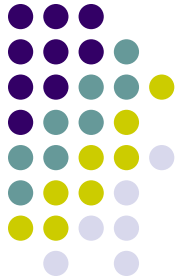
$$p_i = c_i \oplus k_i$$

Так как

$$p_i \oplus k_i \oplus k_i = p_i$$



Потоковый шифр: безопасность



Безопасность системы полностью зависит от свойств генератора потока ключей . Если генератор потока кл ю чей выдает бесконечную строку нулей, шифротекст будет совпадать с открытым текстом, и все операция будет бессмысленна. Если генератор потока ключей выплевывает повторяющийся 16-битовый шаблон, алгоритм б у дет являться простым XOR с пренебрежимо малой безопасностью Если генератор потока ключей выплевывает бесконечный поток случайных (по настоящему, а не псевдослучайных) битов, получаете одноразовый блокнот и идеальную безопасность .

На деле безопасность потокового шифра находится где-то между простым XOR и одноразовым блокнотом.

RC4: шифрование и дешифрование

ПРОФАЙЛ:

- Потоковый шифр
- Ключ переменный
- Работают с 16-битовыми подблоками

Используется S-блок размером 8*8: $S_0, S_1, \ldots, S_{255}$. Элементы представляют собой перестановку чисел от 0 до 255, а перестановка является функцией ключа переменной длины . В алгоритме применяются два счетчика, i и j, с нулевыми начальными значениями.

Для генерации случайного байта выполняется следующее :

$i = (i + 1) \bmod 256$

$j = (j + S_i) \bmod 256$

поменять местами S_i и Sj

$t = (S_i + S_j) \bmod 256$

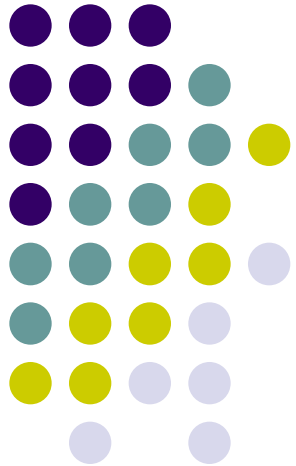
$K = S_t$

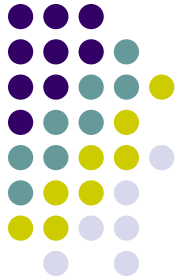
Байт K используется в операции XOR с открытым текстом для получения шифротекста или в операции XOR с шифротекстом для получения открытого текста.

V. Генерация псевдослучайных последовательностей

«Генерация случайных чисел слишком важна, чтобы оставлять её на волю случая».

математик Роберт Кавье





- **Генератор псевдослучайных чисел** (**ГПСЧ**) — алгоритм, порождающий последовательность чисел, **элементы** которой **почти независимы** друг от друга и подчиняются заданному **распределению** (обычно **равномерному**).
- При этом от **качества** используемых **ГПСЧ** напрямую зависит **качество** получаемых **результатов**.
- Любой **ГПСЧ** с ограниченными ресурсами рано или поздно **зацикливается** — начинает повторять одну и ту же последовательность чисел.
- **Длина циклов** ГПСЧ **зависит** от самого **генератора** и составляет около $2^{n/2}$, где n — размер внутреннего состояния в битах

Виды детерминированных генераторов

Линейными конгруэнтными генераторами являются генераторы следующей формы

$X_n = (aX_{n-1} + b) \bmod m$

в которых X_n - это n-ый член последовательности, а X_{n-1} - предыдущий член последовательности. Переменные a, b и m - постоянные: a - **множитель**, b - **инкремент**, и m - модуль. Ключом, или затравкой, служит значение X_0.

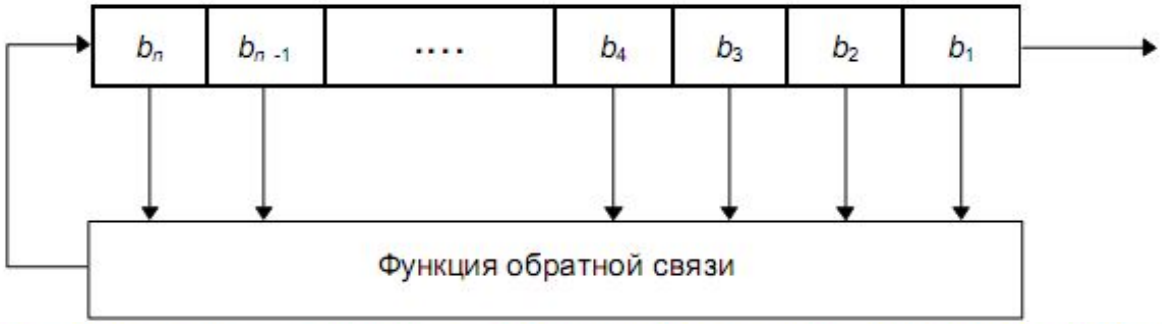
Период такого генератора не больше, чем m. Если a, b и m выбраны правильно, то генератор будет **генератором с максимальным периодом** (иногда называемым максимальной длиной), и его период будет равен m. (Например, b должно быть взаимно простым с m.)

Константы для линейных конгруэнтных генераторов

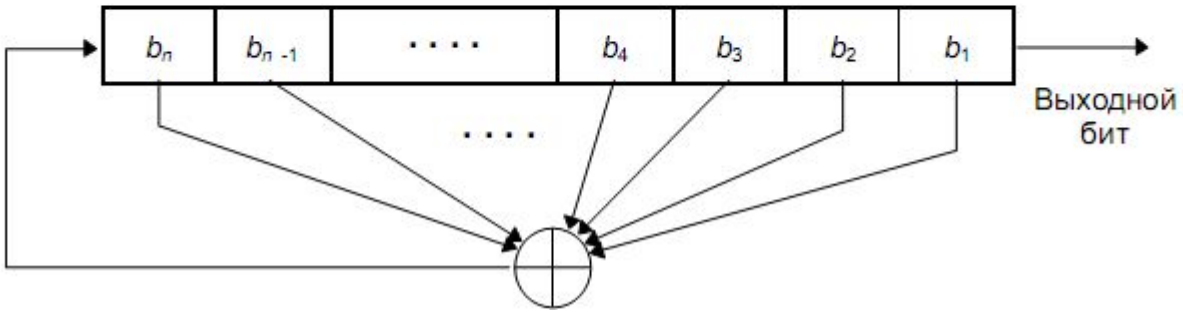
Переполняется при	a	b	m
2^{20}	106	1283	6075
2^{21}	211	1663	7875
2^{22}	421	1663	7875
2^{23}	430	2531	11979
	936	1399	6655
	1366	1283	6075
2^{24}	171	11213	53125
	859	2531	11979
	419	6173	29282
	967	3041	14406
2^{25}	141	28411	134456
	625	6571	31104
	1541	2957	14000
	1741	2731	12960
	1291	4621	21870
	205	29573	139968
2^{26}	421	17117	81000
	1255	6173	29282
2^{33}	2416	374441	1771875
2^{34}	17221	107839	510300
	36261	66037	312500
2^{35}	84589	45989	217728

Сдвиговые регистры с линейной обратной связью

Сдвиговый регистр с обратной связью состоит из двух частей: сдвигового регистра и **функции обратной связи** (см. 15th). Сдвиговый регистр представляет собой последовательность битов . (Количество битов определяется **длиной** сдвигового регистра. Если длина равна *n* битам, то регистр называется *n*-битовым сдвиговым регистром.) Всякий раз, когда нужно извлечь бит, все биты сдвигового регистра сдвигаются вправо на 1 позицию. Новый крайний левый бит является функцией всех остальных битов регистра . На выходе сдвигового регистра оказывается один, обычно младший значащий, бит . **Периодом** сдвигового регистра называется длина получаемой последовательности до начала ее повторения .

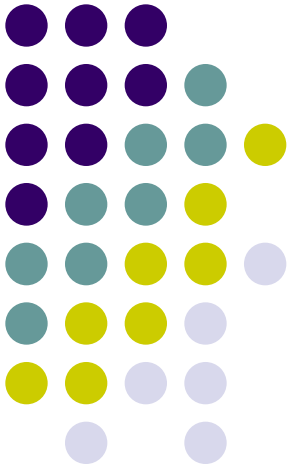


Простейшим видом сдвигового регистра с обратной связью является **линейный сдвиговый регистр с обратной связью** (linear feedback shift register, или LFSR) (см. 14th). Обратная связь представляет собой просто XOR некоторых битов регистра, перечень этих битов называется **отводной последовательностью** (tap sequence). Иногда такой регистр называется **конфигурацией Фиббоначи**. Из-за простоты последовательности обратной связи для анализа LFSR можно использовать довольно развитую математическую теорию . Криптографы любят анализировать последовательности, убеждая себя, что эти последовательности достаточно случа йны, чтобы быть безопасными . LFSR чаще других сдвиговых регистров используются в криптографии.

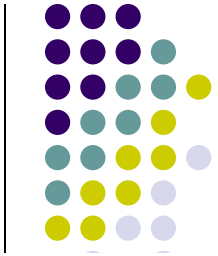


VI. Однонаправленные хэш-функции

Основы MD2



Однонаправленные хэш-функции: основы



Однонаправленная функция $H(M)$ применяется к сообщению произвольной длины M и возвращает значение фиксированной длины h.

$h = H(M)$, где h имеет длину m

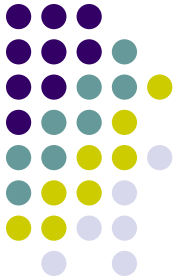
Особенности

Зная M, легко вычислить h.

Зная H, трудно определить M, для которого $H(M)=h$.

Зная M, трудно определить другое сообщение, M', для которого $H(M)= H(M')$.

Message Digest (MD2) – 128 битовая однонаправленная хэш-функция



Безопасность MD2 опирается на случайную перестановку байтов.
Эта перестановка фиксирована и зависит от разрядов π. $S_0, S_1, S_2, \ldots, S_{255}$

Чтобы выполнить хэширование сообщения M:

(1) Дополните сообщение i байтами, значение i должно быть таким, чтобы длина полученного сообщения была кратна 16 байтам.

(2) Добавьте к сообщению 16 байтов контрольной суммы.

(3) Проинициализируйте 48-байтовый блок: $X_0, X_1, X_2, \ldots, X_{47}$. Заполните первые 16 байтов X нулями, во вторые 16 байтов X скопируйте первые 16 байтов сообщения, а третьи 16 байтов X должны быть равны XOR первых и вторых 16 байтов X.

(4) Вот как выглядит функция сжатия:

$t = 0$

For $j = 0$ to 17

For $k = 0$ to 47

$t = X_t$ XOR S_t

$X_k = t$

$t = (t + j)$ mod 256

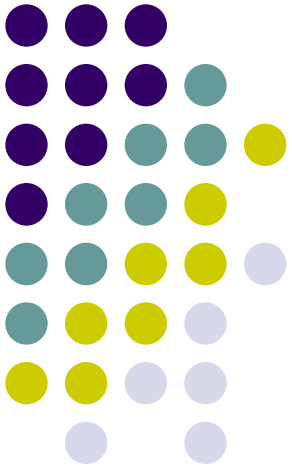
(5) Скопируйте во вторые 16 байтов X вторые 16 байтов сообщения, а третьи 16 байтов X должны быть равны XOR первых и вторых 16 байтов X. Выполните этап (4). Повторяйте этапы (5) и (4) по очереди для каждых 16 байтов сообщения.

(6) Выходом являются первые 16 байтов X.

VII. Алгоритмы шифрования с открытым ключом

RSA

ElGamal



RSA

Шифрование RSA

Открытый ключ:

n произведение двух простых чисел p и q (p и q должны храниться в секрете)

e число, взаимно простое с $(p-1)(q-1)$

Закрытый ключ:

d $e^{-1} \bmod ((p-1)(q-1))$

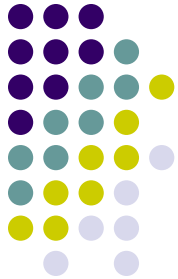
Шифрование:

$c = m^e \bmod n$

Дешифрирование:

$m = c^d \bmod n$

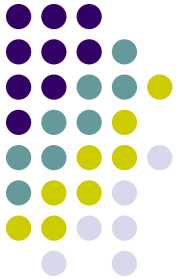
RSA: аппаратная реализация



Существующие микросхемы RSA

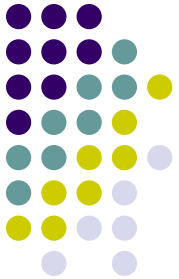
Компания	Тактовая частота	Скорость передачи в Бодах на 512 бит	Тактовые циклы для шифрования 512 бит	Технология	Битов на микросхему	Количество транзисторов
Alpha Techn.	25 МГц	13K	0.98 M	2 микрона	1024	180000
AT&T	15 МГц	19K	0.4 M	1.5 микрона	298	100000
British Telecom	10 МГц	5.IK	1 M	2.5 микрона	256	-----
Business Sim. Ltd.	5 МГц	3.8K	0.67 M	Вентильная матрица	32	-----
CalmosSyst-Inc.	20 МГц	2.8K	0.36 M	2 микрона	593	95000
CNET	25 МГц	5.3K	2.3 M	1 микрон	1024	100000
Cryptech	14 МГц	17K	0.4 M	Вентильная матрица	120	33000
Cylink	30 МГц	6.8K	1.2 M	1.5 микрона	1024	150000
GEC Marconi	25 МГц	10.2K	0.67 M	1.4 микрона	512	160000
Pijnenburg	25 МГц	50K	0.256 M	1 микрон	1024	400000
Sandia	8 МГц	IOK	0.4 M	2 микрона	272	86000
Siemens	5 МГц	8.5K	0.03 M	1 микрон	512	60000

RSA: безопасность



1. Разложить n на множители, чтобы восстановить m по c и e
2. Угадать (p-1)(q-1)

ElGamal



Шифрование ElGamal

Открытый ключ:

p простое число (может быть общим для группы пользователей)

g $<p$ (может быть общим для группы пользователей)

y $=g^x \bmod p$

Закрытый ключ:

x $<p$

Шифрование:

k выбирается случайным образом, взаимно простое с p-1

a (шифротекст) $=g^k \bmod p$

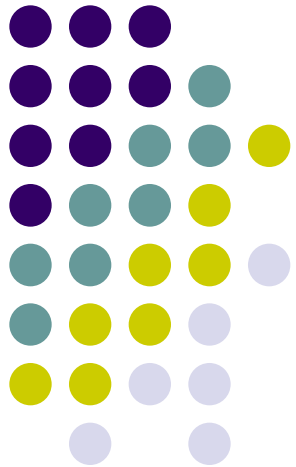
b (шифротекст)$= y^k M \bmod p$

Дешифрирование:

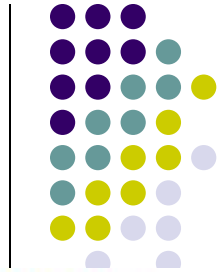
M (открытый текст) $= b/a^x \bmod p$

VIII. Алгоритмы цифровой подписи

DSA
ГОСТ Р 34.10-94
С использованием дискретных логарифмов



Цифровая подпись: отчего так притягательно?



1. Подпись достоверна. Она убеждает получателя документа в том, что подписавший сознательно подп и-сал документ.
2. Подпись неподдельна. Она доказывает, что именно подписавший, и никто иной, сознательно подписал документ.
3. Подпись не может быть использована повторно. Она является частью документа, жулик не сможет перенести подпись на другой документ.
4. Подписанный документ нельзя изменить. После того, как документ подписан, его невозможно изм е-нить.
5. От подписи не возможно отречься. Подпись и документ материальны. Подписавший не сможет вп о-следствии утверждать, что он не подписывал документ.

DSA (Digital Signature Algorithm)

Подписи DSA

Открытый ключ:

p простое число длиной от 512 до 1024 битов (может использоваться группой пользователей)

q 160-битовый простой множитель p-1 (может использоваться группой пользователей)

g $= h^{(p-1)/q} \bmod p$, где h - любое число, меньшее p-1, для которого $h^{(p-1)/q} \bmod p > 1$ (может использоваться группой пользователей)

y $= g^x \bmod p$ (p-битовое число)

Закрытый ключ:

x $< q$ (160-битовое число)

Подпись:

k выбирается случайно, меньшее q

r (подпись) $= (g^k \bmod p) \bmod q$

s (подпись) $= (k^{-1} (H(m) + xr)) \bmod q$

Проверка:

w $= s^{-1} \bmod q$

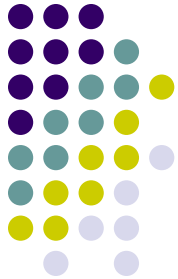
u_1 $= (H(m) * w) \bmod q$

u_2 $= (rw) \bmod q$

v $= ((g^{u_1} * y^{u_2}) \bmod p) \bmod q$

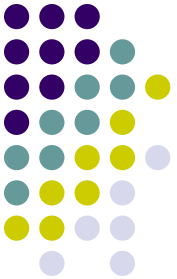
Если $v = r$, то подпись правильна.

DSA: слабости



1. Некоторые криптографы считают, что при 512 бит, алгоритм ненадежен
2. Для каждой новой подписи нужно новое значение k, которое должно выбираться случайно. Злоумышленник, зная свойства ГСЧ, может вычислить k, а по нему x. Добыв два сообщения, даже не зная k, перехватчик сможет раскрыть x и подделать подпись отправителя

ГОСТ Р 34.10-94 : подготовка и ключи



p = простое число, длина которого либо между 509 и 512 битами, либо между 1020 и 1024 битами.

q = простое число - множитель p-1, длиной от 254 до 256 битов.

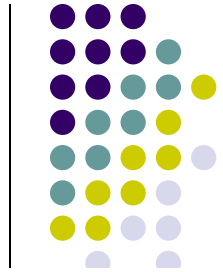
a = любое число, меньшее p-1, для которого $a^q \bmod p = 1$.

x = число, меньшее q.

$y = a^x \bmod p$.

Первые три параметра, p, q и a, открыты и могут использоваться совместно пользователями сети. Закрытым ключом служит x, а открытым - y.

ГОСТ Р 34.10-94 : подписываем и проверяем



Чтобы подписать сообщение m

(1) Алиса генерирует случайное число k, меньшее q

(2) Алиса генерирует $I = (a^* \bmod p) \bmod q$ $s = (ct + k(H(m))) \bmod q$

$r = (a^k \bmod p) \bmod q$

$s = (xr + k(H(m))) \bmod q$

Если $H(m) \bmod q = 0$, то значение хэш-функции устанавливается равным 1. Если $r = 0$, то выберите другое значение k и начните снова. Подписью служат два числа: $r \bmod 2^{256}$ и $s \bmod 2^{256}$, Алиса посылает их Бобу.

(3) Боб проверяет подпись, вычисляя

$v = H(m)^{q-2} \bmod q$

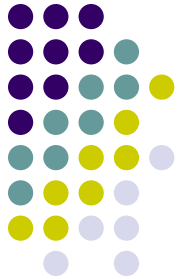
$z_1 = (sv) \bmod q$

$z_2 = ((q-r)^*v) \bmod q$

$u = ((a^{z_1} * y^{z_2}) \bmod p) \bmod q$

Если $u = r$, то подпись правильна.

ГОСТ Р 34.10-94 : особенности



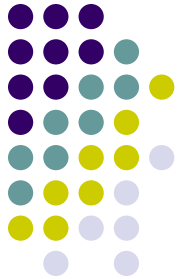
1. Различие со схемой DSA лишь в

$$s = (k^{-1} (H(m) + xr)) \bmod q,$$

Это дает другое уравнение проверки.

2. Для генерации хэш-фунции используем симметричный алгоритм ГОСТ
3. q=256. Обычно у западных криптографов 160. Они назвали это типичной привычкой русских играть в “сверхбезопасность”
4. Используется в РФ с 1995 г. для документов с грифом “Для служебного пользования” (ДСП)

Отличия от нового ГОСТ 34.10-2012



Основное отличие — в старом стандарте часть операций проводится над полем $\mathbb{Z}_p$, а в новом — над группой точек эллиптической кривой, поэтому требования налагаемые на простое число p в старом стандарте($2^{509} < p < 2^{512}$ или $2^{1020} < p < 2^{1024}$) более жёсткие, чем в новом.

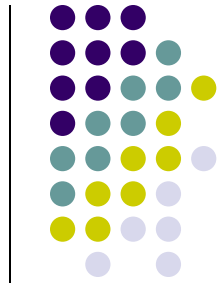
Алгоритм формирования подписи В старом стандарте в этом пункте вычисляются $\tilde{r} = a^k \mod p$ и $r = \tilde{r} \mod q$ и, если $r = 0$, возвращаемся к пункту 3. Где $1 < a < p - 1$ и $a^q = 1 \mod q$.

Алгоритм проверки подписи В старом стандарте в этом пункте вычисляется $R = (a^{z_1} y^{z_2} \mod p) \mod q$, где y — открытый ключ для проверки подписи, $y = a^d \mod p$. Если $R = r$, подпись правильная, иначе неправильная. Здесь q — простое число, $2^{254} < q < 2^{256}$ и q является делителем $p - 1$.

Использование математического аппарата группы точек эллиптической кривой, позволяет существенно сократить порядок модуля p, без потери криптостойкости.

Также старый стандарт описывает механизмы получения чисел p, q и a.

Схемы цифровой подписи с использованием дискретных логарифмов



Выберем p, большое простое число, и q, равное либо p-1, либо большому простому множителю p-1. Затем выберем g, число между 1 и p, для которого $g^q \equiv 1 \pmod{p}$. Все эти числа открыты, и могут быть совместно использованы группой пользователей. Закрытым ключом является x, меньшее q. Открытым ключом служит $y = g^x \bmod q$.

Чтобы подписать сообщение m, сначала выберем случайное значение k, меньшее q и взаимно простое с ним. Если q тоже простое число, то будет работать любое k, меньшее q. Сначала вычислим

$$r = g^k \bmod p$$

Обобщенное **уравнение подписи** примет вид

$$ak = b + cx \bmod q$$

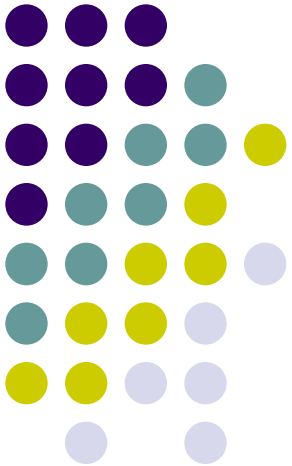
Коэффициенты a, b и c могут принимать различные значения. Каждая строка 16th предоставляет шесть возможностей. Проверяя подпись, получатель должен убедиться, что

$$r^a = g^b y^c \bmod p$$

Это уравнение называется **уравнением проверки**.

IX. Протоколы идентификации

Схема проверки подлинности и подписи Клауса Шнорра



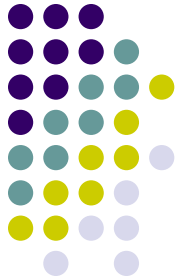
Schnorr

Генерация ключей

Для генерации пары ключей сначала выбираются два простых числа , p и q так, чтобы q было сомножителем p-1. Затем выбирается a, не равное 1, такое что $a^q \equiv 1 \pmod{p}$. Все эти числа могут быть свободно опубликованы и использоваться группой пользователей .

Для генерации конкретной пары ключей выбирается случайное число, меньшее q. Оно служит закрытым ключом, s. Затем вычисляется открытый ключ $v = a^{-s} \bmod p$.

Schnorr: протокол проверки подлинности



Протокол проверки подлинности

(1) Пегги выбирает случайное число r, меньшее q, и вычисляет $x = a^r \bmod p$. Эти вычисления являются предварительными и могут быть выполнены задолго до появления Виктора .

(2) Пегги посылает x Виктору.

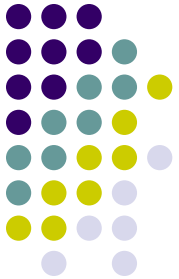
(3) Виктор посылает Пегги случайное число e, из диапазона от 0 до 2^{t-1}.

(4) Пегги вычисляет $y = (r + se) \bmod q$ и посылает y to Виктору.

(5) Виктор проверяет, что $x = a^y v^e \bmod p$.

Безопасность алгоритма зависит от параметра t. Сложность вскрытия алгоритма примерно равна 2^t. Шнорр советует использовать p около 512 битов, q - около 140 битов и t - 72.

Schnorr: протокол цифровой подписи



Протокол цифровой подписи

Алгоритм Schnorr также можно использовать и в качестве протокола цифровой подписи сообщения M. Пара ключей используется та же самая, но добавляется однонаправленная хэш-функция $H(M)$.

(1) Алиса выбирает случайное число r, меньшее q, и вычисляет $x = a^r \bmod p$. Это стадия предварительных вычислений.

(2) Алиса объединяет M и x и хэширует результат:

$e = H(M,x)$

(3) Алиса вычисляет $y = (r + se) \bmod q$. Подписью являются значения e и y, она посылает их Бобу.

(4) Боб вычисляет $x' = a^y v^e \bmod p$. Затем он проверяет, что хэш-значение для объединения M и x' равно e.

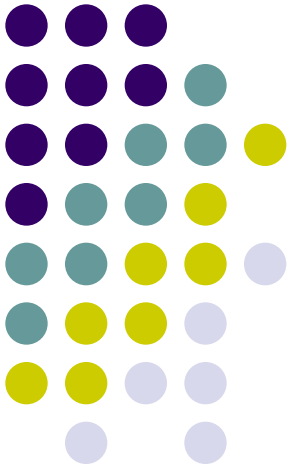
$e = H(M,x')$

Если это так, то он считает подпись верной.

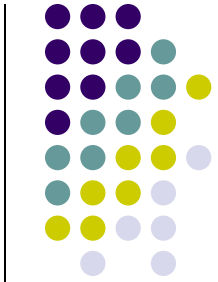
X. Алгоритмы обмена ключами

Алгоритм Диффи-Хэллмана

Трехпроходный протокол Шамира



Алгоритм Диффи-Хэллмана (Diffie-Hellman)



Подготовка:

Алиса и Боб выбирают большие целые числа **n** и **g**.

Их необязательно хранить в секрете.

Протокол:

(1) Алиса выбирает случайное большое целое число x и посылает Бобу

$X = g^x \bmod n$

(2) Боб выбирает случайное большое целое число y и посылает Алисе

$Y = g^y \bmod n$

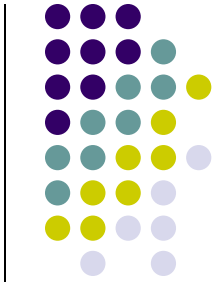
(3) Алиса вычисляет

$k = Y^x \bmod n$

(4) Боб вычисляет

$k' = X^y \bmod n$

Трехпроходный протокол Шамира



Он предполагает использование коммутативного симметричного шифра , для которого:

$E_A(E_B(P)) = E_B(E_A(P))$

Секретный ключ Алисы - A, а Боба - B. Алиса хочет послать сообщение M Бобу. Вто этот протокол.

(1) Алиса шифрует M своим ключом и посылает его Бобу

$C_1 = E_A(M)$

(2) Боб шифрует C_1 своим ключом и посылает Алисе

$C_2 = E_B(E_A(M))$

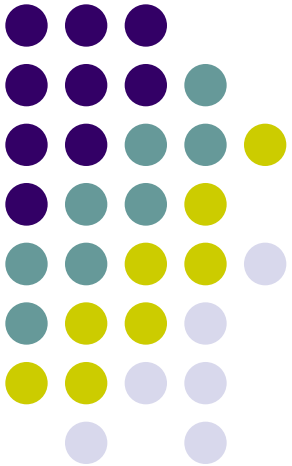
(3) Алиса расшифровывает C_2 своим ключом и посылает Бобу

$C_3 = D_A(E_B(E_A(M))) = D_A(E_A(E_B(M))) = E_B(M)$

(4) Боб расшифровывает C_3 своим ключом, получая M.

Вместо заключения: ЕЩЕ РАЗ о высоком

Квантовая криптография



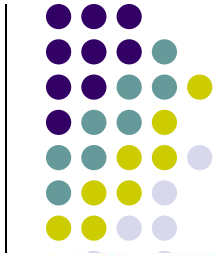
Учитывая неопределенность квантового мира…..

Вспомним квантовую механику:

В соответствии с законами квантовой механики частицы на самом деле не находятся в одном месте, а с определенной вероятностью существуют сразу во многих местах. Однако это так только до тех пор, пока не приходит ученый и не обмеряет частицу, "оказавшуюся" в данном конкретном месте. Но измерить все параметры частицы (например, координаты и скорость) одновременно невозможно. Если измерить одну из этих двух величин, сам акт измерения уничтожает всякую возможность измерить другую величину. Неопределенность является фундаментальным свойством квантового мира, и никуда от этого не денешься.

Эту неопределенность можно использовать для генерации секретного ключа. Путешествуя, фотоны колеблются в определенном направлении, вверх-вниз, влево-вправо, или, что более вероятно, под каким-то углом. Обычный солнечный свет неполяризован, фотоны колеблются во всех возможных направлениях. Когда направление колебаний многих фотонов совпадает, они являются **поляризованными**. Поляризационные фильтры пропускают только те фотоны, которые поляризованы в определенном направлении, а остальные блокируются. Например, горизонтальный поляризационный фильтр пропускает только фотоны с горизонтальной поляризацией. Повернем этот фильтр на 90 градусов, и теперь сквозь него будут проходить только вертикально поляризованные фотоны.

Как это можно использовать?

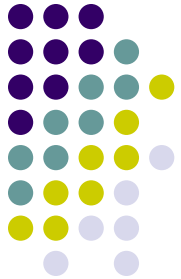


Пусть у вас есть импульс горизонтально поляризованных фотонов . Если они попробуют пройти через гор изонтальный фильтр, то у них у всех прекрасно получится . Если медленно поворачивать фильтр на 90 градусов, количество пропускаемых фотонов будет становиться все меньше и меньше, и наконец ни один фотон не про йдет через фильтр. Это противоречит здравому смыслу. Кажется, что даже незначительный поворот фильтра должен остановить все фотоны, так как они горизонтально поляризованы . Но в квантовой механике каждая частица с определенной вероятностью может изменить свою поляризацию и проскочить через фильтр . Если угол отклонения фильтра невелик, эта вероятность высока, а если он равен 90 градусам, то вероятность равна нулю . А если угол поворота фильтра равен 45 градусам , вероятность фотона пройти фильтр равна 50 процентам .

Поляризацию можно измерить в любой **системе координат**: двух направлениях, расходящихся под прямым диагональная - левая и правая диагонали . Если импульс фотонов поляризован в заданной системе координат, то при измерении в той же системе координат вы узнаете поляризацию . При измерении в неправильной системе координат, вы получите случайный результат .

Знакомьтесь! Квантовый канал передачи информации

Генерация ключа



(1) Алиса посылает Бобу последовательность фотонных импульсов. Каждый из импульсов случайным образом поляризован в одном из четырех направлений: горизонтальном, вертикальном, лево- и праводиагональном. | | / — \ — | — /

(2) У Боба есть детектор поляризации. Он может настроить свой детектор на измерение прямоугольной или диагональной поляризации. Одновременно мерить и ту, и другую у него не получится, ему не позволит квантовая механика. Измерение одной поляризации не даст измерить другую. Итак, он устанавливает свои детекторы произвольным образом: X + + X X X + X + +

В приведенном примере он может получить / | — \ / \ — / — |

(3) Боб сообщает Алисе по незащищенному каналу, какие настройки он использовал.

(4) Алиса сообщает Бобу, какие настройки были правильными. В нашем примере для импульсов 2, 6, 7 и 9.

(5) Алиса и Боб оставляют только правильно измеренные поляризации. В нашем примере они оставляют:

* | * * * \ — * — *

С помощью заранее приготовленного кода Алиса и Боб преобразуют в биты эти результаты измерений поляризации. Например, горизонтальная и леводиагональная могут означать единицу, а вертикальная и праводиагональная - ноль. В нашем примере они оба получат: 0 0 1 1

Итак, Алиса и Боб получили четыре бита. С помощью этой системы они могут генерировать столько битов, сколько им нужно. В среднем Боб правильно угадывает в 50 процентах случаев, поэтому для генерации n битов Алисе придется послать $2n$ фотонных импульсов. Они могут использовать эти биты как секретный ключ симметричного алгоритма или обеспечить абсолютную безопасность, получив достаточно битов для использования

В чем соль?

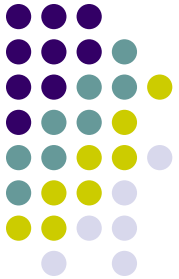
Замечательным является то, что Ева не сможет подслушать. Как и Бобу, ей нужно угадать тип измеряемой поляризации, и, как и у Боба, половина ее догадок будет неправильной. Так как неправильные измерения изменяют поляризацию фотонов, то при подслушивании она неминуемо вносит ошибки в передачу. Если это так, Алиса и Боб получат различные битовые последовательности. Итак, Алиса и Боб заканчивают протокол подобными действиями:

(6) Алиса и Боб сравнивают несколько битов своих строк. По наличию расхождений они узнают о подслушивании. Если строки не отличаются, то они отбрасывают использованные для сравнения биты и используют оставшиеся.

Улучшения этого протокола позволяют Алисе и Боб использовать свои биты даже в присутствии Евы. Они могут сравнивать только четность битовых подмножеств. Тогда, если не обнаружено расхождений, им придется отбросить только один бит подмножества. Это обнаруживает подслушивание с вероятностью 50 процентов, но если они сверят таким образом n различных битовых подмножеств, вероятность Евы подслушать и остаться незамеченной будет равна $1/2^n$.

ВЫВОД

1. КРИПТОГРАФИЯ- НАУКА СЛОЖНАЯ !
2. КРИПТОГРАФИЯ- НАУКА СЛОЖНАЯ !
3. КРИПТОГРАФИЯ- НАУКА СЛОЖНАЯ !



СПАСИБО ЗА ВНИМАНИЕ