Защита информации

П.В. Юдаев

2015

Оглавление

Часть I

Введение

текст введения

Основная литература: [1, 2, 3, 4].

Часть II

Симметричные криптосистемы

1 Алфавитные шифры

Основная цель раздела: показать, что текст, зашифрованный простым алфавитным шифром, можно взломать, используя только информацию, содержащуюся в языке, на котором написан текст.

Пусть K - множество ключей, M - множество сообщений, C - множество шифротекстов.

Определение 1.1. *Симметричный шифр* (symmetric cipher), заданный на множестве (K, M, C) - это пара алгоритмов (E, D) полиномиальной сложности:
$E : K \times M \to C$,
$D : K \times C \to M$
таких, что $\forall m \in M, k \in K\ D(k, E(k, m)) = m$ (свойство *корректности* шифра).

Шифр называется симметричным, так как для шифрования и расшифрования используется один и тот же ключ.

Замечание. Часто алгоритм шифрования E - рандомизированный алгоритм, т.е. он принимает случайное число как параметр. Алгоритм расшифрования D всегда детерминированный.

Исторически, в первых шифрах каждая отдельная буква открытого текста на естественном языке (русском, английском, греческом, и т.д.) заменялась другой буквой алфавита этого же языка по определенному правилу.

Определение 1.2. *Алфавитные шифры* - шифры, которые при преобразовании открытого текста в шифротекст заменяют один очередной символ открытого текста на один символ шифротекста по определенному правилу.

1.1 Шифр простой замены

Определение 1.3. *Шифр простой замены* (substitution cipher), или *моноалфавитный шифр* переводит каждую букву алфавита открытого текста в одну фиксированную букву алфавита шифротекста.

При шифровании текста на естественном языке любой шифр простой замены может быть довольно легко взломан путем анализа только шифротекстов с использованием свойств избыточности естественных языков. Например, буквы алфавита встречаются в тексте с разной частотой, некоторые последовательности букв не возможны и т.д.

Пример 1.1. Шифр Цезаря. Пусть алфавит содержит N букв. Кодируем (нумеруем) буквы числами от 0 до $N-1$. Ключ шифра - целое число k, $0 < k < N$. Шифрование текста - это посимвольное преобразование, циклический сдвиг алфавита на k букв. Пусть p - символ открытого текста, c - символ шифротекста.
Шифрование: $c \equiv (p + k) \bmod N$.
Расшифрование: $p \equiv (c - k) \bmod N$.

Докажем корректность шифра Цезаря: $(c-k) \bmod N \equiv ((m+k)-k) \bmod N \equiv m \bmod N$.

Основной недостаток шифра: малое количество ключей, всего N. Можно подобрать ключ перебором за N попыток, подставляя разные значения ключа, пока не получится осмысленный текст.

Рассмотрим произвольный шифр простой замены. Его ключ - произвольная фиксированная перестановка набора $(0, .., N-1)$. Каждая буква алфавита открытого текста переходит в фиксированную букву алфавита шифротекста. При использовании английского алфавита из 26 букв мощность множества ключей равна $26! \approx 2^{88}$. $\Rightarrow$ перебор по значениям ключа займет слишком долгое время.

Тем не менее, $\exists$ быстрая атака на этот шифр. Пусть злоумышленнику известен только шифротекст. Как по шифротексту найти ключ и исходный текст?

Модель атаки:

- злоумышленнику известен только достаточно длинный шифротекст;
- цель - найти ключ шифра.

Определение 1.4. *Атака с известным шифртекстом* (known ciphertext attack) - это атака, при которой злоумышленнику известен только шифротекст. Цель атаки -

узнать ключ шифра, или расшифровать данный шифротекст, или получить какую-либо новую информацию об исходном тексте.

Покажем, что шифр простой замены не устойчив к атаке с известным шифротекстом.

В любом тексте на естественном языке буквы алфавита встречаются с разной частотой. На этом основан метод взлома шифра простой замены под названием *частотный анализ*. Рассмотрим его на примере английского языка.

Частоты встречаемости (letter frequency) букв английского языка существенно различаются, см. табл. 1.

Буква	e	t	a	o	i	n	...	x	q	z
Частота	0.127	0.091	0.082	0.075	0.070	0.067	...	0.015	0.010	0.007

Таблица 1: Частоты встречаемости некоторых букв английского языка. По данным [5].

Можно дополнительно использовать частоту встречаемости диграмм (пар букв) и триграмм (трех букв). Наиболее часто встречаются диграммы *he*, *an*, *in*, *th*; триграмма *the*.

Частотный анализ шифротекста заключается в следующем. Частоты встречаемости букв в языке известны априори. Подсчитаем частоты встречаемости символов в шифротексте. Если текст достаточно длинный, они будут близки к значениям из таблицы частоты встречаемости букв языка. На основе этих данных построим гипотезу о значении ключа: какой символ шифротекста отображается в какой символ открытого текста, и расшифруем шифротекст с этим ключом. Некоторые части полученного текста окажутся читаемыми, т.е. соответствующие им отображения оказались верными. Остальные можно восстановить по смыслу текста.

1.2 Полиалфавитные шифры

1.2.1 Общие сведения

Определение 1.5. Шифр называется *полиалфавитным*, если при шифровании каждая буква открытого текста переходит в одну букву шифротекста, и конкретное преобразование зависит от позиции символа в открытом тексте.

Определение 1.6. *Шифр Виженера* (Vigener cipher). Пусть k - ключ шифра. Это строка из нескольких букв алфавита. Построим расширенный ключ k_{ext}, равный по длине открытому тексту. Для этого повторим ключ шифра достаточное количество раз. Получим строку той же длины, что и открытый текст.
Все буквы алфавита из N букв кодируются числами от 0 до $N-1$. Пусть p - исходный текст, c - шифротекст. Шифрование и расшифрование происходят посимвольно.
Шифрование: $c[i] \equiv (p[i] + k_{ext}[i]) \bmod N$
Расшифрование: $p[i] \equiv (c[i] - k_{ext}[i]) \bmod N$.

Корректность шифра Виженера очевидна.

Пример 1.2. Зашифруем сообщение `WHATANICEDAYTODAY` шифром Виженера с ключом `k = CRYPTO`:

```
K = CRYPTOCRYPTOCRYPT
P = WHATANICEDAYTODAY
C = YYYITBKTCSTMVJBPR
```

Задача 1.1. Чему равна мощность множества ключей шифра Виженера при длине ключа не более q?

У шифра Виженера очень большое множество возможных ключей. Но он, также как шифры простой замены, уязвим к атаке с известным шифротекстом.

Пусть знаем длину ключа $q = len(k)$. Тогда возьмем только каждый q-й символ шифротекста. Получим текст, зашифрованный шифром Цезаря. Его взломаем с помощью частотного анализа.

Как узнать длину ключа? Рассмотрим два способа.

1.2.2 Взлом методом Касиски

Основная идея метода Касиски (Kasiski): всякий раз, когда триграмма (три буквы) повторяется в открытом тексте и расстояние между первыми буквами триграмм кратно длине ключевого слова q, она будет зашифрована одинаково. На основе этого можно построить гипотезу о значении q.

Пример 1.3. Текст зашифрован шифром Виженера. Подчеркнуты повторяющиеся фрагменты текста и шифротекста.

```
K   = ABCD
```

K_{ext} = ABCDABCDABCDABCDABCDABCDABCD

P = CRYPTOISSHORTFORCRYPTOGRAPHY

C = CSASTPKVSIQUTGQUCSASTPIUAQJB

Создадим ассоциативный массив M, в котором ключами будут числа - делители величины расстояния между первыми символами повторившихся подстрок, а значениями - счетчики, показывающие, сколько встретился тот или иной делитель.

Например, в следующем шифротексте есть несколько повторяющихся сегментов:

DYDUXRMHTVDVNQDQNWDYDUYSNIARTJGWNQD

Расстояние между повторяющимися подстроками DYD равно 18. Увеличим на 1 значения счетчиков у чисел 18, 9, 6, 3 и 2 в массиве M. То же самое сделаем, учитывая повтор подстроки YDU. Расстояние между повторяющимися подстроками NQD равно 20. Увеличим на 1 значения счетчиков у чисел 20, 10, 5, 4 и 2. В результате, максимальное значение счетчика - у числа 2. Значит, скорее всего, длина ключа равна 2.

Также в шифротексте возможны повторы триграмм на расстояниях, не кратных длине ключа. Они носят случайный характер, поэтому значения счетчиков у их делителей, как правило, будут малы.

Если шифротекст достаточно длинный, длина ключа q будет равна значению элемента массива M, у которого значение счетчика максимальное.

1.2.3 Использование индекса совпадений

Рассмотрим на примере английского языка.

Определение 1.7. *Индекс совпадений* (coincidence index) для языка - это вероятность того, что два случайно выбранных из текста символа совпадают.

Индекс совпадений в английском языке равен $\kappa_t = 0.067$.

Вероятность того, что две случайно (равновероятно) выбранные из алфавита буквы совпадают, в английском языке равна $\kappa_a = 1/26 = 0,0385$. При этом всегда $\kappa_t > \kappa_a$.

Задача 1.2. Обосновать неравенство $\kappa_t > \kappa_a$. Указание: обозначить вероятность того, что очередной символ текста - это i-я буква алфавита из N букв, как $1/N + a_i$.

Для определения длины ключа проведем перебор по предполагаемым длинам ключа, от 1 до максимальной предполагаемой величины.

Предположим, что длина ключа равна t. Разобьем шифротекст на t частей: в i-ю часть войдут символы шифротекста в позициях $i + kt$, $k = 0, 1, 2, ...$ Если мы угадали длину ключа, то в каждой отдельной части все символы зашифрованы одной и той же буквой ключа. Поэтому индекс совпадения для каждой отдельной части будет около κ_t.

Если мы не угадали длину ключа, то в каждой отдельной части символы зашифрованы разными сдвигами и индекс совпадения для каждой отдельной части будет ниже κ_t, около κ_a.

Задача 1.3. Обосновать, что в случае, когда предполагаемая длина ключа не совпадает с истинной, индекс совпадения для каждой отдельной части шифротекста будет ниже, чем если предположение о длине ключа верно.

В первой половине XX века, до появления компьютеров, были изобретены и применялись сложные полиалфавитные шифры, реализованные на электромеханических машинах. Например, немецкая шифровальная машина Энигма, использовавшаяся во второй мировой войне, - это реализация полиалфавитного шифра. Мощность множества ключей у Энигмы достигала 2^{36}.

Но все такие шифры были взломаны: были найдены ключи. Для успешного подбора ключа шифра Виженера достаточно перехватить шифротекст небольшого объема. В случае Энигмы взлом (подбор ключа) был осуществлен только после того, как проводящие атаку на шифр выкрали шифровальную машину и смогли получать шифротексты для произвольных открытых текстов, таким образом проведя атаку с выбором открытого текста. То есть для атаки на более стойкий шифр злоумышленнику требуется больше возможностей.

2 Некоторые факты из теории вероятности для дискретных случайных величин

Основная цель раздела: напомнить необходимые сведения из теории вероятности для дискретных случайных величин.

2.1 Общие сведения

Пусть U - конечное множество.

Определение 2.1. *Распределение вероятности* P над U - это функция $P: U \to [0,1] : \sum\limits_{u \in U} P(u) = 1$.

Пример 2.1. Равномерное распределение: $\forall u \in U\ P(u) = 1/|U|$.

Точечное распределение: $P(u_0) = 1, \forall u \neq u_0\ P(u) = 0$.

Определение 2.2. Любое подмножество $A \subseteq U$ называется *событием*. Вероятность события $A \subseteq U\ P(A) = \sum\limits_{u \in A} P(u)$.

По определению, $P(U) = 1$, $P(\emptyset) = 0$.

Определение 2.3. *Случайная величина* - это функция $X: U \to V$.

Определение 2.4. Рассмотрим произвольное множество V. *Распределение вероятности случайной величины* $X: U \to V$ *на множестве* V - это $P(X = v) = P(X^{-1}(v))$, при этом $\forall v \in V\ X^{-1}(v) \subseteq U$ - событие на множестве U.

Определение 2.5. *Равномерно распределенная* случайная величина на множестве U - это тождественная функция X: $\forall u \in U\ X(u) = u$ и $\forall u \in U\ P(X = u) = 1/|U|$.

Пусть $x \in \{0,1\}^n$. Значение функции $lsb_k(x)$ определим как k младших (правых) битов строки x.

Пример 2.2. $U = \{0,1\}^n$, $Y \in U$, случайная величина $X(Y) = lsb_2(Y) \in \{0,1\}^2$. Пусть $A = \{a \in U | lsb_2(a) = 00\}$. Если на множестве U равномерное распределение вероятности, то $P(A) = 1/4$ и $P(X = 00) = 1/4$.

Определение 2.6. События A, B *независимы*, если $P(A\&B) = P(A) \cdot P(B)$.

Определение 2.7. Случайные величины X, Y, принимающие значения из V, *независимы*, если $\forall a, b \in V\ P(X = a\ \&\ Y = b) = P(X = a) \cdot P(Y = b)$.

Пример 2.3. Пусть X, Y - независимые, равномерно распределенные случайные величины на $\{0,1\}$. Пусть $Z = X + Y$. Тогда $P(Z = 2) = P(X = 1 \& Y = 1) = P(X = 1) \cdot P(Y = 1) = 1/2 \cdot 1/2 = 1/4$.

Пример 2.4. Пусть X, Y - равномерно распределенные случайные величины на $\{0,1\}$ и $Y = 1 - X$. Тогда $P(Z = 2) = P(X = 1 \& Y = 1) = 0 \neq P(X = 1) \cdot P(Y = 1)$.

Определение 2.8. *Условная вероятность* события A при условии наступления события B - это величина $P(A|B) = P(A\&B)/P(B)$

Замечание. В определении условной вероятности вертикальная черта | - это формальный символ, а не знак "или". Операция "или" обозначается символом $\vee$.

Пример 2.5. Рассмотрим множество $U = \{n|n \in \mathbb{N}, n \leq 60\}$. Пусть $x \in U$ - равномерно распределенная случайная величина. Событие A: $\{x = 2k, k \in \mathbb{N}\}$. Событие B: $\{x = 6t, t \in \mathbb{N}\} \cup \{x = 6t + 1, t \in \mathbb{N}\}$. $P(A\&B) = 1/6$. Условная вероятность $P(A|B) = (1/6)/(1/2) = 1/3$.

Операция $\oplus$ для чисел $0, 1$ - это сложение по модулю 2. Пусть $a, b \in \{0,1\}^n$ - строки длины n. Определим операцию $\oplus$ над строками: $c = a \oplus b$, если $\forall i \in \{0, ..., n-1\}$ $c[i] = a[i] \oplus b[i]$.

Теорема 2.1. Пусть X - произвольная случайная величина, принимающая значения из $\{0,1\}^n$, Y - равномерно распределенная случайная величина на $\{0,1\}^n$, и X, Y независимы. Тогда $Z = X \oplus Y$ - равномерно распределенная случайная величина на $\{0,1\}^n$.

Доказательство. Операция над строками производится почленно, поэтому достаточно доказать для $n = 1$.

Пусть случайная величина $X : P(X = 0) = p_0, P(X = 1) = p_1 = 1 - p_0$. Т.к. X, Y независимы, то $P\big((X,Y) = (0,0)\big) = P(X = 0) \cdot P(Y = 0) = p_0/2$.

Тогда $P(Z = 0) = P\big((X,Y) = (0,0) \vee (X,Y) = (1,1)\big) = P\big((X,Y) = (0,0)\big) + P\big((X,Y) = (1,1)\big) = p_0/2 + p_1/2 = 1/2$.

$P(Z = 0) = 1/2 \Rightarrow P(Z = 1) = 1/2 \Rightarrow Z$ - равномерно распределенная случайная величина на $\{0,1\}$. □

Задача 2.1. Обобщить эту теорему для сложения по модулю n.

2.2 Парадокс дня рождения

Теорема 2.2. Пусть $X_1, ..., X_n \in U$ - независимые одинаково распределенные случайные величины. Пусть $|U| > 500$. Тогда, если $n = 1.2 \cdot \sqrt{|U|}$, то $P(\exists i \neq j : X_i = X_j) \geq 1/2$.

Доказательство. Легко видеть, что для распределения, отличающегося от равномерного, вероятность совпадения будет выше, чем при равномерном распределении. Поэтому проведем доказательство для равномерно распределенных случайных величин.

Обозначим $N = |U|$. Пусть значение случайной величины $X_1 = a_1$. Пусть $X_2 = a_2$. Вероятность, что они не совпадут, равна $1 - \frac{1}{N}$.

Добавим к набору (a_1, a_2), $a_1 \neq a_2$ третью случайную величину, принявшую значение $X_3 = a_3$. Вероятность того, что при этом нет совпадений, равна $1 - \frac{2}{N}$. И так далее, вплоть до значения последней случайной величины, для которого вероятность несовпадения с предыдущими значениями будет $1 - \frac{n-1}{N}$.

Поэтому вероятность того, что значения всех случайных величин будут различными, равна

$p(n) = 1 \cdot (1 - \frac{1}{n}) \cdot (1 - \frac{2}{N}) \cdot ... \cdot (1 - \frac{n-1}{N})$.

Т.к. $e^{-x} \geq 1 - x \; \forall x \in \mathbb{R}$, то $p(n) \leq \prod_{i=1}^{n-1} e^{-x} = e^{-\frac{1}{N}\sum_{i=1}^{n-1} i} = e^{-\frac{n(n-1)}{2N}}$

Т.к. $n = 1.2 \cdot \sqrt{N}$, то $-\frac{n(n-1)}{2N} = -0.72 + \frac{0.6}{\sqrt{N}}$. Тогда $p(n) \leq e^{-0.72+\frac{0.6}{\sqrt{N}}}$.

Решая неравенство $e^{-0.72+\frac{0.6}{\sqrt{N}}} < 0.5$ относительно N, находим, что оно верно при $N > 499.3$. Поэтому при $N \geq 500$ $p(n) < 0.5$.

Вероятность, что в наборе $(a_1, ..., a_n)$ найдется два равных значения, равна $1 - p(n) > 0.5$. □

Пример 2.6. Пусть $U = \{0, 1\}^{128}$. Тогда, если мы приняли $1.2 \cdot 2^{64}$ случайных сообщений из U, то с вероятностью более 0.5 там найдутся два равных сообщения.

Замечание. Очевидно, что чем больше мощность множества U, тем меньший коэффициент перед $\sqrt{|U|}$ необходим, чтобы выполнялось неравенство $p(n) < 0.5$. Но он не может быть меньше, чем $\sqrt{2ln(2)} \approx 1.177$.

График зависимости вероятности наличия совпадения от объема выборки при $|U| = 10^6$ изображен на рис. 1.

3 Одноразовый блокнот

Основная цель раздела: построить шифр, который не может взломать злоумышленник, не ограниченный в вычислительных ресурсах.

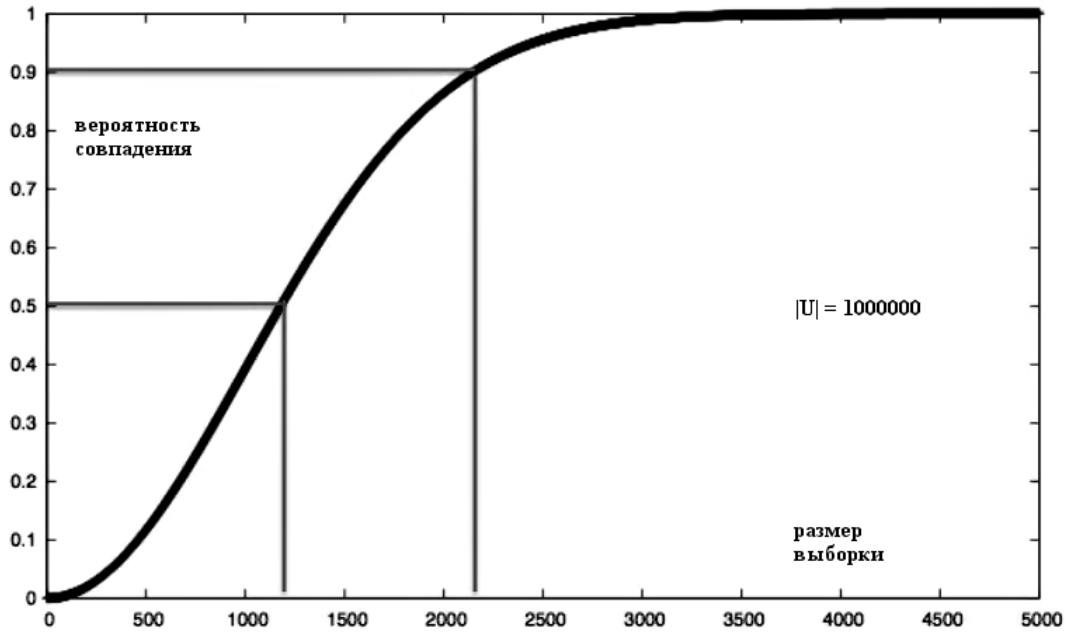

Рис. 1: Зависимость вероятности наличия совпадения от объема случайной выборки, когда мощность множества значений равномерно распределенной случайной величины равна 10^6. По материалам [4].

Шифр одноразовый блокнот (one time pad) изобретен Вернамом в 1917 году.

Определение 3.1. *Шифр одноразовый блокнот.* Множества ключей, сообщений и шифротекстов совпадают: $M = C = K = \{0,1\}^n$. Ключ - случайная, равномерно распределенная строка той же длины, что и сообщение.
Шифрование: $c = E(k,m) = k \oplus m$.
Расшифрование: $D(k,c) = k \oplus c$.

Корректность шифра очевидна: $D(k, E(k,m)) = k \oplus k \oplus m = m$.

Свойства одноразового блокнота: очень быстрая работа, но очень длинный ключ.

Исследуем, является ли одноразовый блокнот стойким шифром.

Какими возможностями обладает злоумышленник и какова его цель? Так как в одноразовом блокноте ключ используется только один раз, реалистичной моделью атаки является следующая: злоумышленник знает только шифротекст. Т.е. злоумышленник проводит атаку с известным шифротекстом.

Возможные цели злоумышленника: узнать ключ шифра, или уметь расшифровывать любые шифротексты, или узнать что-нибудь об открытом тексте по шифротексту.

Определение 3.2. Шифр называется *стойким по Шеннону*, если по шифротексту злоумышленник не может узнать никакой информации о сообщении дополнительно к тому, что он знает априори, до анализа шифротекста.

Определение 3.3. Шифр (E, D) над множествами (K, M, C) называется *абсолютно стойким* (perfectly secure), если $\forall m_0,\ m_1:\ len(m_0) = len(m_1)$ и $\forall c \in C\ P(E(k, m_0) = c) = P(E(k, m_1) = c)$ и ключ шифра k - это значение равномерно распределенной случайной величины на множестве ключей K.

Теорема 3.1. Одноразовый блокнот - это абсолютно стойкий шифр.

Доказательство. $P(E(k, m) = c) = \big|k \in K:\ E(k, m) = c\big|/|K|$. Значит, если $\forall m, c\ |k \in K : E(k, m)$ *const*, то шифр абсолютно стойкий.

Пусть $m \in M$ и $c \in C$. Сколько ключей отображают m в c? $\forall m, c\ k = m \oplus c \Rightarrow |k \in K:\ E(k, m) = c| = 1$. Значит, одноразовый блокнот - абсолютно стойкий шифр. □

Теорема 3.2. Если шифр абсолютно стойкий, то $|K| \geq |M|$.

Доказательство. От противного. Пусть $|K| < |M|$. Пусть $E(k_0, m_0) = c_0$. Пусть $S = \{m|\exists k \in K : D(k, c_0) = m\}$, $|S| \leq |K| < |M| \Rightarrow \forall m_1 \in M \setminus S:\ P[c = E(k, m_1)] = 0$ и при этом $P[c = E(k, m_0)] > 0$. □

Т.е. у абсолютно стойкого шифра длина ключа не меньше длины сообщения. Причем этот ключ знают заранее и отправитель, и получатель сообщения. Не очень практично.

Замечание. Двухразовый блокнот - не стойкая криптосистема. Используем один ключ два раза: $c_1 = m_1 \oplus k$, $c_2 = m_2 \oplus k$. Злоумышленник знает оба шифротекста и вычисляет $c_1 \oplus c_2 = m_1 \oplus m_2$. Естественный язык или протокол обмена данными часто имеет достаточно избыточности, чтобы по значению $m_1 \oplus m_2$ узнать значения m_1 и m_2.

Пример такой атаки: Project Venona. При передаче секретной информации по открытым каналам СССР применял шифр одноразовый блокнот, при этом некоторые фрагменты ключа использовались более одного раза. Поэтому США удалось дешифровать часть сообщений.

4 Поточные шифры

Основная цель раздела: одноразовый блокнот - абсолютно стойкий шифр, но он имеет очень длинный ключ - случайную последовательность. Построим созданную на основе короткого ключа псевдослучайную последовательность чисел, вычислительно не отличимую от случайной, и используем ее для построения поточного шифра.

4.1 Генератор псевдослучайных чисел

Определение 4.1. *Генератор псевдослучайных чисел* (ГПСЧ) (pseudo random number generator) - это функция $G: \{0,1\}^s \to \{0,1\}^n$, где $n \gg s$, вычислимая детерминированным алгоритмом за полиномиальное от n время.

Определение 4.2. Пусть $G(\cdot)$ - генератор псевдослучайных чисел. Тогда *поточный шифр* (stream cipher) - это пара алгоритмов (E, D) :
$E(k, m) = m \oplus G(k)$,
$D(k, c) = c \oplus G(k)$

Корректность поточного шифра очевидна.

Задача 4.1. Может ли поточный шифр быть абсолютно стойким?

Исследуем, какими свойствами должен обладать ГПСЧ, чтобы его можно было использовать в поточном шифре в предположении, что злоумышленник имеет конечную вычислительную мощность. Для этого понадобятся некоторые определения.

4.2 Классы задач и алгоритмов

Пусть $len(x)$ - длина записи x в бинарном алфавите. Например, если x - натуральное число, $len(x) = log_2(x)$.

Определение 4.3. PT - класс детерминированных алгоритмов, имеющих *полиномиальное время работы* (polynomial time). Алгоритм $A \in$ PT, если $\exists k \in$ N : $\forall x \in X : len(x) \leq n$ время работы $A(x) \leq n^k$.

Определение 4.4. *Задача разрешимости* (decision problem) - это задача, имеющая два варианта ответа.

Пример 4.1. Дано описание функции $f : X \to Y$. Существует ли $x \in X : \ f(x) = y_0 \in Y$?

Пример 4.2. Дана КНФ f. Существует ли $x : \ f(x) = 1$? (Это задача выполнимости.)

Определение 4.5. *Задача поиска* (search problem) - задача найти элемент множества, для которого выполнено заданное отношение, или установить, что таких элементов нет.

Пример 4.3. Дано описание функции $f : X \to Y$. Найти $x \in X : \ f(x) = y_0 \in Y$.

Определение 4.6. Функция $\varepsilon(n)$ называется *пренебрежимо малой*, если $\forall$ константы $k > 0$ существует константа $n_0 = n_0(k) : \varepsilon(n) < 1/n^k \ \ \forall n > n_0$.

T.e. при $n \to \infty \ \forall k \in \mathbb{N} \ \varepsilon(n) = o(1/n^k)$.

На практике, когда число n фиксировано, пользуются фиксированным ε, напр. $\varepsilon = 2^{-80}$.

Далее в тексте всегда будет подразумеваться, что $\varepsilon(n)$ - пренебрежимо малая функция.

Определение 4.7. PPT - класс полиномиальных рандомизованных алгоритмов (probabilistic polynomial time). Алгоритм $A \in$ PPT, если

1. Он имеет полиномиальное время работы, может "подбрасывать монету" и принимать случайные решения, и
2. Если задача имеет два вариант ответа, то алгоритм дает верный ответ с вероятностью больше $2/3$.

Замечание. (*) Можно требовать, чтобы алгоритм A давал правильный ответ с вероятностью более $1/2 + c$, где $c > 0$ - любая константа. Тогда алгоритм, который дает правильный ответ с вероятностью более фиксированной величины $c' < 1$, получается из алгоритма A следующим образом. Повторим алгоритм A t раз и примем

решения простым большинством по t результатам. Величина t не зависит от длины входа. $t = t(c, c')$, растет при $c \to 0$ и при $c' \to 1$. Доказательство этого факта использует свойства суммы независимых одинаково распределенных случайных величин и неравенство Чебышева: $P(|X - EX| > \varepsilon) < DX/\varepsilon^2$.

Определение 4.8. P - класс задач, разрешимых детерминированными алгоритмами за полиномиальное время.

Определение 4.9. PP (probabilistic polynomial) - класс задач, разрешимых рандомизированными алгоритмами за полиномиальное время. Если задача имеет два ответа, то вероятность правильного ответа алгоритма - более $1/2$.

Определение 4.10. BPP (bounded probabilistic polynomial) - это класс задач, которые разрешимы рандомизированными алгоритмами за полиномиальное время. При этом, если задача имеет два ответа, то алгоритм дает верный ответ с вероятностью больше константы $2/3$.

Легко видеть, что BPP $\subseteq$ PP. В чем преимущество класса BPP по сравнению с PP?

Пусть вероятность правильного ответа $c = 2/3$ нам не достаточна. Хотим достичь некоторой $c' < 1$. Для этого алгоритм из BPP достаточно повторить t раз и принять решение простым большинством по t результатам, причем величина t не зависит от длины входа: $t = t(c - 1/2, c')$, растет при $c \to 1/2$ и при $c' \to 1$. Для алгоритма из PP этого сделать нельзя, т.к. вероятность правильного ответа не отделена от $1/2$.

4.3 Предсказуемый ГПСЧ

Пусть x - строка или массив. Обозначим набор из элементов x в позициях i, j как $x|_{i,j}$. Позиции нумеруются с 1.

Определение 4.11. ГПСЧ $G: K \to \{0,1\}^n$ называется *предсказуемым* (predictable), если $\exists$ алгоритм $A \in$ PPT и $\exists i(n)$, $1 \le i \le n-1$: при равновероятном выборе ключа k из K $P[A(G(k))|_{1,\dots,i} = G(k)|_{i+1}] > 1/2 + \delta(n)$ для не пренебрежимо малой функции $\delta(n)$. Вероятность вычисляется по случайному выбору ключа k и действиям алгоритма A.

Замечание. Угадывание путем подбрасывания монеты дает
$P[A(G(k))|_{1,\ldots,i} = G(k)|_{i+1}] = 1/2$.

Предсказуемость - нежелательное свойство для ГПСЧ. Пусть $\exists i_0 \in \mathrm{N} : \forall i > i_0$ можно предсказать только один следующий бит, т.е. $\exists A : \forall i > i_0 \; A(G(k)|_{1,..i}) = G(k)|_{i+1}$. Пусть злоумышленник знает начало открытого текста до позиции i_0 включительно. Тогда он сможет дешифровать весь остальной шифротекст бит за битом.

Задача 4.2. Дать определение непредсказуемого ГПСЧ.

Задача 4.3. Пусть $G: \; K \to \{0,1\}^n : \forall k \bigoplus\limits_{i=1,..,n} G(k)|_i = 0$. Является ли он предсказуемым?

Случайность и равномерность последовательности ПСЧ можно проверить статистическими тестами.

Примеры тестов на случайность:

- количество серий из 0 и серий из 1 длины k для разных k в последовательности длины n лежит в определенных пределах
- количество фиксированных наборов битов длины k в последовательности длины n лежит в определенных пределах
- тест Маурера: насколько последовательность быть сжата без потерь информации.

Пример теста на равномерность:

- проверка гипотезы о законе распределения случайной величины:
  количества разных наборов битов длины длины k должны быть примерно равны.

Если последовательность чисел случайная, она удовлетворяет статистическим тестам. Однако, из того, что последовательность ПСЧ удовлетворяет статистическим тестам, не следует, что ГПСЧ непредсказуемый.

Пример 4.4. ГПСЧ $X_n = frac((n + X_0)\sqrt{2})$, где $frac$ - операция взятия дробной части, - равномерный на $[0,1]$, т.к. $\sqrt{2}$ - иррациональное число, но он предсказуемый.

Пример 4.5. Линейный конгруэнтный генератор (linear congruential generator). Следующее значение ГПСЧ определяется по формуле $X_{n+1} = aX_n + b \bmod m$. Параметры генератора: $a, b, m \in \mathbb{N}$, $a < m$, $b < m$. При определенном выборе параметров этот ГПСЧ удовлетворяет статистическим тестам.

Утверждение 4.1. При известном m параметры a, b линейного конгруэнтного генератора можно определить по псевдослучайной последовательности длины 3.

Доказательство. Пусть m известно, a, b - не известны. Если известны три последовательных значения X_1, X_2, X_3, то a, b - решения системы 2 линейных уравнений в группе Z_m:

$$\begin{cases} X_2 \equiv aX_1 + b \bmod m \\ X_3 \equiv aX_2 + b \bmod m \end{cases}$$

$$\begin{cases} a \equiv (X_2 - X_3)(X_1 - X_2)^{-1} \bmod m \\ b \equiv X_2 - aX_1 \bmod m \end{cases}$$

□

4.4 Криптографически стойкий ГПСЧ

Определение 4.12. Алгоритм A называется *оракулом* (oracle), отвечающим на фиксированный вопрос, если он принимает на вход какие-то данные и дает один из двух ответов: выдает значение 1 (ответ - "ДА") или 0 (ответ - "НЕТ"). Оракул может ошибаться.

Пусть оракул A пытается отличить два события C и D. Обозначим $A(C)$ - ответ оракула A при условии, что произошло событие C. Тогда $P[A(C) = 1]$ - это вероятность того, что A даст ответ 1 при условии события C. Если оракул не может отличить два события, то его ответ не зависит от них. В этом случае $P[A(C) = 1] = P[A(D) = 1]$. Аналогичный результат получим, если будем пытаться угадать, какое событие произошло.

Определение 4.13. Пусть оракул A пытается отличить события C и D. Назовем *преимуществом оракула* A над угадыванием, или просто преимуществом (advantage), величину

$Adv[A] = |P[A(C) = 1] - P[A(D) = 1]|$.

Определение 4.14. алгоритм A называется *оракулом для ГПСЧ* G, если он принимает на вход значение $G(k)$ и выдает значение 0 или 1 со следующим смыслом: $A(G(k)) = 0$, если A считает $G(k)$ не случайной последовательностью, $A(G(k)) = 1$ иначе.

Обозначим $x \xleftarrow{R} X$ - случайный, равномерно распределенный выбор значения x из множества X.

Определение 4.15. ГПСЧ $G : K \to \{0,1\}^n$ называется *криптографически стойким* (cryptographically secure), если $\forall$ оракула $A \in$ PPT при $k \xleftarrow{R} K$, $r \xleftarrow{R} \{0,1\}^n$ величина $Adv[A,G] = |P[A(G(k)) = 1] - P[A(r) = 1]| < \varepsilon(n)$, где $\varepsilon(n)$ - пренебрежимо малая функция. Вероятности вычисляются по случайному выбору k и r и действиям алгоритма A.

Т.е., применяя алгоритмы из класса PPT, при случайном выборе ключа результат вычисления $G(k)$ не удается отличить от результата выбора случайной последовательности r из $\{0,1\}^n$. В этом случае говорят, что последовательность, создаваемая ГПСЧ G, *вычислительно не отличима от случайной* (computationally indistinguishable from random).

Замечание. $|G(k)| = |K| \ll |\{r\}| = 2^n$.

Задача 4.4. Пусть $G : K \to \{0,1\}^n$ такой, что $(G(k))|_1 = 0$ для $2/3$ ключей из K. Пусть оракул A выдает 1, если $x|_1 = 0$, иначе 0. Найдите, чему равно $Adv[A,G] = |P[A(G(k)) = 1] - P[A(r) = 1]|$.

Теорема 4.1. Если ГПСЧ предсказуемый, то он не криптостойкий.

Доказательство. Пусть G - предсказуемый ГПСЧ. $\Rightarrow \exists$ алгоритм A, $\exists i$: $P[A(G(k))|_{1,\dots,i} = G(k)|_{i+1}] > 1/2 + \delta(n)$, где $\delta(n)$ не пренебрежимо малая величина.

Определим оракул B: если $A(X|_{1,\dots,i}) = X|_{i+1}$, выдать 1, иначе 0. Тогда $P(B(r)|_{1,\dots,i} = 1) = 1/2$ и $P(B(G(k))|_{1,\dots,i} = 1) > 1/2 + \delta(n)$, $\Rightarrow Adv[B,G] > \delta(n)$. □

Следствие. Криптостойкий ГПСЧ - непредсказуемый.

Теорема 4.2 (Yao)**.** Пусть $G : K \to \{0,1\}^n$ - ГПСЧ. Пусть $\forall i \in \{1,..,n-1\}$ G не предсказуемый в позиции $i+1$. Тогда G - криптостойкий ГПСЧ.

Доказательство. (*) Опишем основную идею доказательства. Наша цель - показать, что последовательность g из n бит, созданная не предсказуемым ГПСЧ G, вычислительно не отличима от случайной, равномерно распределенной последовательности r той же длины.

Обозначим $g_k r_{n-k}$ - последовательность, состоящая из k первых битов g, а остальные биты взяты из r. Для доказательства достаточно показать, что $g_{i-1} r_{n-i+1}$ вычислительно не отличима от $g_i r_{n-i}$ $\forall i$. Обозначим этот факт как $g_{i-1} r_{n-i+1} \approx g_i r_{n-i}$

$\forall i$. Если мы докажем этот факт, то по индукции $g = g_n \approx g_{n-1}r_1 \approx g_{n-2}r_2 \approx ... \approx r_n = r$.

Пусть $\exists i : g_{i-1}r_{n-i+1} \not\approx g_i r_{n-i}$. Т.е. $\exists$ оракул $A \in$ PPT, который отличает $g_{i-1}r_{n-i+1}$ от $g_i r_{n-i}$ с не пренебрежимо малой вероятностью $2\delta(n)$. Без потери общности, $P[A(g_i r_{n-i}) = 1] = 1/2 + \delta(n)$.

Построим алгоритм B, предсказывающий следующий бит g. В позиции i алгоритм B работает так. Возьмем случайный бит b и вычислим $= A(g_{i-1}br_{n-i})$. Если $c = 1$, B выдаст ответ: $g[i] = b$, причем это верно с вероятностью $1/2 + \delta(n)$. Значения битов r_{n-i} случайные, они не несут никакой дополнительной информации. Т.е. B предсказывает i-й бит g по последовательности g_{i-1} с вероятностью $1/2 + \delta(n)$. $\Rightarrow$ ГПСЧ G не является непредсказуемым. Противоречие. $\square$

Строгое доказательство этой теоремы можно найти в [6], теорема 3.3.7, стр. 119-123.

4.5 Псевдослучайные функции и псевдослучайные перестановки

Пусть $Func(\{0,1\}^n \to \{0,1\}^n)$ - множество всех функций из $\{0,1\}^n$ в $\{0,1\}^n$. Пусть $Perms(\{0,1\}^n \to \{0,1\}^n)$ - множество всех перестановок множества $\{0,1\}^n$.

Определение 4.16. Семейство функций $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ - это семейство *псевдослучайных функций* (ПСФ) по второму аргументу (pseudo random functions), если $F \subset$ PT и $\forall$ оракула $A \in PPT$, $k \xleftarrow{R} K$, $r \xleftarrow{R} M$, величина $Adv(A, F) = |P[A(F(k,\cdot)) = 1] - P[A(r(\cdot)) = 1]| < \varepsilon(n)$, пренебрежимо малой функции.

Определение 4.17. Семейство перестановок $P : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ называется семейством *псевдослучайных перестановок* (ПСП) (pseudo random permutations), если $\forall$ оракула $A \in PPT$ $k \xleftarrow{R} K$, $\pi \xleftarrow{R} Perms(\{0,1\}^n \to \{0,1\}^n)$ величина $|P[A(P(k,\cdot)) = 1] - P[A(\pi(\cdot)) = 1]| < \varepsilon(n)$, пренебрежимо малой функции.

Замечание. ПСП всегда обратима, ПСФ - не обязательно обратима.

Задача 4.5. Пусть $F : K \times X \to \{0,1\}^{128}$ - семейство псевдослучайных функций.

Будет ли следующее семейство функций псевдослучайным:

$$G(k,x) = \begin{cases} 0^{128}, & \text{если } x = 0 \\ F(k,x), & \text{иначе} \end{cases}$$

Утверждение 4.2. Пусть $F: K \times \{0,1\}^n \to \{0,1\}^n$ - семейство ПСФ. Тогда следующая конструкция с использованием ПСФ в режиме счетчика будет криптостойким ГПСЧ:
$G: K \to \{0,1\}^{nt}$, $G(k) = F(k,0)||F(k,1)||...||F(k,t-1)$

Доказательство. Криптостойкость следует из того, что "эффективный" оракул не может отличить $F(k,\cdot)$ от случайной функции. □

Замечание. Для маленького множества X можно легко отличить ПСФ от ПСП. Например, рассмотрим все отображения $X \to X$, $X = \{0,1\}$. Проведем 100 испытаний, выбирая случайные, независимые ключи и вычисляя значения отображения в обеих точках: в 0 и в 1. Если каждый раз отображение - перестановка, то перед нами - семейство ПСП, иначе - ПСФ. Вероятность ошибки можете оценить самостоятельно.

Если множество X большое и его мощность растет экспоненциально, нельзя эффективно отличить ПСП от ПСФ.

Лемма 4.1 (О переключении, PRP/PRF switching lemma)**.** Пусть $X = \{0,1\}^n$. Пусть $f \xleftarrow{R} Func(X,X)$, $\pi \xleftarrow{R} Perms(X,X)$. Тогда $\forall$ алгоритма $A \in$ PPT, верно $|P[A(f) = 1] - P[A(\pi) = 1]| < \varepsilon(n)$.

Без доказательства. Любознательный читатель может ознакомиться со строгим доказательством этого факта в [7].

Лемма о переключении используется для обоснования возможности построения криптостойкого ГПСЧ на основе блочного шифра: если блочный шифр - семейство ПСП, то $\forall$ оракул из класса PPT, т.е. любой "эффективный" оракул, не может отличить его от семейства ПСФ. На основе семейства ПСФ можно построить криптостойкий ГПСЧ.

4.6 Семантическая стойкость шифра с одноразовым ключом

Пусть злоумышленник взаимодействует с системой шифрования как с черным ящиком. Пусть шифр использует одноразовый ключ. Цель злоумышленника

- нарушить семантическую стойкость шифра, т.е. суметь различить шифротексты каких-либо двух сообщений по его выбору.

Опишем формально *атаку с выбором открытого текста на шифр с одноразовым ключом* (chosen plaintext attack for one-time key).

Эксперимент CPA-1 (см. рис. 2):

1. Система выбирает случайное значение бита $b \xleftarrow{R} \{0,1\}$. Оно секретное.
2. Злоумышленник выбирает и отправляет два сообщения $m_0 \neq m_1$ равной длины.
3. Система вычисляет $c = E(k, m_b)$ и отправляет его злоумышленнику.
4. Злоумышленник $A \in$ PPT анализирует c и выдает результат - бит b'.
5. Если $b' = b$, A достиг успеха.

Т.е. злоумышленник проводит атаку с выбранным открытым текстом. Ключ шифра одноразовый, поэтому один запрос, один ответ.

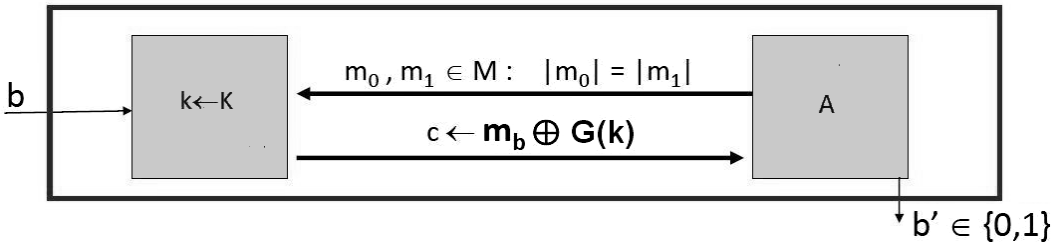

Рис. 2: Эксперимент CPA-1. По материалам [4].

Определение 4.18. Шифр называется *семантически стойким для одноразового ключа* (semantically secure), если в эксперименте CPA-1 $P(A() = b) < \frac{1}{2} + \varepsilon(n)$, где $\varepsilon(n)$ - пренебрежимо малая функция. Вероятность вычисляется по случайным выборам k, b, действиям алгоритма A и шифрования E.

Вероятность угадать правильный ответ равна 1/2. Т.е. в эксперименте CPA-1 вероятность того, что "эффективный" злоумышленник правильно укажет, шифротекст какого из двух сообщений равной длины он получил, отличается от вероятности угадать ответ не более чем на пренебрежимо малую величину.

Эквивалентное определение:

Определение 4.19. Шифр называется *семантически стойким для одноразового ключа*, если в эксперименте CPA-1 преимущество злоумышленника над угадыванием значения b, т.е. величина $Adv_{CPA-1}(A, E) = \left|P(b' = 1|b = 1) - P(b' = 1|b = 0)\right| < \varepsilon(n)$, где $\varepsilon(n)$ - пренебрежимо малая функция.

Задача 4.6. Доказать, что эти два определения эквивалентны.

Пример 4.6. Пусть протокол обмена данными между A и B устроен так, что A отправляет одно из двух фиксированных сообщений. Если шифр семантически стойкий при одноразовом ключе, вероятность того, что злоумышленник, слушающий канал передачи данных, сможет верно определить, какое из двух сообщений было передано, не более $1/2 + \varepsilon$, где ε - пренебрежимо малая величина.

Пример 4.7. Пусть оракул A по шифротексту может точно определить значение некоторой функции $f(m)$ от открытого текста. Построим эксперимент: выберем m_0, m_1 : $f(m_0) = 0, f(m_1) = 1$. По шифротексту можно определить, какое из этих двух сообщений было отправлено:
$Adv_{CPA-1}(A, E) = \big|P(b' = 1|b = 1) - P(b' = 1|b = 0)\big| = |1 - 0| = 1$

Теорема 4.3 (О семантической стойкости поточного шифра)**.** Пусть $G : K \to \{0,1\}^*$ - криптостойкий ГПСЧ. Тогда поточный шифр на основе $G(k)$ семантически стойкий при одноразовом ключе и $\forall$ алгоритма A, пытающегося взломать шифр, $\exists$ алгоритм B атаки на ГПСЧ:
$Adv_{CPA-1}(A, E) \leq 2 \cdot Adv_{PRG}(B, G)$.

Замечание. Из последнего неравенства и того, что ГПСЧ G - криптостойкий, непосредственно следует семантическая стойкость шифра при одноразовом ключе.

Доказательство. Основная идея: выберем произвольные сообщения m_0, m_1 равной длины и оценим возможности злоумышленника отличить друг от друга три ситуации, изображенные на рис. 3.

Рассмотрим эксперимент CPA-1 для произвольного алгоритма A, осуществляющего атаку на поточный шифр. (См. рис. 2)

Обозначим W_q - событие, что в этом эксперименте $b' = 1$ при условии $b = q$. Тогда, формально, $P_A(W_b) = P(b' = 1|b = q)$. $\Rightarrow$ по определению, $Adv_{CPA-1}(A, E) = \big|P_A(W_1) - P_A(W_0)\big|$.

Рассмотрим вторую ситуацию, т.е. попытку алгоритма A узнать значение бита b по принятому шифротексту в эксперименте CPA-1, если шифр - одноразовый блокнот, т.е. $c = m \oplus r$, $r \xleftarrow{R} \{0,1\}^{|m|}$. Обозначим R_q - событие, что ответ алгоритма A $b' = 1$ при условии $b = q$. Тогда вероятность отличить шифротексты сообщений есть $\big|P_A(R_0) - P_A(R_1)\big| = 0$, т.к. это абсолютно стойкий шифр.

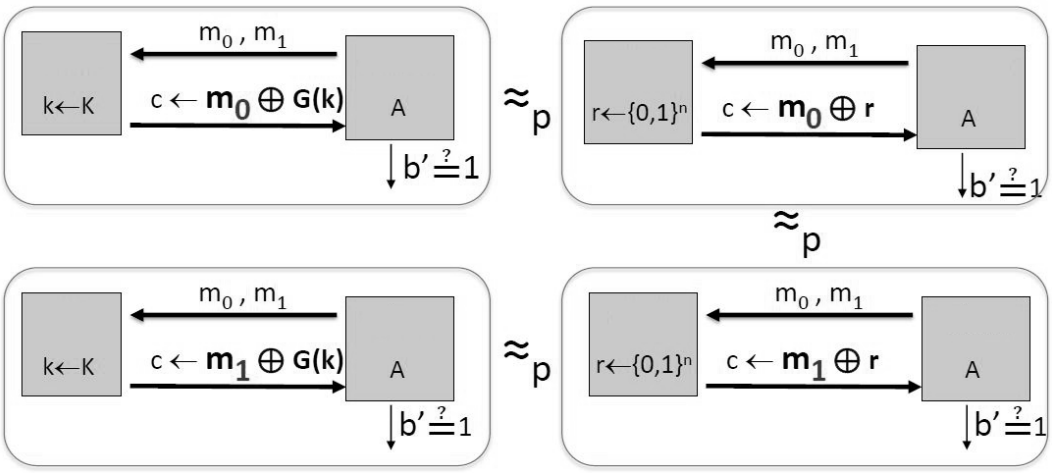

Рис. 3: Основная идея доказательства теоремы о семантической стойкости поточного шифра. По материалам [4].

Построим алгоритм B, пытающийся отличить псевдослучайную последовательность от случайной, на основе оракула A для поточного шифра.

На вход к B подается y - или случайная последовательность r, или значение $G(k)$, в зависимости от значения b. Его цель - угадать b.

B использует A как черный ящик следующим образом. A создает два сообщения равной длины. B отправляет в A шифротекст $c = m_q \oplus y$. A выдает значение b' - оценку того, какому сообщению соответствует шифротекст. B выдает это же b'.

Первая ситуация: B всегда отправляет в A шифротекст $c = m_0 \oplus y$. Поэтому по построению $Adv_{PRG}(B, G) = \left|P(B(G(k)) = 1) - P(B(r) = 1)\right| = \left|P_A(W_0) - P_A(R_0)\right|$. Последнее равенство верно, т.к. по построению $B(G(k)) = 1 \Leftrightarrow$ происходит событие W_0; аналогично $B(r) = 1 \Leftrightarrow R_0$ имеет место. См. рис. 4.

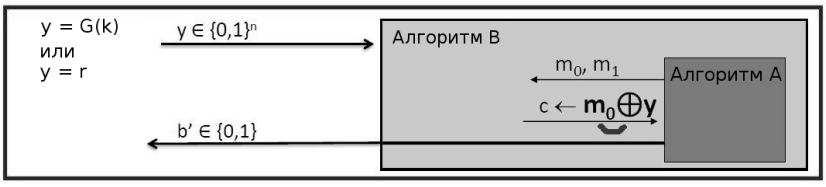

Рис. 4: Алгоритм B атаки на ПСФ, построенный на основе оракула A. По материалам [4].

Третья ситуация: аналогично первой, но для получения соотношения между $P(R_1), P(W_1)$ всегда подаем в A шифротекст $c = m_1 \oplus y$.

$\Rightarrow Adv_{CPA-1}(A, E) = \left|P_A(W_0) - P_A(W_1)\right| = \left|P_A(W_1) - P_A(R_1) + P_A(R_1) - P_A(R_0) +\right.$

$P_A(R_0) - P_A(W_0)\big| \leq \big|P_A(W_0) - P_A(R_0)\big| + \big|P_A(R_0) - P_{OTP}(R_1)\big| + \big|P_A(R_1) - P_A(W_1)\big| =$
$2 \cdot Adv_{PRG}(B, G)$. □

Замечание. Выше был использован прием доказательства, называемый *сведением одной задачи к другой*. Пусть наша цель - доказать, что не существует эффективного алгоритма, решающего задачу X. От противного, предположим, что $A \in$ PPT - алгоритм, решающий X с не пренебрежимо малой вероятностью. Пусть Y - заведомо не решаемая алгоритмами из PPT задача. Построим алгоритм B, решающий задачу Y и использующий алгоритм A описанным выше образом. Это называется *сведением задачи Y к задаче X*. Тогда, если алгоритм A эффективно решает X с не пренебрежимо малой вероятностью, алгоритм $B \in$ PPT - эффективный и решает Y с не пренебрежимо малой вероятностью. Противоречие.

Замечание. Семантическая стойкость шифра с одноразовым ключом к атаке с выбором открытого текста эквивалентна тому, что этот шифр является семейством ПСП. (Любознательный читатель может найти строгое доказательство этого факта в [8], раздел 4.) Для реальных шифров это свойство строго не доказано, но для используемых на практике шифров не известно эффективных алгоритмов, нарушающих их семантическую стойкость.

4.7 Общие недостатки поточных шифров.

1. Детерминированное шифрование. Например, если все сообщения шифруются поточным шифром с одним ключом, псевдослучайная последовательность каждый раз строится с начала и шифротекст зависит только от данных и постоянного ключа, то шифротексты раскрывают некоторую информацию об исходном тексте. Например:

Сообщение 1: "To: Bob. Some text"

Сообщение 2: "To: Eve. Some text"

Шифротексты этих сообщений совпадут, кроме одного фрагмента. Тогда злоумышленник узнает, что различие - только в этом месте.

Вывод: при шифровании данных поточным шифром, необходимо согласовывать новый, независимый ключ для каждой сессии.

2. Поточные шифры не обеспечивают контроль целостности передаваемых данных. Злоумышленник может менять шифротекст c и предсказуемо влиять на

исходный текст m.

Расшифрование: $m = c \oplus G(k)$.

Если злоумышленник заменит шифротекст на c' : $c' = c \oplus p$, тогда получатель расшифрует его как $m' = m \oplus p$.

Получатель не может это обнаружить, если не использует дополнительные средства контроля. Например, если таким образом можно подменить зашифрованный адрес получателя сетевого пакета, то в сетевом протоколе IPSec это приведет к тому, что сервер, расшифровав пакет, отправит расшифрованный текст другому получателю.

4.8 Шифр RC4

Этот шифр изобретен в 1987 г. Роном Ривестом (Ron Rivest). Алгоритм RC4 сначала был секретным, но в 1994 году его анонимно опубликовали в интернете.

Шифр использует нелинейный ГПСЧ для генерации по данному секретному ключу потока значений “гаммы”, которые затем побитово складываются по модулю 2 с открытым текстом. Имеет большую скорость шифрования.

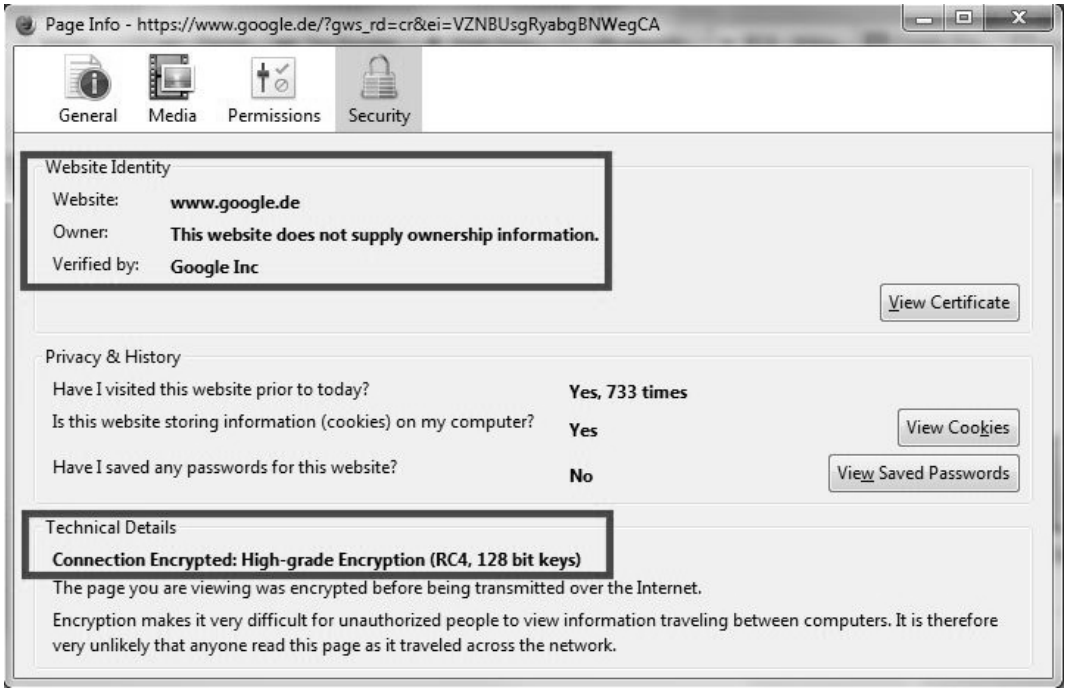

Рис. 5: Пример использования шифра RC4 в 2012 году.

Рассмотрим ГПСЧ, используемый внутри шифра RC4.

4.8.1 Создание начального значения

Начальное значение S (seed) - массив из 256 байтов. Длина секретного ключа - от 1 до 256 байтов, типично - от 5 до 16, т.е. типичная длина ключа - от 40 до 128 бит.

Исходное состояние S - тождественная перестановка. К ней применяется 256 попарных перестановок байтов. Каких именно, зависит от секретного ключа.

Алгоритм вычисления начального значения ГПСЧ:

Пусть `key` - секретный ключ, `keylen` - его длина в байтах. Код создания начального значения на Python:

```
S = range(0, 256)
j = 0
for i in xrange (0, 256):
  j = (j + S[i] + key[i % keylen]) % 256
  S[i], S[j] = S[j], S[i]
```

В результате, даже при случайном, равномерном выборе значений ключа, разные значения S не равновероятны. Найдем причины этого.

Рассмотрим наивный алгоритм перемешивания массива [9]:

```
arlen = len(array)
for i in xrange(0, arlen):   n = rand.Next(arlen)
  array[i], array[n] = array[n], array[i]
```

Для массива длины 3 этот алгоритм дает $3^3 = 27$ возможных путей перемешивания. Но разных перестановок массива - всего $3! = 6$. Результат 600000 испытаний со случайными ключами демонстрирует неравновероятность перестановок массива. Отклонения выходят за рамки статистической погрешности равномерного распределения при данном объеме выборки. См. табл. 2.

Алгоритм перемешивания массива Knuth-Fisher-Yates [9]:

```
arlen = len(array)
for i in xrange(arlen-1, 0, -1):
  int n = rand.Next(i + 1)
  array[i], array[n] = array[n], array[i]
```

Он имеет ровно `arlen!` возможных исходов. При случайном, равномерном генераторе псевдослучайных чисел они равновероятны. Возьмем массив из 6 элемен-

Перестановка	Количество исходов	Доля исходов в %
312	88683	14.78
321	88447	14.74
231	111643	18.61
213	111297	18.55
132	111055	18.51
123	88875	14.81

Таблица 2: Частота результатов перемешивания массива из 3 элементов наивным алгоритмом. По материалам [9].

тов, проведем 600000 испытаний. Перестановки массива алгоритмом Knuth-Fisher-Yates равновероятны в пределах статистической погрешности, в отличие от исходов наивного алгоритма. См. рис. 6.

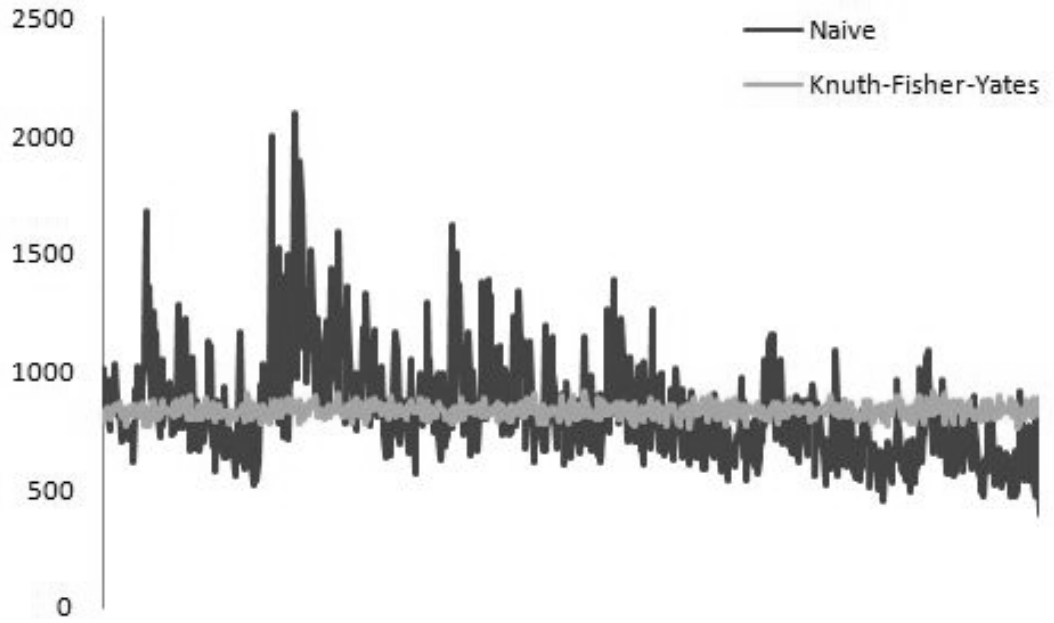

Рис. 6: Сравнение частоты исходов перемешивания массива из 6 элементов. По материалам [9].

Наивный алгоритм перемешивания массива из 6 элементов имеет $6^6 = 46656$ способов перемешивания. Фактически, и по алгоритму Кнута имеется $6! = 720$ разных перестановок массива.

Вывод: в шифре RC4 разные начальные значения имеют разную вероятность из-за особенностей алгоритма перемешивания массива.

4.8.2 ГПСЧ RC4

ГПСЧ использует начальное значение S для инициализации, меняет его при работе: в каждом раунде переставляется одна пара значений внутри S. Секретный ключ уже не используется.

Код алгоритма на Python:

```
i = 0
j = 0
while GeneratingOutput:
  i = (i + 1) % 256
  j = (j + S[i]) % 256
  S[i], S[j] = S[j], S[i]
  K := S[(S[i] + S[j]) % 256]
  Output(K)
```

Набор (S-блок, i, j) имеет $256! \cdot 256^2 \approx 2^{1700}$ состояний. Индекс i обеспечивает изменение каждого байта S-блока через 256 итераций. Индекс j - что для перестановки пары выбираются псевдослучайным образом.

Свойства ГПСЧ RC4:

- нелинейный, проходит статистические тесты при номере байта последовательности > 150
- практически непредсказуемый при при номере байта последовательности > 150
- $P(output[1] = 0) = 2/256$, т.е. второй байт в два раза чаще равен нулю
- начало последовательности не равновероятное

Достоинства шифра RC4:

- быстрый
- не требует вектора инициализации и дополнения до длины блока, поэтому нет уязвимости к атакам: Оракул правильного дополнения до длины блока (padding oracle), BEAST, Lucky13.

Недостатки шифра RC4:

- Общая уязвимость всех поточных шифров: возможна подмена битов шифротекста, приводящая к контролируемым изменениям в открытом тексте. Получатель сообщения не может узнать, что данные были изменены.
- Первые несколько десятков байтов псевдослучайной последовательности не вполне

случайные: не проходят статистические тесты.

- Шифр детерминированный при фиксированном ключе. Как добавить к алгоритму уникальное значение *nonce*, решает пользователь. Из-за плохой реализации использования *nonce* был взломан протокол WEP.

4.8.3 Атака на протокол WEP

В протоколе WEP вектор инициализации IV - это *nonce*, счетчик, длиной 24 бита. Длина сообщения m - от 1 до 2300 байт. Длина ключа k - 104 бита. Используется алгоритм контрольной суммы CRC. Она вычисляется от исходного сообщения перед шифрованием. Шифротекст $c = IV||RC4(IV||k, m||CRC(m))$.
В протоколе WEP IV используется в качестве первых битов ключа:
$c = (m||CRC(m)) \oplus (G(IV||k)[1..|m|])$. Тогда протокол имеет следующие свойства:

- IV повторяется после $2^{24} \approx 16 \cdot 10^6$ сообщений. На некоторых сетевых картах $IV = 0$ после перезагрузки. Результат - многоразовое использование последовательности ПСЧ.
- Не равновероятное начало последовательности ГПСЧ RC4.
- Имеются предсказуемо связанные ключи: $(1||k)$, $(2||k)$, ...
- Существует атака на шифр RC4 на связанных ключах.

Атака на связанных ключах позволяет узнать секретный ключ k. В 2001 году было необходимо 10^6 сетевых фреймов. В 2007 году было достаточно 60000 фреймов, при этом атака занимала 3 минуты.

Протокол WEP можно было бы изменить так, чтобы не было предсказуемо связанных ключей:

- использовать последовательность $G(IV||k)$ со случайным IV для шифрования всего потока данных, не сбрасывать на начало последовательности при новом фрейме;
- или использовать ГПСЧ $G(IV||k)$ для создания ключей $k_1, k_2, ...$, где k_i - псевдослучайный ключ для следующего фрейма: $c = m \oplus G(k_i)$.

Вывод: используйте вместо WEP протокол WPA.

Остальные протоколы, использующие RC4, не имеют связанных ключей и не уязвимы к этой атаке.

Примеры использования шифра RC4:

- WEP (взломан!)

- BitTorrent
- Microsoft Point-to-Point Encryption
- TLS / SSL (один из поддерживаемых шифров)
- SSH (один из поддерживаемых шифров)
- Remote Desktop Protocol
- Kerberos (один из поддерживаемых шифров)
- PDF
- Gpcode.AK - вирус для ОС Windows, созданный в 2008 году. Брал данные на диске в заложники, шифруя их шифрами RC4 и RSA-1024. Расшифрование - за плату.

5 Блочные шифры

Основная цель раздела: описать конструкции блочных шифров и исследовать их стойкость к поиску ключа и семантическую стойкость при одноразовом и многоразовом ключе.

Определение 5.1. Алфавиты ключа, исходного текста и шифротекста *блочного шифра* (block cipher): $\{0,1\}$. При шифровании исходный текст разделяется на группы битов фиксированного размера, называемые *блоками*. Если длина исходного текста не кратна длине блока, перед шифрованием текст дополняется до длины блока. Алгоритм шифрования обрабатывает блоки один за другим. При этом каждый раз создается блок шифротекста того же размера, что и блок исходного текста.

Алгоритм расшифрования работает с блоками шифротекста и использует тот же ключ, что применялся для шифрования. Алгоритм шифрования может быть рандомизированным. Алгоритм расшифрования - всегда детерминированный

Пусть n - длина блока, l - длина ключа. Тогда шифрование одного блока - это отображение $E : \{0,1\}^l \times \{0,1\}^n \to \{0,1\}^n$. Расшифрование блока - это отображение $D : \{0,1\}^l \times \{0,1\}^n \to \{0,1\}^n$.

Эти отображения обладают свойством корректности:
$\forall (k,m) \in \{0,1\}^l \times \{0,1\}^n \; D(k,(E(k,m)) = m$.

Пример 5.1. Шифр DES имеет длину блока $n = 64$ бит, длину ключа $l = 56$ бит. Шифр AES имеет длину блока $n = 128$ бит, варианты длины ключа $l = 128,\ 192,\ 256$ бит.

5.1 Схема Фейстеля

Пусть $F_1, ..., F_n$ - произвольные раундовые функции, Операция $\oplus$ - почленное сложение по модулю два, L - левая половина блока, R - правая, k_i - ключ для данного раунда, который создается по ключу шифра k.

Определение 5.2. *Раунд схемы Фейстеля* с использованием функции $F:\ \{0,1\}^n \to \{0,1\}^n$ - это отображение $H_F:\ \{0,1\}^{2n} \to \{0,1\}^{2n}$: $H_F(L,R) = (R, L \oplus F(R))$

Задача 5.1. Построить обратное преобразование для одного раунда схемы Фейстеля.

Определение 5.3. *Схема Фейстеля* (Feistel network) - это последовательное применение (суперпозиция) нескольких ее раундов. (См. рис. 7.)

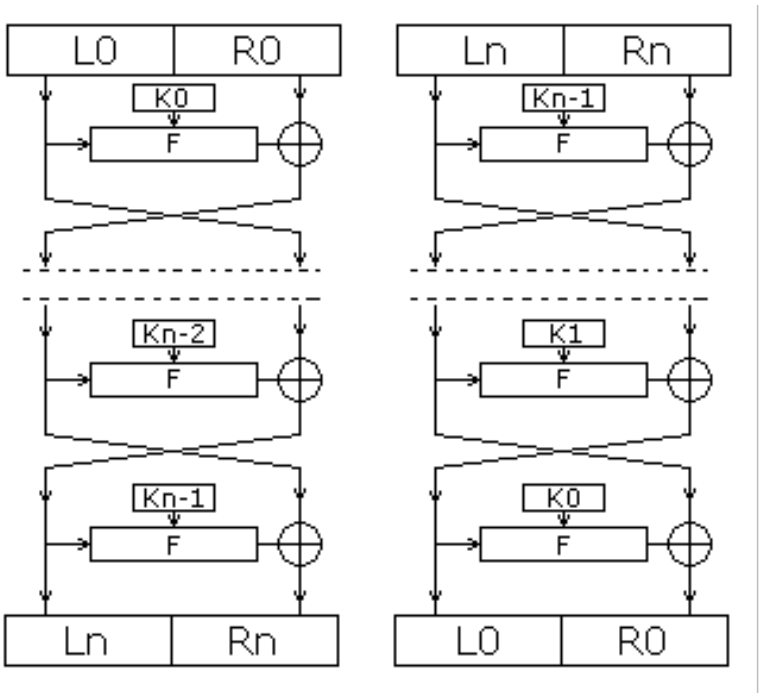

Рис. 7: Схема Фейстеля из n раундов и обратное ей преобразование.

Схема Фейстеля реализует обратимую функцию $\forall$ функции F. Это общий метод построения обратимой функции из нескольких необратимых. Ранее она использовалась для построения многих шифров. Современные шифры ее не используют.

5.2 Шифр DES

5.2.1 Общие сведения

Шифр DES (Data Encryption Standard):

- Разработан фирмой IBM и утвержден правительством США в 1977 году как официальный стандарт шифрования (FIPS-46-3)
- Длина блока - 64 бит. Длина ключа 56 бит. Каждый восьмой бит ключа - бит четности, итого 64 бита.
- Алгоритм использует комбинирование нелинейного (S-box) и линейного (перестановки E, P, IP, $FP = IP^{-1}$) преобразований
- Для шифрования и дешифрования используются одинаковые алгоритм и ключ.
- Состоит из 16 циклов.
- В 1997 году взломан полным перебором ключей (56 бит!). В качестве временного улучшения предложен шифр 3DES.
- В 2001 году утвержден новый стандарт - шифр AES.

Схема работы шифра DES:

- начальная перестановка IP,
- 16 раундов схемы Фейстеля,
- $FP = IP^{-1}$.

Начальная перестановка IP фиксирована, не влияет на криптостойкость алгоритма.

Один раунд алгоритма DES - это один раунд схемы Фейстеля. См. рис. 8.

В последнем раунде шифра DES в схеме Фейстеля левая и правая половины блока не меняются местами. Это сделано для того, чтобы один и тот же алгоритм можно было использовать для зашифрования и расшифрования.

5.2.2 Создание раундовых ключей

Алгоритм создания раундовых ключей k_i устроен так, чтобы каждый раундовый ключ состоял из своего подмножества битов ключа.

Сначала оставим 56 значащих битов ключа. Обозн. k_0.

Далее 16 раз выполняется следующее преобразование, каждый раз создается очередной раундовый ключ:

- 56-битная строка - значение k_{i-1} - делится на 28-битных половины.

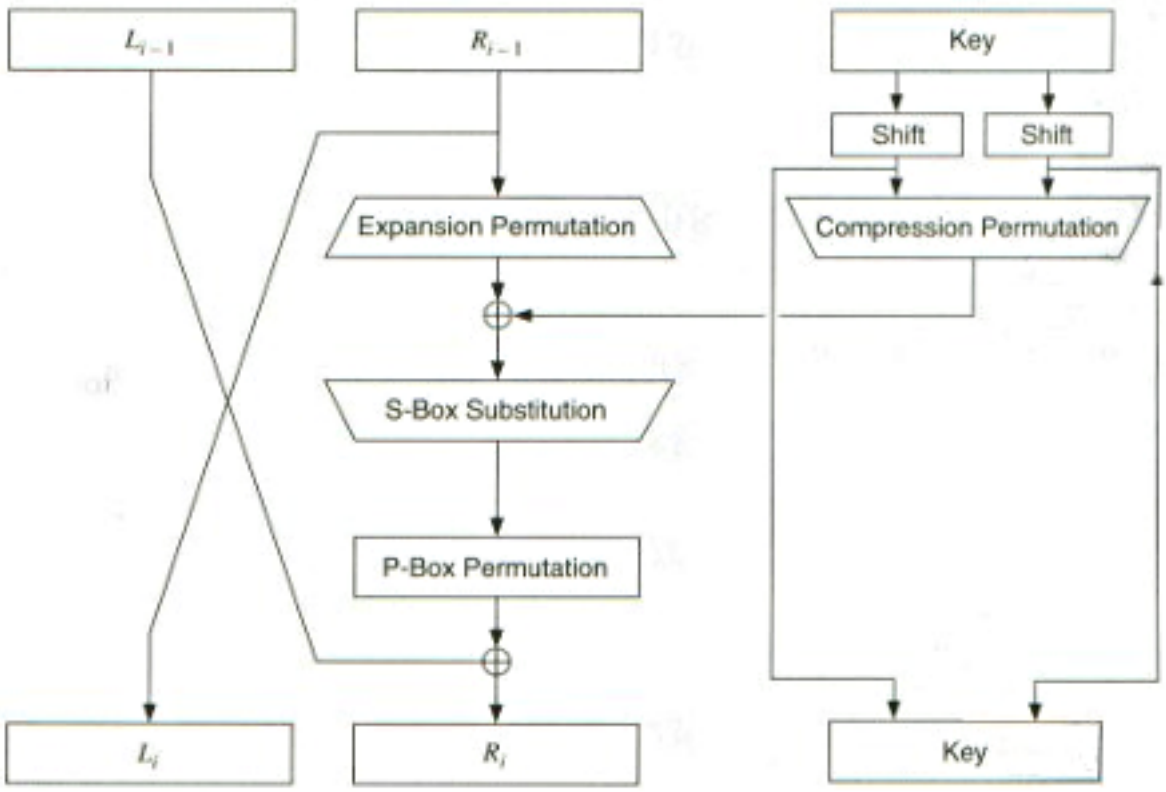

Рис. 8: Один раунд шифра DES.

- Каждая из них независимо сдвигается (с переносом). Величины сдвигов заданы в фиксированной таблице.
- По еще одной фиксированной таблице выбираются 48 бит из 56. Раундовый ключ k_i равен их значению.

При расшифровании алгоритмы обработки блока и создания раундовых ключей те же, но ключи создаются в обратном порядке. Раундовые ключи не зависят от данных, вычисляются заранее.

Пример 5.2. Слабые ключи - это такие ключи, при которых все раундовые ключи равны. Существует 4 слабых ключа:

`01..0101..01`

`01..01FE..FE`

`FE..FE01..01`

`FE..FEFE..FE`

Для них все раундовые функции одинаковы и $\forall m\ E(k,m) = D(k,m)$.

5.2.3 Раундовая функция

Раундовая функция f в схеме Фейстеля состоит из нескольких последовательных преобразований.

1. E-блок - перестановка с расширением. Отображение $\{0,1\}^{32} \to \{0,1\}^{48}$ в соответствии с фиксированной таблицей.

2. Добавление раундового ключа: $x = x \oplus k_i$

3. S-блок - нелинейное преобразование со сжатием. S-блок состоит из 8 подблоков. Каждый осуществляет преобразование $\{0,1\}^6 \to \{0,1\}^4$. Они задаются фиксированными таблицами. Все вместе они задают преобразование $\{0,1\}^{48} \to \{0,1\}^{32}$. Основное свойство S-блоков - отсутствие линейной зависимости результата от входных данных.

4. P-блок - перестановка, $\{0,1\}^{32} \to \{0,1\}^{32}$.

Каждая операция, кроме S-блока, линейная, т.е. может быть описана как $y = B \cdot x$, где B - матрица этой операции.

Блоки P и E подобраны так, что после каждого раунда каждый бит результата зависит от все большего числа бит открытого текста. Т.е. достигается *лавинный эффект*. Его цель - чтобы каждый бит шифротекста зависел от каждого бита открытого текста и ключа.

Пусть за один раунд каждый входной бит влияет на два бита результата, причем они находятся в позициях, которые на следующем раунде влияют на новые позиции, и так далее (идеальный случай). Тогда за i раундов один бит открытого текста повлияет на 2^i битов результата. Если же j-й бит всегда влияет на j-й и $j+1$-й, то за i раундов один бит открытого текста повлияет на $i+1$ битов результата.

Пример 5.3. Пусть значение первого бита открытого текста влияет только на первые 8 битов шифротекста. Пусть $m_1[1] \neq m_2[1],\ m_1[i] = m_2[i]\ \forall i \neq 1$. Тогда верно $c_1[i] = c_2[i]\ \forall i \geq 8\ \forall k \in K$. Т.е. шифр не является псевдослучайной перестановкой.

Подробнее о шифре DES и о блочных шифрах в целом можно прочитать в [10].

5.3 Криптоанализ DES

Проблемы, связанные с шифром DES:

- S-блоки - набор констант без объяснения того, как они получены и для чего нужны.
- алгоритм сертифицирован NSA - Агентством по национальной безопасности США, поэтому возможно наличие лазейки (backdoor), позволяющей NSA легко декодировать сообщения. Наличие лазейки не доказано. Вероятно, ее нет.

После опубликования шифра DES исследовалось,

- почему была выбрана такая длина ключа,
- такое число циклов,
- такой вид S-блоков
- искали способы взлома шифра, кроме полного перебора значений ключа

Обнаружили:

- число циклов - достаточное для распространения лавинного эффекта.
- такой выбор S-блоков обеспечивает гораздо более высокий уровень стойкости шифра, чем (в среднем) обеспечил бы случайный выбор S-блоков.

Модель атаки на шифр DES как ПСП: детерминированное шифрование с многоразовым ключом. Злоумышленник отправляет любые открытые тексты в криптосиситему и получает шифротексты для них. Цель: найти ключ быстрее, чем полным перебором.

Определение 5.4. Если злоумышленник может выбирать открытые тексты один за другим, получать от системы шифрования их шифротексты, [и выбирает следующий открытый текст m, зная все предыдущие пары (c, m),] то он осуществляет *атаку с* [*адаптивным*] *выбором открытого текста* (chosen (adaptive) plaintext attack).

5.3.1 Линейный криптоанализ

Рассмотрим атаку с известным открытым текстом (known plaintext attack). Т.е. злоумышленник знает некоторые пары (m, c), но не может выбирать эти сообщения m и c. Рассматриваем сообщения длиной в один блок.

Пусть существуют такие подмножества индексов I_1, I_2, I_3, что $\forall$ случайных k, m равенство

$$\left(\bigoplus_{i\in I_1\subseteq\{1..64\}} m[i]\right) \oplus \left(\bigoplus_{i\in I_2\subseteq\{1..64\}} c[i]\right) = \left(\bigoplus_{i\in I_3\subseteq\{1..56\}} k[i]\right)$$

выполнено с вероятностью, большей чем от $1/2 + \delta$, где δ - не пренебрежимо малая

величина.

Обозначим этот факт $L_2(m) \oplus L_3(c) = L_1(k)$ - свойство линейности шифра. Обоснуем, что в шифр не должен иметь это свойство.

Пусть верно $P[L_1(k)\ [\oplus 1]\ = L_2(m) + L_3(c)] = 1/2 + \delta,\ \ \delta > 0.$

Алгоритм голосования: для нескольких сообщений m получим их шифротексты c. Пусть $N = |\{m\}|$, $T = |\{m : L_2(m) + L_3(c) = 0\}|$. Если $T > N/2$, принять $L_1(k) = 0$, иначе $L_1(k) = 1$.

Утверждение 5.1. Пусть p_s - вероятность того, что алгоритм голосования дает правильный ответ. $\forall\delta\ \exists N : p_s > 1/2$. Кроме того, p_s стремится к 1 при росте N и δ.

Доказательство. (*) Пусть $L_1(k) = 0$. T - случайная величина (с.в.), сумма N независимых с.в. X_i с распределением Бернулли: $P(X_i = 0) = \frac{1}{2} + \delta$, $P(X_i = 1) = \frac{1}{2} - \delta$.

$E(X_i) = \frac{1}{2} - \delta, \quad D(X_i) = \frac{1}{4} - \delta^2$

$E(T) = N(\frac{1}{2} - \delta)$, $D(T) = N(\frac{1}{4} - \delta^2)$

Алгоритм голосования примет неправильное решение, если $T < N/2$. Вероятность этого $p_f = 1 - p_s \le P\big[|T - E(T)| \ge N\delta\big] \le D(T)/(N\delta)^2 = \frac{1}{N}\left(\frac{1}{4\delta^2} - 1\right)$ (Используется неравенство Чебышева из теории вероятности.) □

Замечание. При $N = 2^{43}$, $\gamma = 2^{-21}$ $\quad p_s > \frac{7}{8}$

Задача 5.2. (*) Вывести формулы для $E(X_i)$, $D(X_i)$, $E(T)$, $D(T)$.

Как меняется вероятность выполнения линейного соотношения при суперпозиции раундов DES?

Пусть $X_i \in \{0, 1\}$ - с.в., $P(X_i = 0) = 1/2 + \varepsilon_i$, с.в. независимы. Легко видеть, что $P(X_1 \oplus X_2 = 0) = 1/2 + 2\varepsilon_1\varepsilon_2$

Лемма 5.1 (Лемма о накоплении (Piling up lemma))**.** Пусть $X_i \in \{0, 1\}$, $P(X_i = 0) = 1/2 + \varepsilon_i$, $i = 1..n$, независимы. Тогда $P(X_1 \oplus ... \oplus X_n = 0) = 1/2 + 2^{n-1}\prod_{i=1}^{n}\varepsilon_i$

Доказательство. По индукции. Докажите самостоятельно. □

Замечание. Значение леммы о накоплении: если $P(X_i = 0) = 1/2 + \varepsilon_i$ и $|\varepsilon_i| < 1/2$, то $|P(X_1 \oplus ... \oplus X_{i-1} = 0) - 1/2| > |P(X_1 \oplus ... \oplus X_i = 0) - 1/2|$, т.е. добавление очередного X_i к сумме с.в. приближает распределение суммы с.в. к равномерному распределению.

Известно, что лучшее линейное соотношение для **14** раундов DES имеет вероятность $\frac{1}{2} + 2^{-21}$. Путем анализа других соотношений и двух оставшихся раундов шифра удается найти 14 бит информации о ключе в виде линейных соотношений. Остальные 42 бита ключа подбираются перебором.

Поэтому существует атака на ПСП DES с параметрами:

- необходимо 2^{43} известных пар $(m, c(m))$
- и еще 2^{43} вычислений DES.
- вероятность успеха 85%

Для взлома полным перебором необходимо 2^{56} вычислений DES. S-блоки - единственная нелинейная часть шифра. Поэтому они должны быть такими, что ни один выходной бит не является "близким" к линейной функции входных битов.

Замечание. Если выбирать S-блоки и P-блоки случайно, шифр DES становится более уязвим к криптоанализу и ключ можно найти существенно быстрее. В некоторых характерных случаях требуется 2^{29} или 2^{37} известных пар $(m, c(m))$ и вычислений шифра DES. Подробное изложение можно найти в [11], а также упоминание в [1], гл. 6.

Задача 5.3. Дан вектор $x \in \{0,1\}^6$ и вектор $y \in \{0,1\}^6$ - перестановка элементов вектора x. Найти матрицу $A \in \mathbb{B}_{6\times 6}$: $Ax = y$. Все операции проводятся по модулю 2.

Задача 5.4. Дан вектор $x \in \{0,1\}^6$ и вектор $y \in \{0,1\}^4$:
$y[1] = x[2] \oplus x[3]$, $y[2] = x[4]$,
$y[3] = x[1] \oplus x[2] \oplus x[5]$, $y[4] = x[6]$.
Найти матрицу $A \in \mathbb{B}_{4\times 6}$: $Ax = y$. Все операции проводятся по модулю 2.

Задача 5.5. В условии предыдущей задачи $y[2] = x[4] \oplus 1$.
Найти A, b : $Ax \oplus b = y$.

Подробнее о линейном криптоанализе можно узнать из [1], гл. 6.

5.3.2 Понятие о дифференциальном криптоанализе

Это атака с адаптивным выбором открытого текста. Злоумышленник выбирает сообщения $m_1, m_2 \Rightarrow \Delta m = m_1 \oplus m_2$. Он получает шифротексты $c_1(m_1), c_2(m_2) \Rightarrow \Delta c = c_1 \oplus c_2$ На основе анализа пар $(\Delta m, \Delta c)$ строится предположение о ключе.

Подробнее о дифференциальном криптоанализе можно узнать из [1], гл. 6.

5.3.3 Атаки по побочным каналам

Пусть злоумышленник получил в свое распоряжение систему шифрования - черный ящик (например, смарт карту).

Первый способ - атака на основе измерений некоторых параметров. Различные операции выполняются за различное время и потребляют разный ток, в зависимости от входных данных. Проводя измерения и их статистический анализ, можно получить полную информацию о ключе. См. рис. 9.

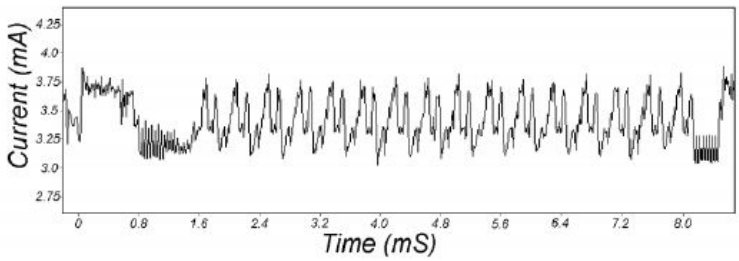

Рис. 9: График силы тока от времени при шифровании DES. По материалам [4].

Второй способ - атака на основе ошибок при вычислениях. Воздействуем электромагнитным излучением в определенные моменты времени на смарт карту, реализующую шифр. Это вызовет ошибки при вычислениях. Это раскрывает некоторую информацию о ключе.

5.3.4 Атака полным перебором значений ключа

Утверждение 5.2. Пусть DES - идеальный шифр, т.е. множество ключей порождает 2^{56} случайных обратимых функций. Тогда $\forall m,\ c \in \{0,1\}^{64}$ с вероятностью более $1 - 1/256 \approx 0.995$ существует не более одного ключа k такого, что $c = DES(k, m)$.

Доказательство. $P[\exists k_1 \neq k :\ DES(k, m) = DES(k_1, m)] \leq$
$\leq \sum_{k_1 \in \{0,1\}^{56}} P[DES(k, m) = DES(k_1, m)] \leq 2^{56} \cdot (1/2^{64}) = 1/256.$ □

DES - не идеальный шифр, поэтому для нахождения ключа перебором используют две пары $(E(k, m_1), m_1), (E(k, m_2), m_2)$.

Применение DES: первые версии протокола Kerberos авторизации в компьютерных сетях были основаны на DES-шифровании. Протокол Kerberos используется и в Unix, и в Windows системах.

5.4 Шифры на основе DES

С ростом мощности компьютеров постепенно уменьшалось время атаки грубой силой (перебором ключей) на шифр DES с ключом в 56 бит:
1977: перебор невозможен
1998: Deep Crack Machine стоимостью 250K$ взломала DES за 3 дня
2006: COPACOBANA (120 FPGAs) стоимостью 10K$ взломала DES за 7 дней

Ключа длиной 56 бит было не достаточно уже в 1998 году. При этом шифр DES еще был стандартом.

Заметим, что E-блок и P-блок шифра DES линейны, т.е. их можно представить в виде матричной операции $y = B \cdot x$. Если бы S-блок тоже был линеен, то один раунд схемы Фейстеля и весь шифр можно было бы представить в виде аффинной операции $c = A \cdot m \oplus b$.

Тогда шифр был бы *алгебраичен*: $\forall k_1, k_2\ \exists k_3 : \forall m\ E(k_1, E(k_2, m)) = E(k_3, m)$. Для шифра DES это соотношение не выполняется. Поэтому последовательное шифрование с разными ключами делает шифр более сложным для взлома полным перебором.

Поэтому, пока разрабатывался новый стандарт шифрования, в качестве временной меры построили более стойкие шифры на базе DES: 3DES, DESX.

5.4.1 Шифр 3DES

Определение 5.5. Пусть (E, D) - шифр DES. Тогда шифр 3DES - это пара алгоритмов (E', D'):
$E'((k_1, k_2, k_3), m) = E(k_1, D(k_2, E(k_3, m)))$,
$D'((k_1, k_2, k_3), c) = D(k_3, E(k_2, D(k_1, c)))$.

Длина ключа $3 \cdot 56 = 168$ бит. 3DES работает в 3 раза медленнее DES. Почему не подошел шифр 2DES? Он уязвим к атаке “встреча посередине”.

Пусть $2DES((k_1, k_2), m) = E(k_1, E(k_2, m))$. Тогда для пары (m, c) найти ключ $k = (k_1, k_2) : c = E(k_1, E(k_2, m))$ эквивалентно задаче найти ключ такой, что $E(k_2, m) = D(k_1, c)$

Реализация атаки: сначала для всех значений k_2 построить таблицу соответствия $z = E(k_2, m) \leftrightarrow k_2$, сортировать ее по z. Размер таблицы $O(2^{56})$. Потом поочередно для каждого значения ключа k_1 проверить, содержится ли значение

$y = D(k_1, m)$ в этой таблице среди значений z. Если это так, то нашли пару (k_1, k_2) : $c = E(k_1, E(k_2, m))$.

Сложность атаки: построение и сортировка таблицы занимает $O(2^{56} \cdot log 2^{56})$, поиск в таблице $O(2^{56} \cdot log 2^{56})$. Итого не более $O(2^{63}) \ll O(2^{112})$.

Задача 5.6. Аналогичная атака на 3DES требует времени $O(2^{118})$, памяти $O(2^{56})$. Почему?

5.4.2 Шифр DESX

Определение 5.6. Пусть (E, D) - шифр DES. Тогда шифр DESX - это пара алгоритмов (E', D'): $E'((k_1, k_2, k_3), m) = k_1 \oplus E(k_2, m \oplus k_3)$,
$D'((k_1, k_2, k_3), c) = D(k_2, c \oplus k_1) \oplus k_3$.

Длина ключа шифра DESX равна $64 + 56 + 64 = 184$ бита, но $\exists$ простая атака, занимающая время $O(2^{64+56}) = O(2^{120})$.

Задача 5.7. Что это за атака?

Задача 5.8. Конструкции $k_1 \oplus E(k_2, m)$ и $E(k_1, m \oplus k_2)$ почти не улучшают стойкость шифра DES. Почему?

5.5 Шифр AES

В 1997-2001 году проходил конкурс на новый стандарт симметричного шифра AES, и его выиграл шифр Rijndael. AES - стандарт шифрования в США с 2001 года (стандарт FIPS 197). Это блочный шифр. Блок - 128 бит. Ключ - 128, 196 или 256 бит. Большая длина ключа обеспечивает большую стойкость, но шифр работает медленнее из-за увеличения количества раундов. Мы рассмотрим только длину ключа в 128 бит. Для этой длины ключа шифрование состоит из 10 раундов.

Шифр AES использует не схему Фейстеля, а сеть замен и перестановок (substitution-permutation network): S-блоки (нелинейные замены) и P-блоки (линейные перестановки).

Раундовые ключи шифра AES имеют размер 128 бит, строятся по ключу шифра. Блок данных представлен в виде матрицы 4x4, где элемент матрицы - 1 байт. Итого 16 байтов, 128 бит. Эту матрицу называют “состояние” (state).

S-блок - это операция SubBytes. Нелинейная операция замены значения байта состояния на другое: $state[i][j] = Sub(state[i][j])$ Программная реализация - таблица или формула (обращение элемента конечного кольца полиномов над конечным полем).

P-блок - это комбинация операций ShiftRows и MixColumns. ShiftRows - сдвиг строк. i-я строка матрицы состояния сдвигается (с переносом) на $i-1$ позицию вправо. Ряды обрабатываются независимо. MixColumns - линейное обратимое преобразование столбца. Эти преобразования могут быть описаны как операции над элементами конечного кольца полиномов над конечным полем.

Шифрование (см. рис. 10):

1. Состояние складывается с первым раундовым ключом.
2. 10 раз повторить раунд шифрования, состоящий из S-блока, P-блока и сложения с очередным раундовым ключом. В последнем раунде P-блок незначительно изменен.

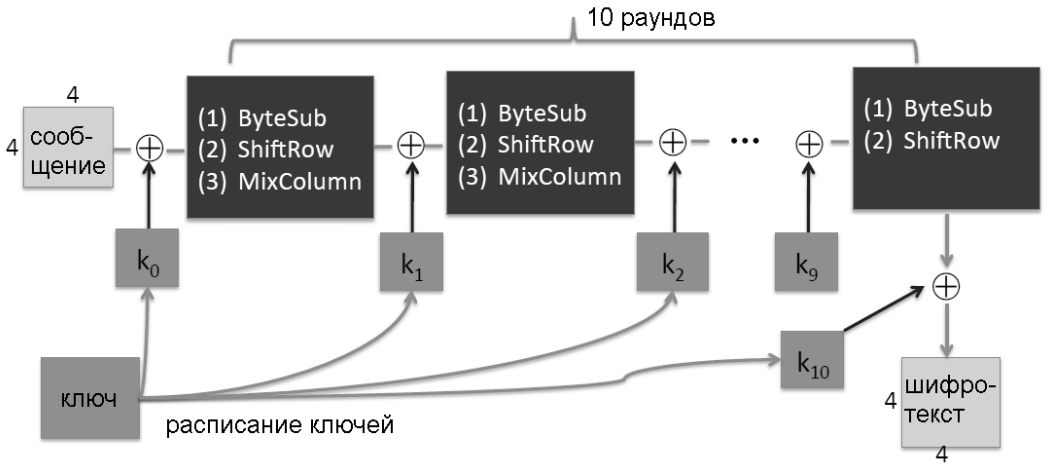

Рис. 10: Шифр AES. По материалам [4].

Расшифрование аналогично шифрованию. Раундовые ключи используются в обратном порядке.

Шифр AES устойчив ко всем известным атакам.

Поиск ключа шифра всего в 4 раза быстрее, чем полный перебор возможных значений. Т.е. для ключа длиной 128 бит он потребует 2^{126} операций.

Существует атака на 4 связанных ключах: если злоумышленнику известны 2^{99} пар (сообщение, шифротекст) для 4 связанных ключей, один из связанных ключей можно найти за 2^{99} операций. Но в реальных сценариях использования шифра AES

нет связанных ключей, поэтому эта атака не применима.

Подробнее о шифре AES можно прочитать в [10].

5.6 Семантическая стойкость шифра при одноразовом ключе

До сих пор рассматривали шифрование одного блока. Перейдем к шифрованию сообщения произвольной длины.

Модель атаки на шифр:

- одноразовый случайный ключ
- сообщение имеет длину более одного блока
- злоумышленник может отправлять в систему сообщения по своему выбору и получать их шифротексты, но ключ шифра каждый раз новый, случайный.
- цель злоумышленника: по шифротексту получить информацию о сообщении

Т.е. это - *атака с выбором открытого текста* (chosen plaintext attack) на сообщение длиной более одного блока при одноразовом ключе.

Эксперимент CPA-1 для одноразового ключа (повтор):

1. Система выбирает случайное значение бита $b \xleftarrow{R} \{0,1\}$. Оно секретное.
2. Злоумышленник выбирает натуральное число n и отправляет два сообщения $m_0 \neq m_1$ длины n.
3. Система вычисляет $c = E(k, m_b)$ и отправляет его злоумышленнику.
4. Злоумышленник $A \in \mathrm{PPT}$ анализирует c и выдает результат - бит b'.
5. Если $b' = b$, A достиг успеха.

Для успеха в этом эксперименте злоумышленнику достаточно суметь различить шифротексты для каких-либо двух сообщений равной длины по своему выбору.

Замечание. Семантическая стойкость шифра с одноразовым ключом к атаке с выбором открытого текста эквивалентна тому, что этот шифр является семейством ПСП. (Любознательный читатель может найти строгое доказательство этого факта в [8], раздел 4.) Для реальных шифров это свойство строго не доказано, но для используемых на практике шифров не известно эффективных алгоритмов, нарушающих их семантическую стойкость.

Определение 5.7. Режим работы блочного шифра, когда при шифровании к каждому блоку исходного текста применяется одна и та же детерминированная перестановка $E(k, \cdot)$, называется *электронная кодовая книга* (Electronic codebook, ECB).

Задача 5.9. Доказать, что режим ECB не семантически стойкий. Указание: использовать сообщение длиной два блока, где эти блоки одинаковые.

Обозначим $z[i]$ - i-й блок сообщения или шифротекста z.

Определение 5.8. Пусть $F : K \times \{0,1\}^n \to \{0,1\}^n$ - семейство ПСФ. Тогда в режиме *детерминированного счетчика* (deterministic counter, DetCTR) i-й блок шифротекста вычисляется по формуле $E_{DetCTR}(k,m)[i] = m[i] \oplus F(k,i)$.

Когда блочный шифр используется как поточный (т.е. шифрование - поблочная операция $\oplus$), это называется режимом *гаммирования*.

Теорема 5.1. Пусть $F : K \times \{0,1\}^n \to \{0,1\}^n$ - семейство ПСФ. Тогда $\forall L > 0$ при одноразовом ключе E_{DetCTR} - семантически стойкий шифр: $K \times \{0,1\}^{nL} \to \{0,1\}^{nL}$. В частности, $\forall$ алгоритма $A \in$ PPT, реализующего атаку на E_{DetCTR}, $\exists$ алгоритм $B \in$ PPT, осуществляющий атаку на семейство ПСФ F: $Adv_{CPA-1}[A, E_{DetCTR}] \leq 2 \cdot Adv_{CPA-1}[B, F]$

Доказательство. Полностью аналогично доказательству теоремы о семантической стойкости поточного шифра. □

5.7 Семантическая стойкость шифра при многоразовом ключе

Если один ключ используется для шифрования несколько раз, злоумышленник видит несколько шифротекстов, зашифрованных одним и тем же ключом.

Модель атаки на шифр:

- Злоумышленник может получать шифротексты для любых сообщений, при этом ключ шифра один и тот же.
- Цель злоумышленника: нарушить семантическую стойкость шифра, т.е. суметь отличить шифротексты для каких-то двух сообщений равной длины по его выбору.

Это *атака с выбором открытого текста* при многоразовом ключе.

Эксперимент CPA-N описывает эту атаку формально. (См. рис. 11.)

Пусть (E, D) - шифр.

1. Система выбирает значение бита $b \xleftarrow{R} \{0,1\}$ и ключ $k \xleftarrow{R} K$, они секретные, не изменяются.

2. $q \in \mathbb{N}$ - число запросов злоумышленника. $q = O(poly(len(m)))$. При каждом запросе он отправляет два сообщения $m_{i,0}, m_{i,1}$ равной длины n_i. В разных запросах длина сообщений может отличаться. Сообщения злоумышленника могут повторяться.
3. Каждый раз система вычисляет $c_i = E(k, m_{i,b})$ и отправляет его злоумышленнику.
4. После q запросов злоумышленник выдает b' - гипотезу о b.

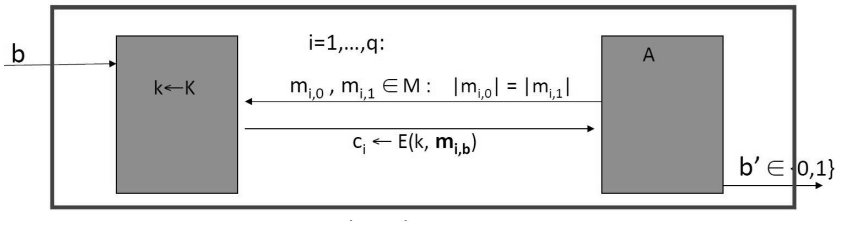

Рис. 11: Эксперимент CPA-N. По материалам [4].

Определение 5.9. шифр $C = (E, D)$ называется *семантически стойким к атаке с выбранным открытым текстом при многоразовом ключе*, если в эксперименте CPA-N $\forall A \in$ PPT

$Adv_{CPA-N}(A, C) = \left|P(b' = 1|b = 0) - P(b' = 1|b = 1)\right| < \varepsilon(n)$

Если шифрование детерминированное, простая атака нарушает семантическую стойкость шифра:

1. Злоумышленник отправляет две копии произвольного сообщения m_0. Получает $c_0 = E(k, m_0)$.
2. Злоумышленник отправляет два сообщения m_0, m_1 равной длины, $m_0 \neq m_1$. Получает шифротекст $c = E(k, m_b)$.
3. Если $c_0 = c$, то $b' = 0$, иначе $b' = 1$.

Рассмотрим способы построения семантически стойкого шифра.

1. Алгоритм шифрования может быть рандомизированным.

$E_r(k, m) = c_1$, повторно $E_r(k, m) = c_2$, с высокой вероятностью $c_1 \neq c_2$. Алгоритм расшифрования D - детерминированный. Очевидно, длина шифротекста должна быть больше длины сообщения.

Пример 5.4. Пусть $F : K \times R \to M$ - семейство ПСФ. R - достаточно большое множество чисел. Шифр $E(k, m) = [r \xleftarrow{R} R, \text{ вернуть } (r, F(k, r) \oplus m)]$

Задача 5.10. Почему этот шифр семантически стойкий к атаке с выбором открытого текста? Указание: ПСФ $F(k, \cdot)$ вычислительно не отличима от случайной функции $g(\cdot)$.

2. Можно использовать *nonce* - уникальное значение, "n used once". Тогда пара $(nonce, m)$ никогда не повторится. Варианты реализации *nonce*:

- *nonce* - случайное число, выбираемое из большого множества, так что оно никогда не повторится с большой вероятностью.
- *nonce* - счетчик, тогда нужна энергонезависимая память для сохранения значения; можно не передавать *nonce*

Эксперимент nCPA-N описывает атаку на шифр с многоразовым ключом, использующий *nonce*. Он отличается от эксперимента CPA-N тем, что каждый раз злоумышленник выбирает и отправляет значение *nonce* n_i. Все *nonce* n_i попарно различны, хотя их и выбирает злоумышленник. См. рис. 12.

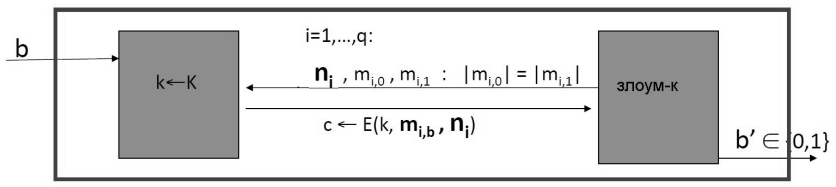

Рис. 12: Эксперимент nCPA-N. По материалам [4].

Определение 5.10. шифр $C = (E, D)$, использующий *nonce*, *семантически стойкий к атаке с выбранным открытым текстом*, если в эксперименте nCPA-N $\forall A \in$ PPT $Adv_{nCPA-N}[A, C] = \left|P(b' = 1|b = 0) - P(b' = 1|b = 1)\right| < \varepsilon(n)$.

Задача 5.11. Пусть $f : K \times R \to M$ - ПСФ. R - конечный диапазон целых чисел. Пусть $E(k, m) = [++r,\ \text{вернуть}\ (r, f(k, r) \oplus m)]$ - шифр с многоразовым ключом. При каком количестве сообщений этот шифр семантически стойкий относительно атаки с выбранным открытым текстом?

5.8 Режим сцепления блоков (CBC)

Используем блочный шифр для передачи сообщений произвольной длины при многоразовом ключе. При многоразовом ключе семантическую стойкость к атаке с выбором открытого текста обеспечивают режимы:

- CBC - режим сцепления блоков (cipher block chaining) (изобретен в 1981 году)
- CFB (1981)
- OFB (1981)
- RandCTR - режим рандомизированного счетчика (randomized counter) (2001)
- и другие

Рассмотрим режим CBC - режим сцепления блоков. Он описан в стандарте NIST: FIPS 81.

Пусть длина сообщения кратна длине блока. Выберем вектор инициализации IV (initialization vector) той же длины, что и блок шифра, разобьем сообщение на блоки.

Шифрование (см. рис. 13): $c[1] = E(k, m[1] \oplus IV)$, $c[i] = E(k, m[i] \oplus c[i-1])$. Шифровать блоки можно только последовательно.

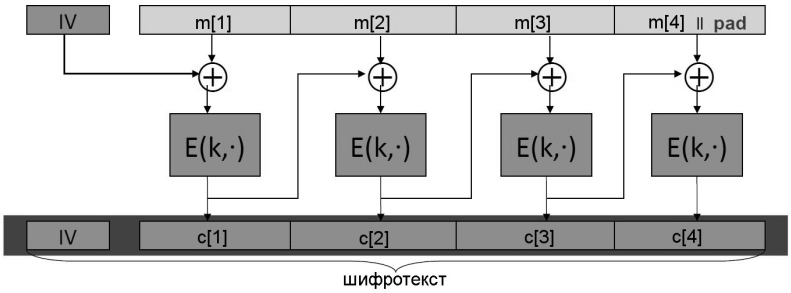

Рис. 13: Шифрование в режиме сцепления блоков. По материалам [4].

Расшифрование (см. рис. 14): $m[1] = D(k, c[1]) \oplus IV)$, $m[i] = D(k, c[i]) \oplus c[i-1])$. Расшифровывать блоки можно параллельно.

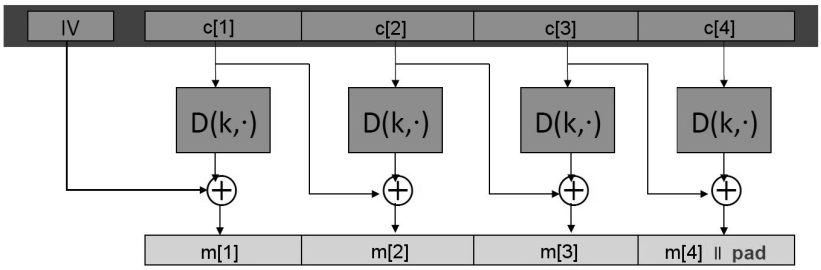

Рис. 14: Расшифрование в режиме сцепления блоков. По материалам [4].

Требования к выбору вектора инициализации IV при режиме CBC:

- не секретный
- уникальный (зачем?)

- не предсказуемый: нельзя предсказать IV для следующего сообщения по предыдущим сообщениям. Если можно предсказать IV, существует атака с выбранным открытым текстом - атака BEAST на протокол SSL/TLS 1.0. Она будет рассмотрена ниже.

Способы создания IV в режиме CBC:

- IV - последний блок предыдущего сообщения (протоколы SSL 2.0, 3.0, TLS 1.0). Тогда шифр не является семантически стойким и уязвим к атаке BEAST.
- IV - значение криптостойкого ГПСЧ. Это медленно.
- IV $= E(k_1, nonce)$, $nonce$ - счетчик, не повторяется для текущего ключа k. $k_1 \neq k$.

Задача 5.12. Почему в случае, если $IV = E(k_1, nonce)$, требуем $k_1 \neq k$?
Указание: какой недостаток, если $k_1 = k$ и $nonce$ не секретный?

Задача 5.13. Почему IV должен быть уникальным?

Задача 5.14 (Атака BEAST с выбранным открытым текстом на режим CBC с предсказуемым IV.)**.** Цель злоумышленника - нарушить семантическую стойкость шифра при атаке с выбранным открытым текстом. Предложить способ ("эксперимент CPA-N"), как злоумышленник пользуется тем, что он может предсказать значение IV для следующего сообщения. Указание: первая пара сообщений: $m_0 = m_1 = 0$; вторая пара: $m_0 = IV_2 \oplus IV_1, m_1 \neq m_0$.

Теорема 5.2 (О стойкости режима CBC)**.** $\forall L > 0$, если E - ПСП: $K \times X \to X$, то шифр E_{CBC} на основе этой ПСП - семантически стойкий к атаке с выбранным открытым текстом над (K, X^L, X^{L+1}).

Доказательство. (*) Докажите самостоятельно аналогично доказательству теоремы о семантической стойкости поточного щифра. □

Дополнение текста до длины блока перед шифрованием в режиме CBC (см. рис. 13) должно позволять однозначно удалить его при расшифровании. Используется следующий алгоритм:

- Если длина сообщения не кратна длине блока, последний блок сообщения дополняем справа до длины блока. Если нужно дополнить n байтов, значение каждого из этих байтов равно n.

- Если длина сообщения кратна длине блока, добавим еще один блок, целиком состоящий из нулей. См. рис. 13.

Проблемы использования режима CBC:

- возможна уязвимость в виде оракула правильного окончания блока: по возвращаемому значению или времени выполнения злоумышленник может получить информацию о том, правильное ли дополнение до длины блока в расшифрованном сообщении. Пример: атака padding oracle attack на протокол SSL (2002 год).
- предсказуемый IV $\Rightarrow$ шифр уязвим к атаке BEAST (атака на протокол SSL/TLS (2004, 2011 годы))
- атака Lucky 13 на протокол TLS (2013 год) - развитие padding oracle attack.

5.9 Режим рандомизированного счетчика (RandCTR)

Режим рандомизированного счетчика (RandCTR) изобретен в 2001 году, описан в стандарте NIST: SP800-38A.

Пусть $F: K \times \{0,1\}^n \to \{0,1\}^n$ - криптостойкая ПСФ для некоторого n.

Шифрование в режиме RandCTR идет следующим образом (см. рис. 15): выберем случайный IV длиной n и используем сумму IV и номера блока как параметр для вычисления "гаммы" $F(k, IV + i)$. Шифрование: $c[i] = m[i] \oplus F(k, IV + i)$.

Рис. 15: Шифрование в режиме RandCTR, первый вариант. По материалам [4].

Шифрование и расшифрование блоков может идти параллельно. В режиме RandCTR не нужно дополнять сообщение до длины блока: последний блок может быть короче остальных.

Улучшенный вариант этого способа: использование *nonce* и счетчика блоков сообщения. Длины *nonce* и счетчика равны половине длины блока. Чтобы пара (k, IV) не повторялась для разных сообщений, $IV = (nonce||i)$, где i - номер блока текущего сообщения. Шифрование: $c[i] = m[i] \oplus F(k, IV||i)$. См. рис. 16.

Рис. 16: Шифрование в режиме RandCTR, второй вариант. По материалам [4].

Теорема 5.3 (О стойкости режима RandCTR)**.** $\forall L > 0$, если F - криптостойкое семейство ПСФ: $K \times X \to X$, то шифр $E_{RandCTR}$ семантически стойкий к атаке с выбранным открытым текстом над (K, X^L, X^{L+1}).

Доказательство. (*) Докажите самостоятельно аналогично доказательству теоремы о семантической стойкости поточного шифра. □

Сравнение режимов CBC и RandCTR:

- используем ПСП или ПСФ?
- возможны ли параллельные вычисления при шифровании, расшифровании?
- нужно ли дополнение открытого текста до длины блока?
- какую длину имеет шифротекст сообщения длиной 1 байт?

Итак, мы исследовали стойкость шифра к атаке с выбранным открытым текстом при однократном и многократном использовании ключа.

- При одноразовом ключе стойкость имеют поточные шифры и DetCTR режим блочных шифров.
- При многоразовом ключе стойкость имеют режимы блочных шифров CBC, RandCTR. (А также некоторые другие режимы блочных и поточных шифров.)

5.10 Сравнение скорости работы шифров

Скорости работы поточных и блочных шифров (AMD Opteron 2.2 GHz, Linux, библиотека crypto++ 5.6.0) указаны в табл. 3.

6 Код аутентичности сообщения

Основная цель раздела: рассмотреть способы контроля целостности открытой информации.

Шифр	Размер блока/ключа, бит	Скорость, МБ/с
Поточные шифры		
RC4 (1987)	- / от 8 до 4096	126
Salsa20 (2005)	- / 256	643
Блочные шифры		
DES (1977)	64 / 56	39
3DES (1997)	64 / 168	13
AES-128 (2000)	128 / 128	109

Таблица 3: Скорость работы шифров. По материалам [4].

6.1 Криптостойкий код аутентичности сообщения

Определение 6.1. Код аутентичности (message authentication code) $I = (S, V)$ - это пара алгоритмов $S, V \in$ PT: $S : K \times M \to T$, $V : K \times M \times T \to \{0, 1\}$. Алгоритм S вычисляет значение (метку) кода аутентичности для сообщения, алгоритм V его проверяет. Они используют общий секретный ключ.

Значение кода аутентичности передается по открытому каналу вместе с сообщением, поэтому контрольная сумма, вычисляемая только по сообщению, не годится. Злоумышленник может изменить сообщение и вычислить новую верную контрольную сумму.

Модель атаки на код аутентичности: атака с выбранным открытым текстом.

Пусть злоумышленник может отправлять любые сообщения $m_1, ..., m_q$ в криптосистему получать их метки $t_i = S(k, m_i)$. Цель злоумышленника: создать новую верную пару $(m, t) \not\in \{(m_1, t_1), ..., (m_q, t_q)\} : V(k, m, t) = 1$, т.е. создать новую верную метку t для нового сообщения m или новую верную метку $t'_i \neq t_i$ для какого-то сообщения m_i.

Определение 6.2. Код аутентичности над K, M, T называется *криптостойким* (cryptographically secure), если $\forall$ злоумышленника $A \in$ PPT вероятность создать новую верную метку пренебрежимо мала.
Т.е. $Adv_{MAC}[A, I] = P($A создал новую верную метку$) < \varepsilon(min(log(|K|), log(|T|)))$, где $\varepsilon(n)$ - пренебрежимо малая функция.

На практике требуем $\forall A:$ время работы $A < N$ $Adv_{MAC}[A, I] = P($A создал новую верную метку$) < \varepsilon$ - пренебрежимо малой величины.

Задача 6.1. пусть (S, V) - код аутентичности и всем известно сообщение m_0: для половины значений ключей $k \in K$ злоумышленник может найти $m_1 \neq m_0 : S(k, m_0) = S(k, m_1)$. Будет ли этот код аутентичности криптостойким?

Задача 6.2. пусть (S,V) - код аутентичности и $T = \{0,1\}^5$. Будет ли он криптостойким? Указание: чему равна вероятность угадать метку?

Рассмотрим реализацию кода аутентичности с помощью ПСФ.

Теорема 6.1. Пусть $F : K \times X \to Y$ - ПСФ и $1/|Y| < \varepsilon$ - пренебрежимо малой величины. Тогда $I_F : S(k, m) = F(k, m)$ - криптостойкий код аутентичности. В частности, $\forall$ атаки $A \in$ PPT на I_F, $\exists$ атака $B \in$ PPT на ПСФ F :

$$Adv_{MAC}[A, I_F] \leq Adv_{proof}[B, F] + 1/|Y|$$

Доказательство. Злоумышленник выбирает произвольное сообщение. Он может или просто угадать метку для него, или использовать атаку на ПСФ. Поэтому вероятность успешной атаки на код аутентичности = P(успешная атака на ПСФ) + P(угадали метку) − P(оба эти события). □

Лемма 6.1. Пусть $F : K \times X \to Y$ - ПСФ. Тогда $F_t(k, m) = F(k, m)[1, ..., t]$ - ПСФ $\forall 1 \leq t \leq n$. Т.е., если обрежем криптостойкую ПСФ, снова получим криптостойкую ПСФ.

Задача 6.3. Доказать лемму (от противного).

Задача 6.4. Пусть $F : K \times X \to Y$ - ПСФ. При каком условии на t код аутентичности, построенный по обрезанной стойкой ПСФ $F_t(k, m) = F(k, m)[1, ..., t]$, будет криптостойким? Указание: использовать лемму.

Пример 6.1. для сообщения длиной один блок можно использовать шифр как код аутентичности: $t = E(k, m)$.

6.2 Построение кода аутентичности для длинного сообщения по ПСФ для короткого сообщения

ECBC-MAC - зашифрованный код аутентичности сообщения, использующий ПСФ в режиме сцепления блоков. См. рис. 17.

Определение 6.3. Пусть $F : K \times X \to Y$ - криптостойкая ПСФ. Тогда ECBC-MAC $F_{ECBC} : K^2 \times X^{\leq L} \to X$ определяется следующим образом. Пусть сообщение m имеет длину q блоков.

$g(m[1]) = F(k, m[1])$;

$\forall 1 < i \leq q \; g(m[1, ..., i]) = F(k, g(m[1, ..., i-1]) \oplus m[i])$.

$F_{ECBC}(k, k_1, m) = F(k_1, g(m))$

Если используем ECBC-MAC на основе блочного шифра, необходимо дополнять сообщения до длины блока. Дополним сообщение последовательностью 10..0, а если длина сообщения кратна длине блока, добавим целый блок 10..0.

Замечание. Если дополнять сообщение нулями, то $MAC(k, m) = MAC(k, m||0..0)$.

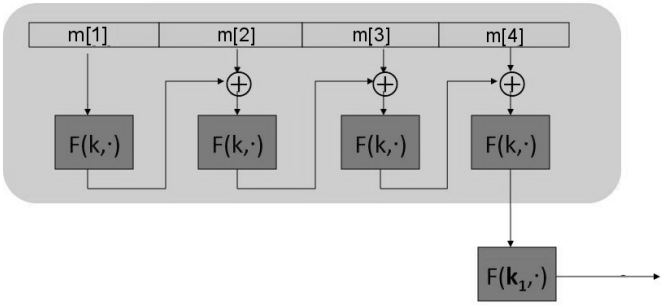

Рис. 17: Код аутентичности ECBC-MAC. По материалам [4].

NMAC - вложенный MAC.

Определение 6.4. Пусть $F : K \times X \to Y$ - криптостойкая ПСФ. NMAC $F_{NMAC} : K^2 \times X^{\leq L} \to K$ определяется следующим образом. Пусть сообщение m имеет длину q блоков.

$g(m[1]) = F(k, m[1])$;

$\forall 1 < i \leq q \; g(m[1, ..., i]) = F(g(m[1, ..., i-1], m[i]). \; F_{NMAC}(k, k_1, m) = F(k_1, g(m)||pad)$.

NMAC медленнее ECBC-MAC, т.к. NMAC на каждом шаге требует вычисления нового расписания ключей у шифра, реализующего F.

Обозначим raw CBC MAC - ECBC MAC без последнего шага (шифрования) и cascade NMAC - NMAC без последнего шага (дополнения и шифрования).

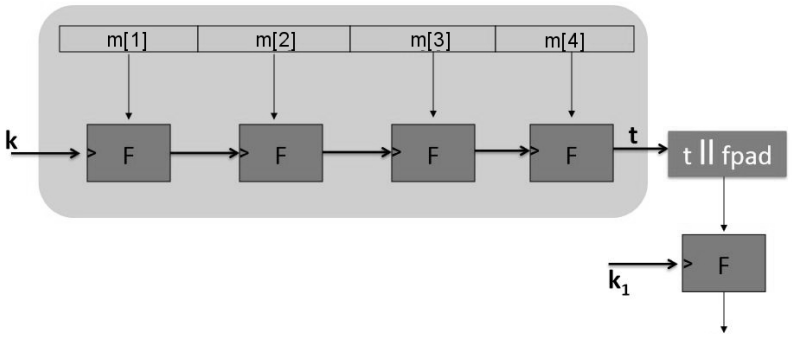

Рис. 18: Код аутентичности NMAC. По материалам [4].

Замечание (Свойство "продолжения" у raw CBC-MAC и cascade NMAC)**.** Заметим, что для raw CBC-MAC и cascade NMAC $\forall x, y, w$ верно
$g(k, x) = g(k, y) \Rightarrow g(k, x||w) = g(k, y||w)$.

Задача 6.5. Обосновать уязвимости кодов аутентичности raw CBC MAC и cascade NMAC. Указания:

- Пусть для cascade NMAC известны значения $g(k, m)$ и w - продолжение сообщения m. Получить $g(k, m||w)$ при неизвестном k.
- Пусть для raw CBC-MAC известна пара (m, t), длина m - один блок. Предложить значение второго блока u и сообщение вида $m||u$, для которого можно найти верную метку, и вычислить верную метку для этого сообщения.

Утверждение 6.1. Для любого фиксированного числа блоков в сообщении, ECBC-MAC и NMAC, построенные на основе ПСФ, являются криптостойкими кодами аутентичности.

Без доказательства. Строгое доказательство этого факта весьма нетривиально. Оно изложено в [1], гл. 4.

На основе ECBC-MAC построен стандарт NIST - код аутентичности CMAC.

По ключу k построим два ключа k_1, k_2 и модифицируем схему CBC-MAC: если длина сообщения кратна длине блока, перед обработкой последнего блока сложить его с k_2, иначе дополнить до длины блока и сложить с k_1.

Определение 6.5. Пусть $F : K \times X \to Y$ - криптостойкая ПСФ. Тогда CMAC $F_{CMAC} : K \times X^{\leq L} \to X$ определяется следующим образом. Пусть сообщение m имеет длину q блоков.

$g(m[1]) = F(k, m[1])$;

$\forall 1 < i \leq q \; g(m[1, ..., i]) = F(k, g(m[1, ..., i-1]) \oplus m[i])$.

Если последний блок неполный, дополнить его и сложить с ключом k_1:

$F_{CMAC}(k, k_1, k_2, m) = F(k, g(g(m[1, ..., q-1]) \oplus m[q] \oplus k_1))$.

Если длина сообщения кратна длине блока, сложить последний блок с ключом k_2:

$F_{CMAC}(k, k_1, k_2, m) = F(k, g(g(m[1, ..., q-1]) \oplus m[q] \oplus k_2))$.

Достоинства CMAC:

- не нужно шифрование результата, т.к. он складывается по модулю 2 с не известным злоумышленнику ключом k_i
- вычисление происходит быстрее, чем ECBC-MAC: не нужно вычислять новое расписание ключей для блочного шифра, выступающего в роли ПСФ на последнем шаге
- не нужен лишний блок, если длина сообщения кратна длине блока.

6.3 Атака на код аутентичности на основе парадокса дня рождения

Пусть Y - множество меток кода аутентичности. Атака заключается в следующем:

- получить $1.2 \cdot \sqrt{|Y|}$ пар (m_i, t_i) для случайных сообщений.
- по парадоксу дня рождения, с вероятностью более $1/2$ в полученных метках $\exists$ коллизия $t_u = t_v$ при $m_u \neq m_v$.

Сложность атаки - $O(\sqrt{|Y|})$ по памяти, $O(\sqrt{|Y|} log(\sqrt{|Y|}))$ по времени. Если код аутентичности имеет свойство продолжения, по одной коллизии можно построить любое количество новых коллизий: $(m_v||w, t)$, $(m_u||w, t)$ $\forall w$.

7 Криптографические хэш функции

Основная цель раздела: построить и исследовать криптографические хэш функции.

7.1 Определения

Определение 7.1. Детерминированная функция $H : M \to T$ - *хэш функция* (hash function), если $|M| \gg |T|$.

Определение 7.2. *Коллизия* (collision) для функции H - это пара $m_0, m_1 \in M$: $H(m_0) = H(m_1)$.

Определение 7.3. Функция H называется *устойчивой к коллизиям* collision resistant), если $\forall$ явно описанного алгоритма $A \in$ PPT,
$Adv_{CR}[A, H] = P(A$ дает коллизию для H$) < \varepsilon(log(|T|))$, где $\varepsilon(\cdot)$ - пренебрежимо малая функция.

На практике требуем $\forall A : time(A) < N$ $Adv_{CR}[A, H] = P(A$ дает коллизию для $H) < \varepsilon$ - пренебрежимо малой величины.

Следствие (Стойкость к поиску второго прообраза)**.** Если H устойчива к коллизиям, по паре (m, t) трудно найти $m_1 : t = H(m_1)$.

Следствие. Если H устойчива к коллизиям, трудно найти два сообщения m, m_1 : $H(m) = H(m_1)$

Определение 7.4. Функция H называется *стойкой к поиску прообраза* (first preimage resistant), если $\forall t \in T$ трудно найти прообраз m, т.е. $m : t = H(m)$. А именно, $\forall A \in$ PPT $Adv_{PI}[A, H] = P(A(t) = x : H(x) = t) < \varepsilon(log(|T|))$. Вероятность вычисляется по действиям алгоритма A.

Следствие. Если H - стойкая к поиску прообраза функция, то H - односторонняя функция.

Определение 7.5. Функция H называется *криптографической хэш функцией* (secure hash function), если она стойкая к коллизиям и к поиску прообраза.

Утверждение 7.1. $\forall$ хэш функции $H : M \to \{0,1\}^n$ за время $O(2^{n/2})$ можно найти коллизию с вероятностью более $1/2$.

Доказательство. Применить парадокс дней рождения. □

7.2 Хэш функция на основе функции сжатия

Конструкция Меркля-Дамгарда (Merkle, Damgard) описывает способ построения хэш функций на основе функции сжатия. Разобьем сообщение на блоки и при необходимости дополним до длины блока. По функции сжатия $h : T \times X \to T$, $|X| \gg |T|$ построим хэш функцию $H : X^{\leq L} \to T$, как показано на рис. 19.

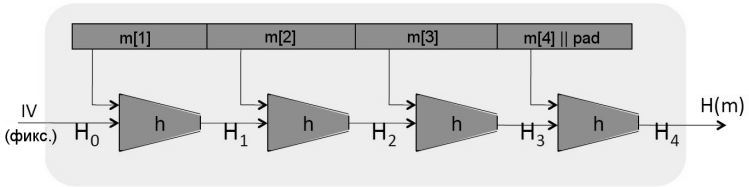

Рис. 19: Хэш функция на основе функции сжатия. По материалам [4].

pad - дополнение до длины блока вида $(10..0||msg\_len)$, где длина поля msg_len - 64 бита. IV - фиксированный вектор инициализации.

Теорема 7.1. Пусть хэш функция H - это конструкция Меркля-Дамгарда, использующая функцию сжатия h. Тогда, если h устойчива к коллизиям, то H - тоже.

Т.е., чтобы построить устойчивую к коллизиям хэш функцию, достаточно построить устойчивую к коллизиям функцию сжатия.

Доказательство. Покажем, что коллизия $H \Rightarrow$ коллизия h.

Пусть $H(M) = H(M'),\ M \neq M'$. Построим коллизию для h. В конструкции Меркля-Дамгарда для M и M' имеем две последовательности, см. табл. 4.

$IV = H_0$	H_1	...	H_t	$h(H_t,\ M_t\|\|PB) = H_{t+1} = H(M)$
$IV' = H'_0$	H'_1	...	H'_r	$h(H'_r,\ M'_r\|\|PB') = H'_{r+1} = H(M')$

Таблица 4: Вычисления в конструкции Меркля-Дамгарда.

Справа $h(H_t,\ M_t||PB) = H_{t+1} = H'_{r+1} = h(H'_r,\ M'_r||PB')$. Если $H_t \neq H'_r$ или $M_t \neq M'_r$ или $PB \neq PB'$, то имеем коллизию для h на последнем шаге. Иначе имеем $H_t = H'_r\ \&\ M_t = M'_r\ \&\ PB = PB', \Rightarrow t = r$.

Следовательно, на предыдущем шаге величины были равны:
$h(H_{t-1},\ M_{t-1}) = H_t = H'_t = h(H'_{t-1},\ M'_{t-1})$. Снова, если $H_{t-1} \neq H'_{t-1}$ или $M_{t-1} \neq M'_{t-1}$, то имеем коллизию для h. $\Rightarrow$ на предыдущем шаге величины были равны: $H_{t-1} = H'_{t-1}\ \&\ M_{t-1} = M'_{t-1}$. Пройдем по всем шагам, тогда либо мы встретим коллизию для

h, либо $\forall i\ M_i = M'_i$ (т.к., по определению, $IV = IV'$), т.е. $M = M'$ - противоречие с тем, что $M \neq M'$. □

7.3 Функция сжатия на основе блочного шифра

Пусть (E, D) - блочный шифр, $E,\ D : K \times \{0,1\}^n \to \{0,1\}^n$.

Если функция сжатия $h(H, m) = E(m, H)$, то она имеет следующие коллизии $(H, m), (H', m')$: возьмем случайные одноблочные сообщения m, m'. Пусть $H' = D(m', E(m, H)) \Rightarrow\ \ h(H', m') = E(m', H') = h(H, m)$. $\Rightarrow$ она не подходит для конструкции Меркля-Дамгарда.

Определение 7.6. Функция сжатия Дэвиса-Мейера (Davies, Meyer) - это $h(H, m) = E(m, H) \oplus H$.

Определение 7.7. Шифр (E, D) над множествами $K, M, C = M$ называется *идеальным*, если это $|K|$ различных перестановок множества M.

Теорема 7.2. Пусть h - функция Дэвиса-Мейера на основе идеального шифра E. Тогда функция h устойчива к коллизиям. В частности, если ее множество значений имеет мощность 2^l и алгоритм поиска коллизии делает не более $q < 2^{l/2}$ вычислений E или D, то он находит коллизию h с вероятностью не более $q^2/2^l$.

Т.е., чтобы наверняка найти коллизию, необходимо совершить $2^{l/2}$ запросов. (Сравните этот факт с парадоксом дня рождения.)

Без доказательства. Подробное доказательство можно найти в [1], гл. 6.

Пример 7.1. Хэш функция SHA-256:

- Конструкция Меркля-Дамгарда
- Функция сжатия Дэвиса-Мейера на основе блочного шифра SHACAL-2

Итак, криптографическая хэш функция может быть построена следующим образом. Возьмем блочный шифр, который при случайном, равномерном выборе ключа является семейством ПСП для одного блока. На его основе построим функцию сжатия Дэвиса-Мейера. Ее используем в конструкции Меркля-Дамгарда.

Некоторым недостатком такого подхода является тот факт, что устойчивость к коллизиям функции сжатия Дэвиса-Мейера доказана для идеального шифра, но ни один из реальных шифров не является идеальным.

7.4 HMAC

7.4.1 Описание HMAC

Хэш-MAC, HMAC - код аутентичности на основе хэш функции, устойчивой к коллизиям.

Построим код аутентичности на основе хэш функции SHA-256.

Примитивный способ: $S(k,m) = H(k||m)$, или использовать k вместо IV, и т.п. Он имеет следующий недостаток: пусть известно $S(k,m)$. $\Rightarrow$ можно вычислить метку для продолжения сообщения $S(k,m||PB||w) = h(w, S(k,m))\ \forall w$.

Определение 7.8. HMAC $S(k,m) = H(k \oplus opad || H(k \oplus ipad||m))$. Здесь хэш функция H - это SHA256. *opad*, *ipad* - разные, открытые, одноблочные константы.

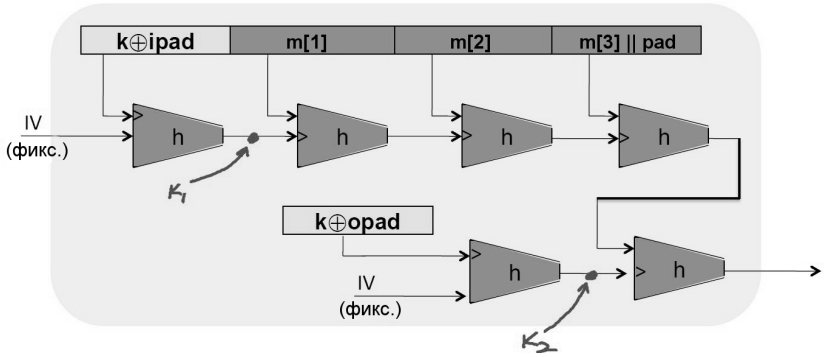

Рис. 20: Код аутентичности HMAC. По материалам [4].

Конструкция похожа на NMAC: $HMAC_k(m) = NMAC_{(k_1,k_2)}(m)$, но ключи k_1, k_2 не являются независимыми.

Теорема 7.3. HMAC - стойкий код аутентичности, если хэш функция H устойчива к коллизиям и функция сжатия h, на основе которой строится H, является криптостойким кодом аутентичности для сообщений фиксированной длины.

Замечание. Для SHA-256 условия теоремы выполняются.

Стойкость HMAC вытекает из следующей теоремы.

Определение 7.9. Пусть $I = (S,V) : K \times M \to T$ - код аутентичности короткого сообщения фиксированной длины, например, шифр AES для нескольких блоков. Пусть $H : M^{big} \to M$ - хэш-функция. Определим $I^{big} = (S^{big}, V^{big}) : K \times M^{big} \to T$:
$S^{big}(k,m) = S(k, H(m))$,
$V^{big}(k,m,t) = V(k, H(m), t)$.

Теорема 7.4 (Достаточность)**.** Если I - криптостойкий код аутентичности и H^{big} - устойчивая к коллизиям хэш функция, то I^{big} - криптостойкий код аутентичности.

Доказательство. От противного. Пусть $y = H(m),\ t = S(k, y) = S(k, H(m))$. Пусть I^{big} не криптостойкий, т.е. эффективный алгоритм A может найти коллизию.
Возможны два варианта:

1. A может найти $m' \neq m :\ H(m') = H(m)$. Противоречие: H устойчива к коллизиям.
2. A не может найти коллизию H, но может найти новую верную пару (m, t) для $I^{big} \Leftrightarrow A$ может найти новую верную пару (y, t) для I. Противоречие: код аутентичности I - криптостойкий.

□

Теорема 7.5 (Необходимость)**.** Если I^{big} - криптостойкий код аутентичности, то H - устойчивая к коллизиям хэш функция.

Доказательство. От противного. Пусть злоумышленник может найти коллизию $m_0 \neq m_1 : H(m_0) = H(m_1)$. Тогда код аутентичности S^{big} не криптостойкий:

1. Злоумышленник получает $t = S(k, m_0)$.
2. Злоумышленник предъявляет пару (m_1, t).

□

Срок жизни ключа: HMAC криптостойкий при $q^2/|T| < \varepsilon$ - пренебрежимо малой величины.

Пример использования: HMAC используется в TLS/SSL и во многих других сетевых протоколах.

Более подробное описание кодов аутентичности на основе криптографических хэш функций и обоснование их стойкости можно найти в [12].

7.4.2 Атаки на HMAC по побочным каналам

Рассмотрим атаку по времени выполнения сравнения. Тривиальная реализация:

```
def Verify(key, msg, tag):
  return tag == HMAC(key, msg)
```

Размер метки - 256 бит. Сравнение ленивое, побайтное. Тогда злоумышленник за 256 запросов с разными значениями первого байта метки по времени выполнения

сравнения может узнать его правильное значение. Потом - значение второго байта и т.д.

Если напишем код, который принудительно сравнивает все байты, компилятор может его оптимизировать Поэтому, решение:

```
def Verify(key, msg, tag):
  mac = HMAC(key,msg)
  return HMAC(key, mac) == HMAC(key, tag)
```

Злоумышленник не знает, какие значения сравниваются.

8 Заверенное шифрование

Основная цель раздела: описать совместное использование шифрования и кода аутентичности сообщения для обеспечения одновременно конфиденциальности и неизменности данных.

8.1 Общие сведения

Семантическая стойкость шифра против атаки с выбором открытого текста обеспечивает тайну сообщения от того, кто подслушивает открытый канал передачи данных. Но если злоумышленник может менять шифротекст, получатель этого не заметит. Криптостойкий код аутентичности сообщения обеспечивает контроль целостности передаваемого сообщения. Рассмотрим, как их совместить, чтобы обеспечить одновременно конфиденциальность и неизменность информации.

Пример использования: пусть отправлено конфиденциальное платежное поручение фиксированного формата. Необходимо, чтобы в нем не поменяли получателя или сумму.

Пусть шифр (E,D) - это набор из двух алгоритмов $E : K \times M[\times N] \to C$ и $D : K \times C[\times N] \to M \bigcup \perp$, где N - множество *nonce*, используется опционально; символ $\perp \notin M$ обозначает отказ от приема данного шифротекста.

Определение 8.1. Шифр (E, D) *обеспечивает целостность шифротекста* (ciphertext integrity), если $\forall A \in$ PPT
$Adv_{CI}[A, E, D] = P(A$ создаст новый шифротекст, который примет система $) < \varepsilon(n)$,
где $\varepsilon(n)$ - пренебрежимо малая функция. n - параметр шифра.

Определение 8.2. Шифр (E, D) - *шифр с заверенным шифротекстом* (authenticated encryption), если он семантически стойкий к атакам с выбором открытого текста и обеспечивает целостность шифротекста.

Замечание. Ни один из рассмотренных ранее шифров не является шифром с заверенным шифротекстом, потому что расшифрование всегда дает какое-то сообщение.

Рассмотрим стойкость заверенного шифрования к атаке с выбранным шифротекстом. Модель атаки: злоумышленник может получать от системы шифротексты для любых сообщений по своему выбору, и он может отправлять новые шифротексты (т.е. отличающиеся от тех, которые он получил из системы) в систему шифрования, стремясь получить расшифрованные сообщения. Цель злоумышленника: нарушить семантическую стойкость шифра, т.е. различить шифротексты двух сообщений.

Эксперимент CCA - атака с выбором шифротекста (chosen ciphertext attack), см. рис. 21:

1. система выбирает $b \xleftarrow{R} \{0,1\}$ и ключ $k \xleftarrow{R} K$. Значение b фиксировано. Ключ многоразовый, он фиксирован.
2. Злоумышленник делает по очереди q запросов любого вида:
   или отправляет два сообщения m_{i0}, m_{i1} одинаковой длины и получает шифротекст $c_i = E(k, m_{ib})$,
   или отправляет шифротекст c, отличный от всех ранее полученных шифротекстов c_i, и получает результат его расшифрования.
3. Злоумышленник выдает значение b', пытается угадать значение b.

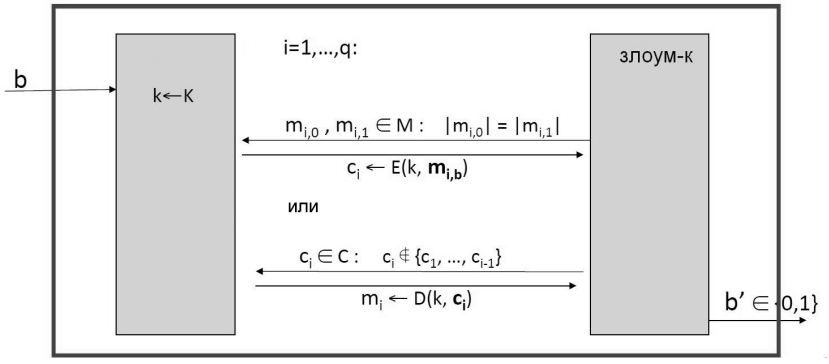

Рис. 21: Эксперимент CCA. По материалам [4].

Определение 8.3. Шифр (E, D) *стойкий к атаке с выбранным шифротекстом*, если в эксперименте CCA

$Adv_{CCA}[A, E, D] = |P(b' = 1|b = 1) - P(b' = 1|b = 0)| < \varepsilon(n)$

Пример 8.1. Режим CBC со случайным IV не стойкий к атаке с выбором шифротекста. Опишем успешную атаку. Сначала отправляются сообщения $m_0 \neq m_1$ длиной 1 блок (n бит). Пусть получен шифротекст $(IV, c[1])$. На втором шаге отправляется шифротекст $c = (IV \oplus 1^n, c[1])$. Это новый шифротекст. Он будет расшифрован как $m_b \oplus 1 \Rightarrow$ злоумышленник узнает значение b. См. рис. 22.

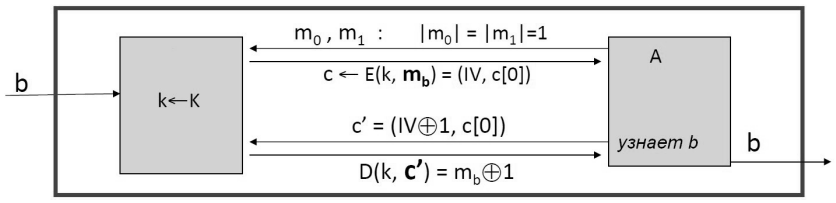

Рис. 22: Атака с выбранным шифротекстом на шифр в режиме CBC. По материалам [4].

Теорема 8.1. Пусть (E, D) - шифр с заверенным шифротекстом. Тогда шифр (E, D) стойкий к атаке с выбором шифротекста, включающей в себя q запросов, причем $\forall$ алгоритма $A \in$ PPT $\exists$ алгоритмы $B_1,\ B_2 \in$ PPT :
$Adv_{CCA}[A, E, D] \leq 2q \cdot Adv_{CI}[B_1, E, D] + Adv_{CPA}[B_2, E, D]$

Доказательство. Идея доказательства: сравнить возможности злоумышленника в четырех ситуациях (см. рис. 23). Вероятность того, что в первой ситуации созданный злоумышленником шифротекст будет принят, пренебрежимо мала, т.к. шифр обеспечивает целостность шифротекста. Поэтому первые две ситуации (почти) равновероятны. Для сравнения вероятности второй и третьей ситуаций используется стойкость к атаке с выбранным открытым текстом, из которой следует, что шифротексты любых двух сообщений злоумышленник (почти) не может различить; третьей и четвертой - снова целостность шифротекста. □

Вывод: шифр с заверенным шифротекстом (authenticated symmetric encryption) гарантирует защиту от злоумышленника, который не только слушает канал передачи данных, но может менять передаваемые сообщения и создавать новые. Если шифротекст был расшифрован (не был отвергнут), принимающая сторона уверена, что автор сообщения знал ключ k.

Шифр с заверенным шифротекстом не гарантирует:

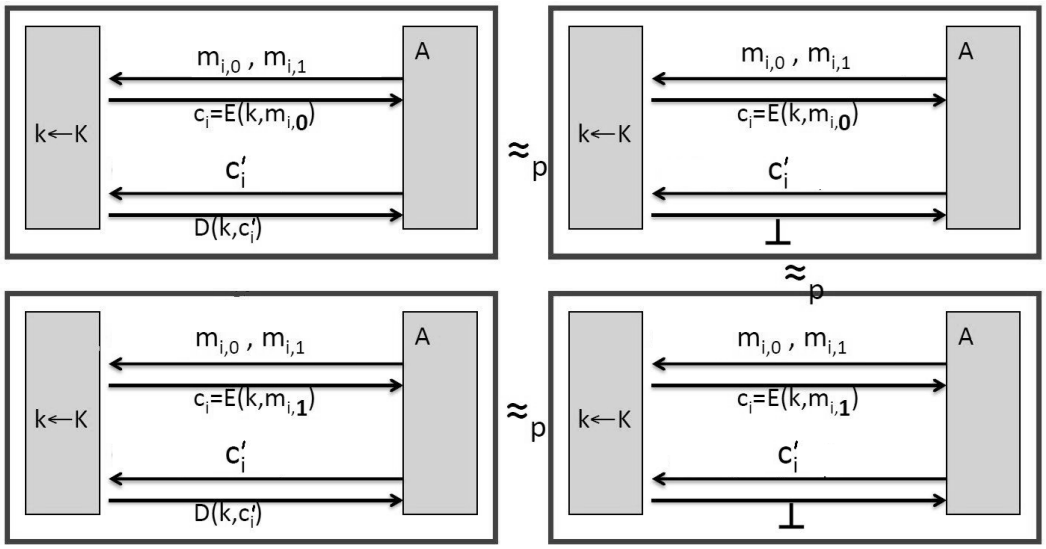

Рис. 23: Идея доказательства стойкости шифра к атаке с выбранным шифротекстом. По материалам [4].

- стойкость к атаке воспроизведения. Поэтому многие протоколы используют метки времени и требуют точное системное время, например, сеть i2p. Или используется счетчик сообщений от начала сессии.
- стойкость к атакам по вторичным каналам (например, по времени выполнения).
- можно сфабриковать сообщение от контрагента, т.к. у двух сторон общий секретный ключ.

8.2 Варианты сочетания шифрования и кода аутентичности

Пусть k_E, k_I - ключи для шифрования и кода аутентичности. Комбинировать шифр и код аутентичности можно по-разному:

1) MtE, MAC-then-Encrypt. $m \to m||S(k_I, m) \to E(k_E, S(k_I, m)||m)$. Пример использования: TLS/SSL.

2) EtM, Encrypt-then-MAC. $m \to E(k_E, m) \to E(k_E, m)||S(k_I,\ E(k_E, m))$. Пример использования: IPSec.

3) EaM, Encrypt-and-MAC. $m \to E(k_E, m) \to E(k_E, m)||S(k_I, m)$. Пример использования: SSH.

Теорема 8.2. Если шифр обеспечивает стойкость к атаке с открытым текстом, а код аутентичности - целостность шифротекста, режим EtM всегда обеспечивает заверенное шифрование, независимо от реализаций шифра и кода аутентичности.

Доказательство. Идея доказательства. По определению, криптостойкий код аутентичности обеспечивает целостность шифротекста.

Шифр (E, D) сам по себе стойкий к атаке с выбранным открытым текстом. К шифротексту добавлено значение метки MAC. MAC от шифротекста вычисляется с независимым случайным ключом, поэтому значение $t = S(k_i, E(k_E, m))$ не сообщает злоумышленнику никакой новой информации об открытом тексте при атаке с выбранным открытым текстом. $\Rightarrow$ шифр (E', D'), где $E'(k, m) = (E(k_E, m), t)$, $D'(k, (c, t)) = D(k, c)$ стойкий к атаке с выбором открытого текста.

Таким образом, выполнены оба условия из определения заверенного шифрования. □

Остальные два варианта могут быть уязвимы к атакам с выбором шифротекста. Например, в случае EaM криптографическая хэш функция может раскрывать информацию о тексте, например, возможно, что $\forall m \exists f : H(m)[0] = f(m)$. На практике, детали реализации протоколов SSH (режим EaM) и TLS (режим MtE) обеспечивают заверенное шифрование. Подробное исследование стойкости разных комбинаций шифрования и кода аутентичности можно найти в работе [13].

Стандартные режимы заверенного шифрования:

GCM: зашифровать сообщение в режиме RandCTR и заверить шифротекст кодом аутентичности GMAC.

EAX: зашифровать сообщение в режиме RandCTR и заверить шифротекст кодом аутентичности CMAC.

CCM: заверить сообщение кодом аутентичности CBC-MAC, затем зашифровать вместе сообщение и метку в режиме RandCTR.

Пример реализации: библиотека OpenSSL на языке C++ содержит реализацию AES128-GCM-SHA256 и другие варианты GCM режима.

Определение 8.4. Если шифруется одна часть сообщения, а другая часть - нет, и кодом аутентичности при этом заверяется всё сообщение, то такой режим называется *заверенное шифрование с присоединенными данными* (Authenticated Encryption with Associated Data, AEAD).

Пример 8.2. Открытые данные - сетевые заголовки, зашифрованные данные - содержимое сетевого пакета.

8.3 Скорость работы режимов заверенного шифрования, шифров и хэш функций

Данные для AMD Opteron, 2.2 GHz, Linux, библиотеки Crypto++ 5.6 указаны в табл. 5.

Алгоритм	Скорость, МБ/с
AES/GCM	108
AES/CCM	61
AES/EAX	61
AES-CTR	139
AES-CBC	109
HMAC(SHA1)	147
SHA-256	111

Таблица 5: Скорости работы шифров и криптографических хэш функций. По материалам [4].

8.4 Протокол TLS/SSL после согласования ключей

Обмен информацией по протоколу TLS состоит из двух этапов:

1. Согласование симметричных ключей клиента и сервера (рассмотрим позже). В результате обе стороны имеют по 4 ключа: для симметричного шифра и для кода аутентичности, разные ключи для передачи информации в одну сторону и в другую: k^E_{cs}, k^I_{cs} и k^E_{sc}, k^I_{sc}
2. Обмен информацией, зашифрованной симметричным шифром.

Каждая сторона хранит состояние - два 64-битных счетчика сообщений, переданных от клиента к серверу и обратно: ctr_{cs}, ctr_{sc}. Счетчики сбрасываются в 0 при инициализации сессии, увеличиваются на 1 при каждом сообщении. Они обеспечивают защиту от атаки воспроизведения.

Используется режим заверенного шифрования MtE, например, код аутентификации HMAC-SHA256 и шифр CBC-AES-128.

Пусть S - вычисление кода аутентичности, E - шифрование. Сетевой пакет создается клиентом следующим образом:

1. *header* - заголовок пакета. Содержит тип, версию протокола, длину сообщения (без заголовка). Значение счетчика не посылается.
2. данные сжимаются: $payload = compress(data)$
3. $tag = S(k_I, (++ctr_{cs}||header||payload))$
4. $pad =$ дополнение до длины блока сообщения $(payload||tag)$
5. данные шифруются: $E(IV, k_E, payload||tag||pad)$ с новым случайным IV.
6. в начало сообщения приписывается открытый *header*.

Сообщение: $header||E(payload||tag||pad)$. Т.к. *tag* зависит от *header*, это режим AEAD.

Когда сервер принял сетевой пакет, он обрабатывает его следующим образом:

1. Расшифровать сообщение с ключом k_{cs}.
2. Проверить *pad*. Если не правильный, ответить сообщением bad_record_mac
3. Проверить *tag* для сообщения $(++ctr_{cs}||header||payload)$. Если он не правильный, ответить bad_record_mac.
4. Если всё верно, расшифровать, разархивировать *payload*.

8.5 Атака на шифр при оракуле правильного окончания блока в режиме сцепления блоков

Реализация алгоритма обработки принятого сообщения в протоколе TLS 1.0 содержит оракул правильного окончания блока. Алгоритм обработки сообщения следующий:

- Расшифровать сообщение с ключом k_{cs}.
- Проверить *pad*. Если не правильный, ответить сообщением decryption_failed.
- Проверить *tag* для $(++ctr_{cs}||header||payload)$. Если он не правильный, ответить сообщением bad_record_mac.

Таким образом утекает информация об открытом тексте: злоумышленник посылает шифротекст и узнает, последние байты открытого текста - правильный *pad* или нет. Также различается время отклика в случае правильного и неправильного *pad*.

Протокол TLS 1.2 отвечает одним и тем же сообщением при любой ошибке и согласовывает заново ключи симметричного шифра и кода аутентичности после неправильного *pad* или *tag*, поэтому он не уязвим к этой атаке.

Опишем атаку на шифр при оракуле правильного окончания блока.

Пусть злоумышленник подслушал шифротекст из трех блоков, $c = (IVc[1]c[2])$ и хочет узнать $m[2]$ - значение последнего блока исходного текста.

Злоумышленник предполагает, что последний байт $m[2]$ равен g. Он добавляет $g\oplus$ 0x01 к последнему байту $c[1]$ и отправляет шифротекст $(IV, c'[1], c[2])$. См. рис. 24

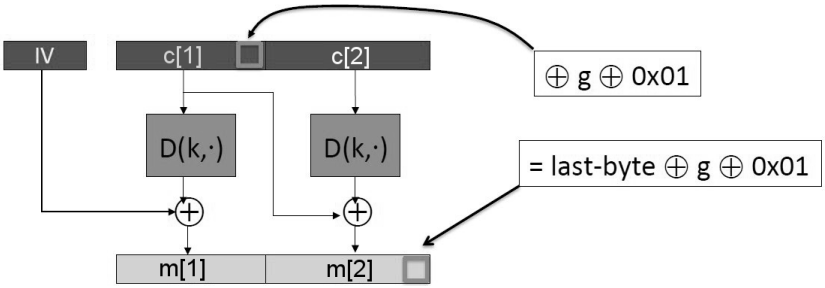

Рис. 24: Подбор значения последнего байта сообщения. По материалам [4].

Если последний байт $m[2]$ равен g, дополнение сообщения до длины блока будет верным, 0x01. Иначе - неверным. Проделаем это для $g = 0, 1, ..., 255$ - узнаем значение g_0 последнего байта $m[2]$. Потом - для второго с конца байта: к последнему байту $c[2]$ добавляем $g_0\oplus$ 0x02, тогда последний байт расшифрованного текста равен 0x02; к предпоследнему байту $c[2]$ добавляем (g$\oplus$0x02) с перебором по $g = 0, 1, ..., 255$. Когда два последних байта расшифрованного текста будут равны 0x0202, дополнение до длины блока будет верным. Соответствующее значение g сохраним как g_1 - это значение второго с конца байта блока $m[2]$. И т.д.

Замечание. Режим шифрования MAC-then-Encrypt, где шифр - в режиме CBC, может быть уязвим к этой атаке. Режим Encrypt-then-MAC не подвержен этой уязвимости. Режим MAC-then-Encrypt, где шифрование в режиме RandCTR, также не подвержен этой уязвимости, т.к. сообщение не нужно дополнять до длины блока.

9 Создание сессионных ключей

В целях безопасности срок жизни сессионного ключа ограничен по времени или объему передаваемых данных. Поэтому по одному первичному ключу нужно создать много сессионных ключей.

Пусть SK - первичный ключ, $SK \xleftarrow{R} K$. Пусть F - ПСФ, CTX - уникальный контекст для процесса, L - количество сессионных ключей.

Определение 9.1. *Функция создания сессионных ключей* (key derivation function) $KDF(SK, CTX, L) = F(SK, (CTX||0)), ..., F(SK, (CTX||L-1))$.

Пользователь выбирает пароль $\Rightarrow$ не равномерное распределение SK. Возможен подбор производного ключа путем перебора по словарю из наиболее часто использующихся SK. Добавим случайности.

Определение 9.2. *Соль* (salt) - случайные, равномерно распределенные данные, которые подаются на вход функции создания ключей вместе с секретным ключом для получения сессионных ключей или хэш-значения от секретного ключа.

Пусть H - криптостойкая хэш функция. Вычисление сессионного ключа идет в два шага:
1) $k = H(salt, SK)$ - равномерно распределенный случайный ключ (extract).
2) $KDF(k, CTX, L) = F(k, (CTX||0)), ..., F(k, (CTX||L-1))$ (expand).

Пример 9.1. HKDF (hash based key derivation function) - функция создания производных ключей, основанная на криптографической хэш функции (документ RFC-5869). Современные функции: PBKFD2, bcrypt. Функция PBKDF2 реализована в библиотеке OpenSSL на языке C++, а также в других библиотеках и языках.

10 Хранение паролей

Возможно несколько способов хранения паролей. Перечислим их в порядке возрастания безопасности.

1. Храним пары логин, пароль открытым текстом. К ним имеет доступ администратор системы. Их возможно украсть.

2. Зашифруем пары логин, пароль. Ключ шифра хранится на той же машине, поэтому администратор все равно знает все пароли.

3. Пусть H - криптографическая хэш функция. Храним пары $(login, H(pwd))$, при вводе пароля сравниваем $H(user\_input)$ с соответствующей записью. Администратор не знает наши пароли. Возможна атака перебором по словарю. Действия злоумышленника:

- если он выкрадет файл с парами $(login, H(pwd))$, то быстро узнает простые пароли по заранее вычисленным для всего словаря значениям хэшей.

- если не сможет выкрасть этот файл, будет перебирать пароли из словаря, система будет вычислять хэши от них, и с высокой вероятностью найдется совпадение.

4. Используем медленную хэш функцию $H^{(c)}$ - это суперпозиция H, сделанная c раз. Храним $(login, H^{(c)}(pwd))$. Значение c - публичная величина. Сохраняется уязвимость к атаке перебором по словарю. Действия злоумышленника:

- если он выкрадет файл, быстро узнает пароли по заранее вычисленным для всего словаря значеним хэшей.
- если не сможет выкрасть файл, будет перебирать пароли из словаря, система будет вычислять хэши от них, с высокой вероятностью найдется совпадение. Атака займет больше времени, т.к. хэш функция вычисляется медленно (до 0.1 секунды пользователь не заметит). Атаку ускоряет использование так называемых радужных таблиц, обеспечивающих управляемый перебор паролей для данной хэш функции $H^{(c)}$.

5. Лучшее решение: используем медленную хэш функцию и соль. Храним $(login, salt, H^{(c)}(salt, pwd))$. Вычисляем $H^{(c)}(salt, user\_input)$. Для каждого пользователя предпочтительно иметь свою соль.

- если злоумышленник выкрадет файл, то чтобы узнать пароли пользователей, ему необходимо заново вычислить для всего словаря значения функции $H^{(c)}(salt, \cdot)$.
- если злоумышленник не сможет выкрасть файл, он будет перебирать пароли из словаря и для каждого значения пароля - все возможные варианты соли. Это очень долго и защищает от использования атаки с радужными таблицами.
- если для разных пользователей разные значения соли, то у них разные функции $H^{(c)}(salt, \cdot)$, поэтому злоумышленнику придется подбирать каждый пароль по отдельности, а не все пароли параллельно.

Часть III

Асимметричные криптосистемы

11 Теория чисел и конечные группы

Основная цель раздела: изложить факты теории чисел и теории групп, необходимые для изучения асимметричных криптосистем.

11.1 Обобщенный алгоритм Евклида

Пусть даны $n \geq m \in \mathbb{N}$. Требуется найти n', m': $nn' + mm' = \text{НОД}(n, m)$. Найдем их с помощью обобщенного алгоритма Евклида. Обозначение алгоритма: $ExtGCD$.

1. Условие остановки. $\text{НОД}(n, 0) = n$ по определению. Тогда

   $n \cdot 1 + 0 \cdot 0 = n$

   $n' = 1$, $m' := 0$

2. Шаг рекурсии (вниз). $n_i n'_i + m_i m'_i = \text{НОД}(n, m)$.

   $n_{i+1} = m_i$

   $m_{i+1} = n_i - [n_i/m_i]m_i = n_i \bmod m_i \qquad (*)$

   и снова верно

   $n_{i+1}n'_{i+1} + m_{i+1}m'_{i+1} = \text{НОД}(n, m)$.

   Шаг рекурсии сделан.

3. Вычисление n'_i, m'_i (вверх).

   В уравнение $n_{i+1}n'_{i+1} + m_{i+1}m'_{i+1} = \text{НОД}(n, m)$ подставим (*):

   $m_i n'_{i+1} + (n_i - [n_i/m_i]m_i)m'_{i+1} = \text{НОД}(n, m)$, перегруппируем:

   $n_i \cdot m'_{i+1} + m_i \cdot (n'_{i+1} - [n_i/m_i]m'_{i+1}) = \text{НОД}(n, m)$.

   $n_i \cdot n'_i + m_i \cdot m'_i = \text{НОД}(n, m)$

   Значит, $n'_i := m'_{i+1}$, $m'_i := n'_{i+1} - [n_i/m_i]m'_{i+1}$.

Пример 11.1. Найти $9^{-1} \bmod 160$. Все вычисления алгоритма $ExtGCD(160, 9)$ запишем в таблицу (см. табл. 6). Первый шаг: $160 - [160/9] \cdot 9 = \mathbf{7}$.

$\Rightarrow 9^{-1} \bmod 160 = -71 = 89 \bmod 160$

i	n	m	$[n/m]$	n'	m'
1	160	9	17	4	$-3-[160/9]\cdot 4=-71$
2	9	**7**	1	-3	4
3	7	2	3	1	-3
4	2	1	1	0	$\underline{1}-[2/1]\cdot\underline{0}=1$
5	1	0		$\underline{1}$	$\underline{0}$

Таблица 6: Вычисление НОД с помощью расширенного алгоритма Евклида.

Утверждение 11.1. Расширенный алгоритм Евклида имеет полиномиальную сложность.

Доказательство. Арифметические операции + - * и деление с остатком имеют полиномиальную сложность. Докажем, что число вычислений НОД(n_0, m_0), $n_0 \geq m_0$ равно $O(log\, m_0)$.

Ход рекурсии: $ExtGCD(n_0, m_0) \to ExtGCD(n_1, m_1) \to .. \to$ НОД$(n_k, 0)$. Докажем, что на любом шаге $m_{i+2} \leq m_i/2$. Из описания алгоритма $m_{i+1} < m_i\ \forall i$. Если $m_{i+1} \leq m_i/2$, то сразу доказано.

Пусть $m_i > m_{i+1} \geq m_i/2$. Тогда следующий рекурсивный вызов имеет вид: $ExtGCD(n_{i+1} = m_i,\ m_{i+1})$ и на следующем за ним шаге $m_{i+2} = m_i\ mod\ m_{i+1} = m_i - m_{i+1} < m_i/2$.

Следовательно, число рекурсивных вызовов не более $2log(m_0)$. □

11.2 Конечные группы и поля

Определение 11.1. Пусть G - непустое множество, $*$ - бинарная операция, определенная $\forall a, b \in G$. $(G, *)$ - *группа*, если:

- $\forall a, b \in G\ a * b \in G$ (замкнутость)
- $\forall a, b, c \in G\ (a * b) * c = a * (b * c)$ (ассоциативность)
- $\exists e \in G : \forall a \in G\ a * e = e * a = a$ (существование нейтрального элемента)
- $\forall a \in G\ \ \exists a^{-1} \in G : a * a^{-1} = a^{-1} * a = e$ (существование обратного элемента)

Определение 11.2. Группа $(G, *)$ называется *абелевой (коммутативной)*, если $\forall a, b \in G\ a * b = b * a$.

Определение 11.3. *Порядок группы* $(G, *)$ - мощность G, обозначается $|G|$.

Определение 11.4. Возведение элемента группы в степень: пусть $a \in G, i \in N \;\; a^i = (a * ... * a)$ i раз. $a^0 = e$.

Определение 11.5. *Порядок элемента* $a \in G$ - это $min\ n \neq 0 : \;\; a^n = e$.

Определение 11.6. *Подгруппа* группы $(G, *)$ - подмножество G, которое является группой относительно $*$.

Определение 11.7. *Смежный класс* множества H по элементу a - это $aH = \{ax | x \in H\}$

Пример 11.2. $\mathbb{Z}_{10}^+$ - группа. $H = \{0, 5\}$ - ее подгруппа. Смежные классы: $0 + H, \; 1 + H, \; 2 + H, \; 3 + H, 4 + H$.

Теорема 11.1 (Лагранжа)**.** Пусть H - подгруппа G. Тогда $|G| = |H|\cdot$(количество смежных классов

Без доказательства.

Пример 11.3. Группа вычетов по модулю N по сложению: $\mathbb{Z}_N = (\{0, 1, ..N-1\}, +)$. Группа по умножению $\mathbb{Z}_N^* = (\{i \in \mathbb{Z}_N | \text{НОД}(i, N) = 1\}, *)$.

В группе $\mathbb{Z}_N^*$ для вычисления обратного элемента используется обобщенный алгоритм Евклида.

Следствие. Если G - группа конечного порядка k, то порядок k_1 любой ее подгруппы G_1 является делителем порядка группы.

Определение 11.8. Группа называется *циклической*, если $\exists a \in G : \;\; \forall b \in G \; \exists i \in N : \; b = a^i$. a называется *генератором* (*порождающим элементом*) группы. Обозначение: $G = \; \langle a \rangle$.

Следствие. Порядок любого элемента - делитель порядка группы.

Следствие. Пусть $m = |G|$. Тогда $\forall a \in G \; a^m = 1$.

Задача 11.1. Вычислить: $5^{38} \bmod 9$, $3^{663} \bmod 7$.

Оценим сложность операций в группе Z_m^*. Пусть $l = log_2(m)$ - длина m. Пусть $n = |Z_m^*|$. Позже будет показано, что $|Z_m^*| = \varphi(m)$ - значение функции Эйлера. Если известно разложение числи m на простые множители, оно может быть вычислено алгоритмом полиномиальной сложности от l.

Очевидно, что умножение двух элементов в Z_m^* имеет полиномиальную сложность от l.

Возведение $a \in Z_m^*$ в степень $k \in \mathbb{N}$ происходит с помощью быстрого алгоритма возведения в степень (через разложение k по степеням числа 2). Алгоритм имеют полиномиальную сложность от l и длины k. Если значение n известно, можно предварительно вычислить $k' \equiv k \bmod n$, тогда алгоритм будет иметь полиномиальную сложность от l.

Если значение n известно, для вычисления a^{-1} можно использовать следствие из теоремы Лагранжа: $\forall a \in G\ a^{|G|} = 1 \Rightarrow a^{|G|-1} = a^{-1}$ и быстрый алгоритм возведения в степень. В группах вычетов Z_m^* для вычисления a^{-1} используется обобщенный алгоритм Евклида, т.к. он не использует значение n и работает немного быстрее. $ExtGCD(a, m) \Rightarrow aa' + mm' = 1 \Rightarrow aa' \equiv 1 \bmod m$. В любом случае, вычисление обратного элемента имеет полиномиальную сложность от l.

Т.е. все операции в Z_m^* имеют полиномиальную сложность от l.

Определение 11.9. *Кольцо* $(R, +, *)$ (или просто R) - это множество с двумя операциями:

- R - абелева группа по $+$
- $\forall a, b \in G\ a * b \in G$ (замкнутость $*$)
- $\forall a, b, c \in G\ (a * b) * c = a * (b * c)$ (ассоциативность $*$)
- $\forall a, b, c \in G\ a(b + c) = ab + ac$ и $(b + c)a = ba + ca$ (дистрибутивность справа и слева)

Если операция $*$ коммутативна, то это коммутативное кольцо.

Определение 11.10. Если в кольце R $a * b = 0$, то a, b называются *делителями нуля.*

Определение 11.11. *Поле* $(F, +, *)$ (или просто F) - это множество с двумя операциями:

- $(F, +, *)$ - кольцо
- $(F \setminus \{0\}, *)$ - абелева группа

Нейтральный элемент поля по $+$ обозначается 0. Нейтральный элемент по $*$ обозначается 1. Поле - коммутативное кольцо с $1 \neq 0$ без делителей нуля.

Определение 11.12. *Характеристика поля* - это $min\ n \in \mathbb{N}$ такое, что сумма n копий единицы равна нулю: $n \cdot 1 = 0$. Если такого числа не существует, то характеристика равна 0.

Определение 11.13. Множество $H \subseteq F$ - *подполе* поля $(F, +, *)$, если H замкнуто относительно операций $+, *$.

Определение 11.14. Поле $H \supset F$ - *расширение* поля F, если F - подполе H.

Определение 11.15. *Поле Галуа* - это конечное поле. Обозначается $GF(q)$, где q - порядок поля.

Определение 11.16. *Простое поле* - это поле, не содержащее строго вложенных в него подполей.

Свойства поля:

- Характеристика конечного поля - простое число. (Доказать от противного.)
- Пусть F - конечное поле, k - его порядок. Тогда $k = p^n$, p - простое, $n \in N$. Причем p - характеристика поля. Без доказательства.
- Для любого p^n (p - простое) существует единственное (с точностью до изоморфизма) поле порядка p^n, обозначаемое $GF(p^n)$. Без доказательства.
- Мультипликативная группа F_q^* поля $GF(q)$ - циклическая группа порядка $(q-1)$. Ее генератор называется *примитивным элементом* поля. Без доказательства.

Пример 11.4. Не циклическая группа: $\mathbb{Z}_8^* = (\{1, 3, 5, 7\}, *)$. $|G| = 4$. $g_i^2 = e\ \forall i$.

Пример 11.5. Циклическая группа: $\mathbb{Z}_7^*$.

Пример 11.6. GF(2) - это поле: {0,1}, + это XOR, $*$ это AND.

Обозначим $F_p[x]$ - кольцо полиномов не ограниченной степени с коэффициентами из поля $GF(p)$. Полином - это $a_n * x^n + ... + a_0, \quad a_i \in F_p, \quad x$ - формальный символ.

Произведение полиномов вычисляется по обычным правилам: степени x складываются в кольце $\mathbb{Z}$; операции над коэффициентами производятся в поле $GF(p)$. Деление полиномов с остатком производится по правилам деления “в столбик”, операции над коэффициентами производятся в поле $GF(p)$.

Определение 11.17. Многочлен называется *неприводимым*, если он не имеет делителей в $F_p[x]$, т.е. не разлагается в произведение других многочленов.

Пример 11.7. $x^4 + x^2 + 1 = (x^2 + x + 1)(x^2 + x + 1)$ - не является неприводимым, не имеет корней.

Пусть $q = p^n$. $\forall a \in GF(q)$ одночлен $x - a$ имеет один корень a. Рассмотрим многочлен $f(x) = x^q - x$ над $GF(q)$. В поле нет делителей нуля, поэтому $|GF^*(q)| = q - 1$. $\Rightarrow \forall a \in GF^*(q)\ a^{q-1} = 1$. $\Rightarrow$ $f(x)$ имеет q различных корней в $GF(q)$, т.е. $f(x) = \prod\limits_{a \in GF(q)} (x - a)$.

11.3 Поиск генератора циклической группы

Пусть G - циклическая группа по умножению. Как найти генератор группы? Возьмем $a \overset{R}{\leftarrow} GF^*(q) \setminus \{1\}$. Проверим, является ли этот элемент генератором.

Пусть $q - 1 = p_1^{i_1} * .. * p_k^{i_k}$ - разложение на простые множители. Вычислим $b_j = a^{(q-1)/p_j}$, $j = 1, ..., k$. Если $\exists b_j = 1$, то a - не генератор. Иначе a - генератор $GF^*(q)$.

Задача 11.2. Обосновать этот алгоритм.

Пример 11.8. $GF^*(13)$. $|GF^*(13)| = 12 = 2 \cdot 2 \cdot 3$.

$2^4 \equiv 3 \bmod 13$; $2^6 \equiv 12 \bmod 13$. $\Rightarrow$ 2 - генератор $GF^*(13)$

$5^4 \equiv 1 \bmod 13$. $\Rightarrow$ 5 - не генератор $GF^*(13)$

11.4 Квадратичные вычеты

Пусть p - простое. Линейное уравнение $ax + b = 0$ всегда имеет решение в $\mathbb{Z}_p^*$: $x = -ba^{-1}$. Существует ли решение уравнения $x^k - c = 0,\ \ k \geq 2$ в $\mathbb{Z}_p^*$?

Определение 11.18. $x \in \mathbb{Z}_p : x^k = c$ называется *корнем k-й степени из c в* $\mathbb{Z}_p$.

Пример 11.9. В $\mathbb{Z}_{11}^*$ $7^{1/3} = 6$, $3^{1/2} = 5$, $2^{1/2}\ \nexists$.

В общем случае, не известно полиномиальных алгоритмов вычисления $c^{1/a}$ в $\mathbb{Z}_n^*$, которые не использует разложение n на множители.

Лемма 11.1. Пусть F - поле, например $GF(p)$, где p - простое число. Полином $f(x)$ имеет корень $c \in F \Leftrightarrow f(x) = (x - c)g(x)$, где $g(x)$ - полином от x.

Доказательство. $\Leftarrow$: очевидно.

$\Rightarrow$: Пусть полином $f(x)$ имеет корень c. Поделим его на $x - c$ с остатком: $f(x) = (x - c)g(x) + d$, где d - это константа. $f(c) = 0,\ (c - c)g(c) = 0 \Rightarrow d = 0.$ $\square$

Определение 11.19. $x \in \mathbb{Z}_n^*$ называется *квадратичным вычетом* (quadratic residue), если $\exists\, x^{1/2} \in Z_n*$.

Введем обозначение: $QR(n) = \{x \in Z_n^* | \exists y \in Z_n^* : y^2 = x\}$ - множество квадратичных вычетов в группе Z_n^*. $QNR(n) = Z_n^* \setminus QR(n)$ - множество квадратичных невычетов.

Утверждение 11.2. $|QR(p)| = \frac{1}{2}|Z_p^*|$.

Доказательство. Очевидно, $QR(p)$ - подгруппа Z_p^*. $\Rightarrow |QR(p)| \leq \frac{p-1}{2}$. Пусть $x, y \in Z_p^*$, $x \neq \pm y$. Пусть $x^2 = y^2$. $\Rightarrow (x + y)(x - y) = 0$, т.е. либо $x = y$, либо $x = -y$. Противоречие. Значит, $x \neq \pm y \Rightarrow x^2 \neq y^2$, т.е. возведение в квадрат - отображение 2 в 1: каждым двум элементам $\pm x$ ставится в соответствие один элемент x^2. $\Rightarrow |QR(p)| = \frac{p-1}{2}$. $\square$

Лемма 11.2. Если $p = 3 \bmod 4$, p - простое, $c \in \mathbb{Z}_p^*$ - квадратичный вычет, то $c^{1/2} = c^{(p+1)/4}$

Доказательство. $(c^{(p+1)/4})^2 = c^{(p+1)/2} = c^{(p-1)/2} \cdot c = 1 \cdot c = c$. Использовали факт, что $(p + 1)/4$ - целое. $\square$

Замечание. если p - простое, $p = 1 \bmod 4$, в Z_p^* также $\exists$ полиномиальный алгоритм вычисления квадратичного корня.

11.5 Функция Эйлера

Определение 11.20. Пусть $n \in \mathbb{N}$. *Функция Эйлера* $\varphi(n) = |\mathbb{Z}_n^*|$.

Свойства функции Эйлера:

- Если p - простое, $\varphi(p) = p - 1$.
- Пусть $p \neq q$ - простые. $\varphi(pq) = pq - p - q + 1 = (p - 1)(q - 1)$.
- Пусть НОД$(n, m) = 1$. Тогда $\varphi(nm) = \varphi(n) \cdot \varphi(m)$.
- Пусть $n = \prod\limits_i p_i^{e_i}$ - разложение на простые множители. Тогда $\varphi(n) = \prod\limits_i p_i^{e_i - 1}(p_i - 1)$, вычисляется полиномиальным алгоритмом.

Задача 11.3. Доказать свойства функции Эйлера.

Теорема 11.2 (Эйлера). $\forall x \in \mathbb{Z}_N^* \;\; x^{\varphi(N)} = 1$

Доказательство. Это следствие малой теоремы Ферма. □

11.6 Символ Лежандра

Пусть a - целое число, и p - простое нечетное число.

Определение 11.21. $\left(\frac{a}{p}\right)$ - *символ Лежандра.* $\left(\frac{a}{p}\right) = a^{(p-1)/2} \in \{0, 1, -1\}$ в $\mathbb{Z}_p$.

Свойства символа Лежандра:

- $\left(\frac{a}{p}\right) = 0 \Leftrightarrow a = kp, \; k \in N$
- $\left(\frac{a}{p}\right) = 1 \Leftrightarrow a \in QR(p)$
- $\left(\frac{a}{p}\right) = -1 \Leftrightarrow a \in QNR(p)$
- $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$

Доказательство. $\forall a \in \mathbb{Z}_p^* \; a^{p-1} - 1 = 0 \Leftrightarrow (a^{(p-1)/2} - 1)(a^{(p-1)/2} + 1) = 0$.

Если $a \in QR(p)$, то $\exists b \in \mathbb{Z}_p^* : a = b^2$, $a^{(p-1)/2} = b^{p-1} = 1$. Ровно $(p-1)/2$ квадратичных вычетов - корни первого сомножителя. Значит, все квадратичные невычеты - корни второй скобки.

Коммутативность напрямую следует из определения. □

11.7 Китайская теорема об остатках

Утверждение 11.3. Пусть G, H - группы. Определим прямое произведение групп $Q = G \times H$:

- элементы Q - все упорядоченные пары элементов (g, h)
- групповая операция: $q_1, q_2 \in Q$, $q_1 \circ_Q q_2 = (g_1 \circ_G g_2, h_1 \circ_H h_2)$

Тогда $Q = G \times H$ - группа.

Доказательство. Проверить аксиомы группы. Самостоятельно. □

Очевидно, прямое произведение обобщается на любое конечное количество групп.

В 100 г. до н.э., китайский ученый Sun Tsu решил следующую задачу: найти число, дающее при делении на 3, 5, 7 остатки 2, 3, 2 соответственно.

Теорема 11.3 (Китайская теорема об остатках, формулировка для чисел)**.** Если числа $a_1, .., a_n$ попарно взаимно просты, то для любых остатков $r_1, .., r_n : 0 \leq r_i < a_i$ найдется число $m : m = r_i \bmod a_i \ \forall i = 1, .., n$

Теорема 11.4 (Китайская теорема об остатках, формулировка для классов вычетов.)**.** Пусть $n = pq$, НОД$(p, q) = 1$. Тогда $Z_n \simeq Z_p \times Z_q$ и $Z_n^* \simeq Z_p^* \times Z_q^*$. Кроме того, пусть $f : Z_n \to Z_p \times Z_q,\ f(x) = (x \bmod p,\ x \bmod q)$.
Тогда f - изоморфизм $Z_n^+ \to Z_p \times Z_q$ и изоморфизм $Z_n^* \to Z_p^* \times Z_q^*$.

Доказательство. Очевидно, что f действует из Z_n в $Z_p \times Z_q$ и образ x единственный. Докажем, что f - биекция.

1. Инъективность. Докажем от противного для Z_n^*. Пусть $x \in Z_n^*,\ f(x) = (x_p, x_q)$. Пусть $x_p \notin Z_p^*$. Тогда НОД$(x \bmod p, p) \neq 1. \Rightarrow$ НОД$(x, p) \neq 1. \Rightarrow$ НОД$(x, pq) \neq 1$. Противоречие. $\Rightarrow x_p \in Z_p^*$. Аналогично $x_q \in Z_q^*$. $\Rightarrow (x_p, x_q) \in Z_p^* \times Z_q^*$.
2. Сюръективность. Докажем для Z_n^+ и Z_n^*. Отображение $f(x)$ определено однозначно. Пусть $\exists x \neq x' \in Z_n,\ f(x) = (x_p, x_q) = f(x')$. Тогда $x \equiv x_p \equiv x' \bmod p$ и $x \equiv x_q \equiv x' \bmod q$. $\Rightarrow (x - x')$ делится на p и на q. НОД$(p, q) = 1,\ pq = n$ $\Rightarrow (x - x')$ делится на n. Это невозможно, т.к. $0 < |x - x'| < n$. Значит, $x \neq x' \in Z_n \Rightarrow f(x) \neq f(x')$.
   $\Rightarrow f$ - отображение 1 в 1. Множества Z_n, $Z_p \times Z_q$ конечные, равной мощности. $\Rightarrow f$ - биекция.
3. f сохраняет операцию. Докажем для группы по сложению Z_n^+. Обозначим $x +_p x = x + y \bmod p$. По определению, $(a_p,\ a_q) + (b_p, b_q) = ((a_p +_p b_p),\ (a_q +_q b_q))$.
   Заметим, что т.к. $n = pq$, то $\forall c \in Z\ c \bmod n \bmod p = c \bmod p$. По определению функции f и группы Z_n имеем $f(a +_n b) = ((a +_n b) \bmod p,\ (a +_n b) \bmod q) = ((a +_p b),\ (a +_q b)) = (a \bmod p, a \bmod q) + (b \bmod p, b \bmod q) = f(a) + f(b)$.
   Сохранение операции для групп по умножению доказывается аналогично.

□

Замечание. Для $n = p_1 \cdot ... \cdot p_t$ утверждение доказывается, применив эту теорему $(t - 1)$ раз.

Мы доказали, что отображение $x \to (x_p, x_q)$ - биекция и сохраняет операцию. Как сделать обратный переход, от вектора остатков к числу?

Очевидно, $\exists e_p, e_q \in \mathbb{Z}_n : (1, 0) \leftrightarrow e_p \in \mathbb{Z}_n,\ (0, 1) \leftrightarrow e_q \in \mathbb{Z}_n$. Пусть $x \to (x_p, x_q)$. Тогда $(x_p, x_q) = x_p \cdot (1, 0) + x_q \cdot (0, 1) \leftrightarrow x_p \cdot e_p + x_q \cdot e_q \bmod n = x \in \mathbb{Z}_n$.

Как найти $e_p \in \mathbb{Z}_n$? По обобщенному алгоритму Евклида, $\exists p', q' : p'p + q'q = 1$ $\Rightarrow e_p = q'q$, $e_q = p'p$. Таким образом, мы полностью описали алгоритм обратного переход от пары (x_p, x_q) к числу x.

Если $n = p_1^{a_1} \cdot ... \cdot p_t^{a_t}$, нужно применить обобщенный алгоритм Евклида $(t-1)$ раз. Первый раз - для $p_1^{a_1} \cdot ... \cdot p_{t-1}^{a_{t-1}}$ и $p_t^{a_t}$, и т.д., отщепляя по одному делителю.

Итак, мы полностью обосновали операции с большими числами с использованием китайской теоремы об остатках.

Пример 11.10. Применение КТО: пусть $a_1, ..., a_n$: НОД$(a_i, a_j) = 1$ $\forall i \neq j$. Тогда система уравнений $x = r_i \bmod a_i$, $i = 1..n$ всегда имеет единственное решение x на множестве чисел $\{i | 0 \leq i < m = a_1 \cdot ... \cdot a_n\}$.

Поэтому в прикладных библиотеках (например, библиотеке GMP) для работы с большими целыми числами используются классы вычетов по модулю больших простых чисел, каждое из которых укладывается в целочисленный тип данных. Операции с большими числами производятся так: числа отображаются в векторы классов вычетов, производятся операции над векторами классов вычетов, их результат отображается в число.

Пример 11.11. Пусть $n = 9 \cdot 20$, вычислить $7 \cdot 11$.

Имитируем работу с большими числами через векторы классов вычетов. Важно, что $7 \cdot 11 < n$, иначе получили бы результат по модулю n (или надо было бы добавлять третье число к набору взаимно простых чисел $(9, 20)$).

$7 \cdot 11 \to (7, 7) \cdot (2, 11) = (7 \cdot 2 \bmod 9, 7 \cdot 11 \bmod 20) = (5, 17)$.

$ExtGCD(9, 20) \Rightarrow -11 \cdot 9 + 5 \cdot 20 = 1$

$(1, 0) \leftrightarrow 5 \cdot 20 \equiv 100 \bmod n$, $(0, 1) \leftrightarrow -11 \cdot 9 \equiv 81 \bmod n$

$7 \cdot 11 \leftrightarrow 5 \cdot 100 + 17 \cdot 81 \equiv 140 + 117 \equiv 77 \bmod n$.

11.8 Поиск простых чисел

11.8.1 Тест на основе малой теоремы Ферма

Тест простоты числа на основе малой теоремы Ферма:

- выбираем случайное a из $\{1, .., n-1\}$.
- если НОД$(a, n) \neq 1$, n - составное.
- если $a^{n-1} \not\equiv 1 \bmod n$, то n - составное. Назовем число a свидетелем этого.
- иначе ответ не известен, но можно повторить тест еще раз.

Утверждение 11.4. Если n составное и в $\mathbb{Z}_n^*$ $\exists$ свидетель того, что n - составное, то количество свидетелей в $\mathbb{Z}_n^*$ не меньше $|\mathbb{Z}_n^*|/2$.

Доказательство. пусть $B \subseteq \mathbb{Z}_n^*$: все элементы B - не свидетели простоты n. Заметим, что $1^{n-1} = 1 \bmod n$, т.е. $1 \in B$. Если $a, b \in B$, то $a^{n-1} \cdot b^{n-1} = 1 \cdot 1 = 1$ т.е. $a \cdot b \in B$. Значит B - подгруппа $\mathbb{Z}_n^*$ и $|B| \leq |\mathbb{Z}_n^*|/2$. □

Пусть в $\mathbb{Z}_n^*$ $\exists$ свидетели того, что n - составное. Оценим вероятность, что на одном шаге теста мы выберем либо свидетеля того, что n - составное, либо число не взаимно простое с n.

$$p \geq \frac{|\mathbb{Z}_n^*|/2 + \big((n-1) - |\mathbb{Z}_n^*|\big)}{n-1} = 1 - \frac{|\mathbb{Z}_n^*|/2}{n-1} \geq 1 - \frac{|\mathbb{Z}_n^*|/2}{|\mathbb{Z}_n^*|} = \frac{1}{2}$$

При повторении алгоритма t раз вероятность ложно положительного теста на простоту будет не более 2^{-t}.

Недостаток этого теста: $\exists$ псевдопростые по этому тесту числа n, для которых $\forall a \; a^{n-1} = 1 \bmod n$.

Определение 11.22. *Числа Кармайкла* - это составные числа n: $\forall b \in Z_n^* \; b^{n-1} = 1 \bmod n$.

Т.е. для них нет свидетелей того, что они составные. Первые числа Кармайкла: 561, 1105, 1729, 2465, 2821. Их бесконечно много.

11.8.2 Тест Миллера-Рабина

Утверждение 11.5. Пусть n - простое нечетное число и $s, t: \; n - 1 = t2^s$, где t - нечетное, и пусть $a \in \mathbb{Z}_n^*$. Тогда

- или $\exists b \in \{a^t, a^{2t}, ..a^{(2^{s-1})t}\}$: $b \equiv -1 \bmod n$,
- или $a^t \equiv 1 \bmod n$

Это необходимое условие того, что число простое.

Доказательство. n - простое нечетное $\Rightarrow$ уравнение $x^2 = 1 \bmod n$ имеет только тривиальные корни 1, -1. $\forall a \in \mathbb{Z}_n^* \; a^{n-1} = 1 \bmod n$. Поэтому, извлекая из a^{n-1} квадратный корень, мы получим одно из двух чисел: 1, -1.

Извлечем квадратный корень не более s раз. Если на некотором шаге получили -1, то $\exists q : 0 \leq q \leq s-1$, $a^{t2^q} \equiv -1 \bmod n$. Если ни разу не получили -1, в т.ч. $a^t \not\equiv -1 \bmod n$, то $a^t \equiv 1 \bmod n$. □

Определение 11.23. Пусть n - составное. $a < n$ - *лжесвидетель* того, что n - простое, если для него выполняются условия предыдущего утверждения.

Теорема 11.5 (Рабина)**.** Пусть n - нечетное составное. Тогда количество лжесвидетелей того, что n - простое, не более $\varphi(n)/4$.

Без доказательства. Доказательство этого факта не сложное, но довольно длинное. Его можно прочитать в статье [14].

Тест Миллера-Рабина на простоту числа n.

Представим $n-1$ в виде $n-1 = t2^s$, где t - нечетное.

- выберем $a \xleftarrow{R} \{2, ..., n-1\}$.
- если НОД$(a, n) \neq 1$, то n - составное.
- если $a^t \equiv \pm 1 \bmod n$, то n может быть простым.
- если среди чисел $a^{2t}, .., a^{(2^{s-1})t} \bmod n$ нашлось $-1 \bmod n$, то n может быть простым.
- если ни одно из этих чисел не равно $-1 \bmod n$, n - составное, т.к. не выполнено необходимое условие простоты.

Из теоремы Рабина следует, что вероятность ложно положительного ответа одного теста (т.е. признать составное число простым) равна $\frac{1}{4}$. Можно повторить этот тест несколько раз с разными значениями a.

11.8.3 Детерминированный полиномиальный критерий простоты

$\exists$ несколько детерминированных алгоритмов - тестов простоты. Самый быстрый из них имеет полиномиальную сложность относительно длины входа - числа бит записи n. Он всегда дает точный ответ, не использует не доказанную гипотезу. Он не дает разложение на множители. Это алгоритм AKS, опубликованный в статье [15].

Определение 11.24. $f(x)$ - *субэкспоненциальная функция*, если
$\lim\limits_{x\to\infty} f(x)x^{-n} = \infty \; \forall n$ и $\lim\limits_{x\to\infty} (log\, f(x))/x = 0$.

Например, алгоритм со сложностью 2^{n^α}, $0 < \alpha < 1$ - субэкспоненциальный.

Замечание. $\exists$ детерминированные субэкспоненциальные алгоритмы факторизации чисел.

Замечание. Если p - простое, но $p-1$ имеет *только* малые простые множители, то $\exists$ полиномиальный алгоритм факторизации числа $p-1$. Поэтому для построения криптосистем выбираются p: у $p-1$ есть хотя бы один большой простой множитель.

11.8.4 Выбор случайного простого числа

Для ряда асимметричных криптосистем нужно выбирать большие простые числа случайно и равновероятно. Для этого выбирается случайное целое число из заданного диапазона и проверяется на простоту по одному из вероятностных тестов. Например:

- выбрать случайно, равномерно n-битовое число p
- установить старший и младший биты в 1
- выполнить тест Миллера-Рабина пять раз для разных (случайных) чисел a
- если число p не проходит хотя бы один из тестов, оно составное.

Задача 11.4. Оценить сверху вероятность того, что составное число будет признано простым.

Пример 11.12. В программе Maple `isprime()` - детерминированная функция, реализована как тест Миллера-Рабина для $a = 2, 3, 5, 7, 11$.

11.9 Дискретный логарифм

Пусть $a, b \in Z_n^*$. Задача: найти $x \in \mathrm{N} : a^x = b$ - это задача поиска дискретного логарифма.

Для группы $\mathbb{Z}_p^*$, где p - простое, сложность вычисления дискретного логарифма - того же порядка, что сложность задачи факторизации числа $n \approx p : n = st$, где s, t - простые числа примерно одного порядка.

Т.е. задача поиска дискретного логарифма в $\mathbb{Z}_p^*$, где p - простое, имеет субэкспоненциальную сложность.

12 Асимметричные шифры

Основная цель раздела: описать асимметричный шифр и условия его стойкости к различным атакам, а также исследовать некоторые его реализации.

12.1 Основные определения

Определение 12.1. *Асимметричный шифр* (asymmetric cipher) - это тройка алгоритмов G, E, D:

$G(\cdot)$ - рандомизированный генератор пары ключей (открытый ключ, закрытый ключ) - (pk, sk). Может принимать $seed$ - начальное значение в качестве аргумента.
$E(\cdot,\cdot)$ - рандомизированный алгоритм шифрования: $c = E(pk, m)$.
$D(\cdot,\cdot)$ - детерминированный алгоритм расшифрования: $m = D(sk, c)$.
$\forall(pk, sk)$, созданной $G(\cdot)$, верно $D(sk, E(pk, m)) = m$

Порядок действий при использовании асимметричной криптосистемы:

1. Пользователь А создает пару ключей (pk_A, sk_A).
2. А распространяет свой ключ pk_A по открытым каналам. Например, сообщает его пользователю В.
3. В шифрует сообщение ключом pk_A, отправляет шифротекст А по открытому каналу.
4. А расшифровывает сообщение ключом sk_A.

Все, кто хочет отправить зашифрованное сообщение к А, будут шифровать сообщения одним и тем же публичным ключом pk_A пользователя А. Тот расшифровывает все входящие сообщения одним и тем же ключом sk_A. Только А знает ключ sk_A и может расшифровать шифротекст, зашифрованный с помощью pk_A.

Замечание. В таком виде система уязвима к атаке посредника, если злоумышленник встраивается в канал передачи данных на этапе передачи открытого ключа, и подлинность открытого ключа нельзя проверить в обход злоумышленника. (Например, проверить электронно-цифровую подпись ключа или значение хэш функции от ключа.) Т.е. существует проблема - управление открытыми ключами.

Эквивалентные определения:

Определение 12.2. *Односторонняя функция* (one way function) - это функция, для которой $\exists A \in$ PT, вычисляющий ее для любого входного значения. Но $\nexists B \in$ PPT : $P[f(B(f(x))) = f(x)] > \delta(n)$, гду $\delta(n)$ - не пренебрежимо малая функция, n - длина x.

Определение 12.3. Функция $f : \{0,1\}^* \to \{0,1\}^*$ является *односторонней функцией*, если она вычисляется за полиномиальное время на детерминированной машине Тьюринга, но не существует полиномиальной вероятностной машины Тьюринга, которая обращает эту функцию с более чем пренебрежимо малой вероятностью.

Замечание. Предположительно, это - односторонние функции (не доказано, что это так, но нет и свидетельств против):

- любая криптографическая хэш функция
- умножение двух больших простых чисел одинаковой длины и факторизация результата
- возведение в степень и логарифм в конечном поле

Если односторонняя функция $\exists \Rightarrow \mathrm{P} \neq \mathrm{NP}$.

Определение 12.4. *Односторонняя функция с секретом* (trapdoor one way function) - при неизвестном секрете это односторонняя функция по обоим аргументам, при известном секрете $\exists B \in \mathrm{PT}$, который вычисляет обратную функцию.

Замечание. Из определения односторонней функции следует, что она должна быть тяжело обращаема почти всегда, а не в худшем случае. Это отличается от определения принадлежности к классу NP, где сложность берется в худшем случае.

Пример 12.1. Пусть $\exists A \in \mathrm{PT}$, который для не пренебрежимо малой доли множества значений функции f находит прообраз. Тогда f - не односторонняя функция. Однако, если удастся выделить подмножество множества определения функции f, для образа которого не известен полиномиальный алгоритм обращения с не пренебрежимо малой вероятностью угадать прообраз, то на нем f будет (кандидатом на) односторонней функцией.

Пример 12.2. Факторизация $n = pq$, где p, q - простые числа одного порядка, - кандидат на одностороннюю функцию с секретом. Факторизация чисел вида $n = 2^q$ тривиальна.

На функциях, для которых не известно эффективного алгоритма вычисления обратной функции для всех допустимых входных данных, основываются методы асимметричной криптографии.

12.2 Стойкость к различным типам атак

Стойкость против подслушивания.

В случае симметричной криптосистемы злоумышленник не знал ключ и не мог сам строить по открытому тексту шифротекст, который примет система.

В случае асимметричной криптосистемы злоумышленник знает открытый ключ, он может сам создавать шифротексты для любых сообщений. Поэтому в асиммет-

ричной криптосистеме ключ всегда многоразовый. ⇒ асимметричное шифрование обязано быть рандомизированным.

Эксперимент CPA - атака с выбором открытого текста (см. рис. 25):

1. Система создает и фиксирует пару ключей pk, sk и значение бита $b \xleftarrow{R} \{0,1\}$. n - параметр длины ключа.
2. Злоумышленник получает значение публичного ключа pk.
3. Злоумышленник посылает несколько пар открытых текстов m_0, m_1 равной длины в систему, каждый раз получает шифротекст для одного из них, т.е. $c(pk, m_b)$.
4. Злоумышленник возвращает значение b' - пытается угадать значение b.

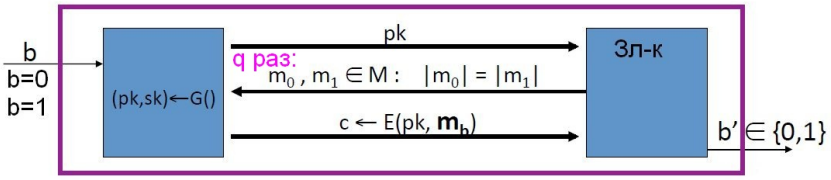

Рис. 25: Атака на асимметричную криптосистему с выбором открытого текста. По материалам [4].

Определение 12.5. Шифр $S = (G, E, D)$ *семантически стойкий к атаке с выбранным открытым текстом*, если $\forall A \in \text{PP}$

$Adv_{CPA}(A, S) = \left|P(b' = 1|b = 0) - P(b' = 1|b = 1)\right| < \varepsilon(n)$

Стойкость против активных атак.

Эксперимент CCA - атака с выбором шифротекста (см. рис. 26):

1. Система создает и фиксирует пару ключей pk, sk и значение бита $b \xleftarrow{R} \{0,1\}$. n - параметр длины ключа.
2. Злоумышленник получает значение публичного ключа pk.
3. Злоумышленник отправляет несколько сообщений в систему. Она возвращает ему расшифрованные сообщения или отказ в принятии сообщения.
4. Злоумышленник посылает пару открытых текстов m_0, m_1 равной длины в систему, получает шифротекст для одного из них, т.е. $c(pk, m_b)$.
5. Злоумышленник может повторить шаг 3), при этом он не может отправлять сообщение, полученное на шаге 4).
6. Злоумышленник возвращает значение b' - пытается угадать значение b.

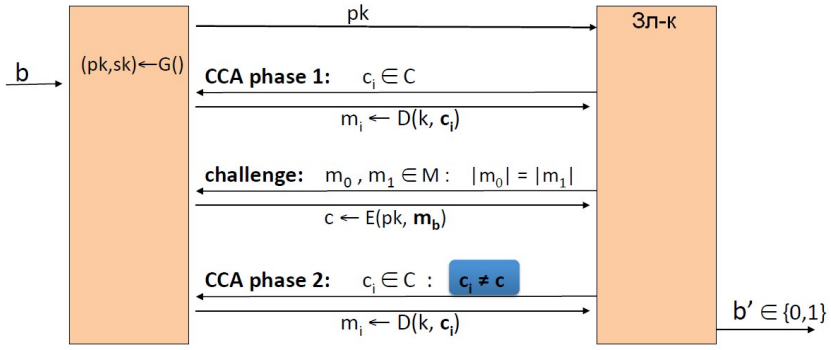

Рис. 26: Атака на асимметричную криптосистему с выбором шифротекста. По материалам [4].

Определение 12.6. $S = (G, E, D)$ *семантически стойкая к атаке с выбранным шифротекстом*, если в эксперименте CCA $\forall A \in$ PP
$Adv_{CCA}(A, S) = \left|P(b' = 1|b = 0) - P(b' = 1|b = 1)\right| < \varepsilon(n)$

Пример 12.3. Пусть мы можем воздействовать на шифротекст так, что контролируемо подменяем часть исходного текста, и что система примет и расшифрует новый шифротекст. Тогда шифр не будет стойким к атаке с выбранным шифротекстом: злоумышленник узнает, чему равно b.

Злоумышленник A отправляет два сообщения: “Alice, 0” и “Alice, 1”. Получает шифротекст c для одного из них. Потом он создает шифротекст c' по принятому шифротексту c так, что в исходном тексте “Alice” меняется на “David”, не изменяя значение бита b. Это новый шифротекст. Система его расшифровывает и сообщает злоумышленнику, чему равно b. См. рис. 27.

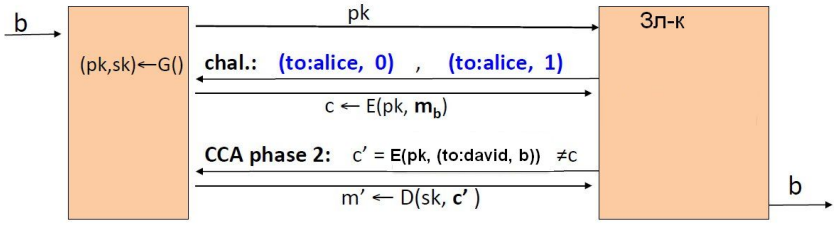

Рис. 27: Уязвимость к атаке с выбранным шифротекстом. По материалам [4].

Сравнение с симметричной криптосистемой.

Для симметричной криптосистемы заверенное шифрование обеспечивает стойкость к атаке с выбранным открытым текстом и целостность шифротекста. Т.е. злоумышленник не может создавать новые шифротексты, которые будут приняты.

В случае асимметричной криптосистемы, злоумышленник может создавать новые шифротексты, которые будут приняты. Поэтому непосредственно требуем стойкость к атаке с выбранным шифротекстом.

12.3 Перестановка RSA

Определение 12.7. Три алгоритма $G(), F(\cdot,\cdot), F'(\cdot,\cdot)$ определяют *перестановку с секретом* (trapdoor permutation), если

$G()$ выдает пары pk, sk - открытый ключ, закрытый ключ.

$F(pk,\cdot) : X \to X$.

$F'(sk, y)$ - это функция $F^{-1}(pk,\cdot)$: если $y = F(pk, x)$, то $F'(sk, y) = x$.

Определение 12.8. Если $\forall$ пары pk, sk функция $F(pk,\cdot) : X \to X$ - односторонняя функция при неизвестном sk, то $G(), F(\cdot,\cdot), F'(\cdot,\cdot)$ определяет *криптостойкую перестановку с секретом* (secure trapdoor permutation).

Сложность вычисления обратной функции для перестановки RSA основана на вычислительно трудной задаче разложения больших чисел на множители. Детерминированный шифр в виде перестановки RSA часто называют “RSA из учебника”.

Опишем создание пары ключей.

1. Выбираем любые p, q - большие простые числа одного порядка, например, 128-битовые. Держим их в секрете.
2. $N = pq$
3. $\varphi(N) = \varphi(p)\varphi(q) = (p-1)(q-1)$ - невозможно быстро вычислить, не зная p, q.
4. Выбираем $\forall e > 1 : \text{НОД}(e, \varphi(N)) = 1$. Число e - “небольшое” и с малым числом единичных битов, т.к. это ускоряет вычисления: можно пользоваться сдвигами. Например, $e = 0x100001$ или $e = 2^{16} + 1$ - простые числа.
5. $d = e^{-1} mod(\varphi(N))$ вычислим с помощью $ExtGCD$.

Число N называется модулем. Число e - открытая экспонента, d - закрытая экспонента. Пара (N, e) - открытый ключ, d - закрытый ключ. Числа p, q больше не нужны, но они секретные.

Опишем шифрование и расшифрование “RSA из учебника”.

Сообщение делится на части, каждой части ставится в соответствие число m, $0 < m < N$. Шифротекст $c = m^e \bmod N$. Расшифрование: $c^d = m^{ed} \bmod N = m^{k\varphi(n)+1} \bmod N = m \bmod N$ по теореме Эйлера для $m \in Z_N^*$.

С вероятностью порядка $1 - 2/min(p,q) \approx 1 - 2^{-log(min(p,q))}$, число m не кратно p или q. Если m не взаимно простое с N? Вероятность этого $\approx 2^{-log(min(p,q))}$ - никогда не случится. Или можно проверять, что m не делится на p, q.

Пример 12.4. $p = 11,\ q = 17,\ N = 187$. e не должно иметь общих делителей с $10 \cdot 16 = 160$. Пусть $e = 9$. Тогда $d = 9^{-1} \bmod 160 = 89$ (использовали $ExtGCD$).

Пусть $m = 688232$. $m_1 = 68$, $m_2 = 82$, $m_3 = 32$. Зашифрование: $c_1 = 68^9 \bmod 187 = 17$. Расшифрование: $17^{89} \bmod 187 = 68 = m_1$.

Замечание. Перестановка RSA не обеспечивает семантическую стойкость шифрования к атаке с выбранным открытым текстом, т.к. шифрование детерминированное.

Является ли перестановка RSA, $m \to m^e \bmod N$, односторонней функцией?

Пусть d не известно; известно $c = x^e$; требуется найти $x = c^{1/e}$ в $\mathbb{Z}_N$, вычислив корень степени e. Лучший известный сейчас алгоритм вычисления корней степени e в $\mathbb{Z}_N$ действует в два шага:

1. разложить N на множители (сложно, субэкспоненциальный алгоритм), $N = pq$.
2. вычислить корни степени e по модулю простых p и q (полиномиальный алгоритм).

Можно ли проще? Не известно.

Сложность взлома (разложения числа на множители) субэкспоненциальная, поэтому асимметричная криптосистема при той же стойкости имеет более длинный ключ и работает медленнее, чем симметричная криптосистема

Примеры использования асимметричной криптосистемы:

1. инициализация сессии, создание общего ключа для симметричного шифра;
2. односторонняя коммуникация, например, e-mail.

12.4 Атаки на перестановку RSA

12.4.1 Атака “встреча посередине”

Типичный сценарий: асимметричное шифрование используется для создания ключа для двусторонней сессии симметричного шифрования. Пусть (e, N) - публичный ключ сервера, d - секретный ключ. Простейший протокол выглядит так:

1. Client $\to$ Server: ClientHello.
2. Server $\to$ Client: ServerHello, (e, N).

3. Client создает ключ симметричного шифра $k \leftarrow K$.
4. Client $\rightarrow$ Server: $c = RSA(e, k)$.
5. Server вычисляет $k = RSA^{-1}(d, c)$.

Длина ключа k для симметричного шифра небольшая. Пусть это 64 бита. Злоумышленник видит шифротекст $c = k^e \in \mathbb{Z}_N$. Если k можно представить в виде $k = k_1 \cdot k_2$, $0 < k_1, k_2 < 2^{34}$ (это произойдет с вероятностью около 0.35 для ключа k длиной 64 бита), то $c/(k_1)^e = (k_2)^e$.

Атака:

1. Построить таблицу: $c/1^e$, $c/2^e$, $c/3^e$, ... $c/(2^{34})^e$ - требуется время $O(2^{34})$ и память $O(2^{34} \cdot 34)$.
2. Для $k_2 = 1, ..., 2^{34}$ попробовать найти $(k_2)^e$ в таблице - требует 2^{34} операций поиска, т.е. время пропорционально $2^{34} \cdot log_2(2^{34})$.

Если нашли значение в таблице, то вернуть пару (k_1, k_2); $k = k_1 \cdot k_2$. Таким образом нашли ключ k, который будет использован в симметричном шифре.

Общее время атаки $\approx 2^{40} \ll 2^{64}$.

Поэтому сейчас используются другие способы создания ключа для симметричного шифра. См., например, RSA-OAEP (далее).

12.4.2 Атака на перестановку RSA с малым d

Большое $d \Rightarrow$ долго расшифровывать. Что будет, если его уменьшить? Существует атака Винера: $d < N^{0.25} \Rightarrow$ можно найти закрытый ключ d по открытому ключу (N, e).

12.4.3 Атака при одинаковой малой публичной экспоненте $e = 3$

В перестановке RSA малая публичная экспонента e обеспечивает быстрое шифрование. Рекомендуется использовать $e = 2^{16} + 1$ - простое число.

Пусть $e = 3$. Пусть у сообщения 3 получателя с разными модулями, но общим значением e: (N_1, e), $(N_2, e), (N_3, e)$. $m \rightarrow c_1, c_2, c_3$.

Обозначим $N = N_1 N_2 N_3$, где N_1, N_2, N_3 взаимно простые. Обозначим $c = m^3 < N$. По КТО, $\exists! c \in \mathbb{Z}_N : \ c = c_i \bmod N_i$, $i = 1, 2, 3$. Найдем его, используя КТО. $\Rightarrow m = c^{1/3}$ как целые числа. Корень в $\mathbb{Z}$ (не в $\mathbb{Z}_n^*$) вычисляется полиномиальным алгоритмом.

Защита от этой атаки:

- для атаки нужно e получателей одного сообщения. Не использовать $e = 3$.
- рандомизация шифрования (см., например, криптосиситему PKCS 1)

12.4.4 Атака на перестановку RSA с общим модулем

Пусть есть i ключей с общим модулем: (N, e_i, d_i). Пусть e_i взаимно простые. Злоумышленник видит два шифротекста $c_1 = m^{e_1} \mod N$; $c_2 = m^{e_2} \mod N$.

НОД$(e_1, e_2) = 1$, поэтому $ExtGCD(e_1, e_2) \Rightarrow e'_1, e'_2 : \ e_1 e'_1 + e_2 e'_2 = 1$.

Очевидно, одно из чисел e'_1, e'_2 отрицательное. Пусть это e'_1. Тогда вычислим $c_1^{-1} \mod N$, используя $ExtGCD$. Далее, $(c_1^{-1})^{-e'_1} \cdot c_2^{e'_2} \equiv m^{e_1 e'_1 + e_2 e'_2} \equiv m \mod N$. Нашли исходное сообщение.

Защита от этой атаки: не использовать общий модуль.

12.4.5 Атака на перестановку RSA: проблема малой энтропии

Пусть алгоритм создания параметров шифра выглядит следующим образом:

```
prng.seed(seed)
p = prng.generate_random_prime()
prng.add_randomness(bits)
q = prng.generate_random_prime()
N = p*q
```

Малая начальная энтропия ГПСЧ $\Rightarrow$ одинаковые значения p. Тогда для N_1, N_2 из разных открытых ключей НОД$(N_1, N_2) = p$. Так факторизуется 0.4% от взятых в сети публичных ключей.

12.4.6 Атака на перестановку RSA по побочным каналам

Используем график потребляемой мощности при вычислении $c^d \mod N$. Или воздействуем на микросхему ЭМИ $\Rightarrow$ воспользуется ошибкой при вычислении $c^d \mod N$.

Так можно узнать секретный ключ d.

12.5 Шифр Эль-Гамаля

Стойкость шифра Эль-Гамаля (El Gamal) основана на сложности задачи Диффи-Хеллмана. А именно, пусть группа $G = \langle g \rangle$, a, b - случайные целые числа и известны

g^a, g^b. Тогда g^{ab} нельзя отличить от случайного элемента группы G.

Опишем шифр Эль-Гамаля.

1. Создание ключей:

- выбираем группу G порядка q. Обычно G - циклическая подгруппа некоторой группы $\mathbb{Z}_p^*$ (см. далее).
- выбираем элемент g - генератор G.
- выбираем $x \in \mathrm{N}$, $x \xleftarrow{R} \{1, ..., q\}$.
- вычисляем $y = g^x$
- открытый ключ: (G, q, g, y); закрытый ключ: x

2. Шифрование: все операции в группе G.

- сообщение $m \in G$.
- выбираем случайное число k, $1 < k < q - 1$. Это аналог IV, но k не передается.
- $c_1 = g^k$
- $c_2 = y^k m$
- (c_1, c_2) - шифротекст, в 2 раза длиннее m.

3. Расшифрование: $m = c_2 \cdot c_1^{-x}$.

4. Корректность шифра: $c_2 * c_1^{-x} = y^k m \cdot c_1^{-x} = g^{kx} m g^{-kx} = m$.

Утверждение 12.1. Если в группе G задача Диффи-Хеллмана - сложная и алгоритм шифрования не разглашает информацию об m, то шифр Эль-Гамаля семантически стойкий к атаке с выбором открытого текста.

Без доказательства. Любознательный читатель может найти доказательство в статье [16].

Следствие. Шифр Эль-Гамаля на подгруппе $m \in \langle g \rangle = \{a \in Z_p^* | \left(\frac{a}{p}\right) = 1\}$, p - простое, является стойким к атаке с выбранным открытым текстом.

Утверждение 12.2. Шифр Эль-Гамаля не является стойким к атаке с выбором шифротекста.

Доказательство. Пусть $E(m) = (c_1, c_2)$ и $\forall t \in G$. Тогда, по определению шифра Эль-Гамаля, $E(tm) = (c_1, tc_2)$ при том же значении случайного параметра k. Т.е. по шифротексту для сообщения m злоумышленник может создать новый верный шифротекст для сообщения tm и использовать его для получения преимущества в эксперименте CCA. А именно, отправить сообщения $m_0 \neq m_1$, получить шифротекст $c = E(m_i)$, $i \in \{0, 1\}$. Затем отправить шифротекст tc и получить сообщение tm_i. $\square$

Утверждение 12.3. Произвольная группа $\mathbb{Z}_p^*$ и ее генератор g не подходят для использования в шифре Эль-Гамаля, т.к. шифротекст раскрывает 1 бит информации о сообщении m.

Доказательство. Пусть g - генератор $\mathbb{Z}_p^*$. $\Rightarrow g \in QNR(Z_p^*)$, $\left(\frac{g}{p}\right) = -1$.

Символ Лежандра вычисляется эффективно. Вычислим $\left(\frac{c_1}{p}\right) = \left(\frac{g^k}{p}\right) = \left(\frac{g}{p}\right)^k$ $\Rightarrow$ узнаем четность k. Публичный ключ $y = g^x \Rightarrow$ узнаем четность x.

$c_2 = g^{kx}m$, вычислим $l_1 = \left(\frac{c_2}{p}\right)$. Из предыдущих вычислений, знаем $l_2 = \left(\frac{g^{kx}}{p}\right)$. Тогда узнаем $\left(\frac{m}{p}\right) = l_1/l_2$. □

12.6 Гибридная криптосистема

Пусть сервер имеет пару ключей асимметричной криптосистемы (sk, pk). Клиент хочет с ним связаться и передавать данные, зашифрованные симметричным шифром, т.к. такой шифр быстрее. Клиент шифрует сессионный ключ с помощью асимметричного шифра, а сами данные - симметричным шифром.

- пусть (G, F, F^{-1}) - односторонняя функция с секретом, $X \to Y$.
- пусть $H : X \to K$ - криптографическая хэш функция
- пусть (E_s, D_s) - симметричный шифр, обеспечивающий заверенное шифрование.

1. Шифрование.
   - $x \xleftarrow{R} X$
   - $y = F(pk, x)$
   - $c = E_s(H(x), m)$
   - шифротекст - это сообщение $(y, c) = F(pk, x)||E_s(H(x), m)$
2. Расшифрование.
   - $x = F^{-1}(sk, y)$
   - исходный текст $m = D_s(H(x), c)$

Защита от атаки “встреча посередине” при использовании перестановки RSA: $|\{H(x)\}|^{1/2}$ должно быть достаточно велико. Использование хэщ функции H обеспечивает нужный размер результата и запутывание.

Определение 12.9. *Модель случайного оракула* (random oracle model) для хэш функции заключается в следующем. Пусть хэш функция вычисляется случайным черным ящиком, так что результат вычисления хэш функции от любого аргумента нельзя

предсказать до ее вычисления, но повторное вычисление хэш функции от того же аргумента дает тот же самый результат.

Теорема 12.1. Если (G, F, F') - односторонняя функция с секретом, симметричный шифр (E_S, D_S) обеспечивает заверенное шифрование, хэш функция $H : X \to K$ - "случайный оракул", то эта гибридная криптосистема - стойкая к атаке с известным шифротекстом (в модели со случайным оракулом).

Без доказательства. (TODO: поставить ссылку.)

На практике эта криптосистема не используется, т.к. здесь с каждым сообщением передается ключ, зашифрованный детерминированной односторонней функцией с секретом. Недостатки:

- ключ симметричного шифра многоразовый, его не нужно передавать с каждым сообщением;
- ключ шифруется детерминированно;
- при использовании перестановки RSA для части передаваемых ключей проходит атака типа "'встреча посередине".

На практике сначала согласуется первичный ключ для симметричного шифра, потом используется заверенное симметричное шифрование с вычисленными по нему сессионными ключами.

12.7 Шифр RSA-OAEP

RSA-OAEP (RSA optimal asymmetric encryption padding) - рандомизированный асимметричный шифр. Пусть G, H - криптографические хэш функции (на практике - SHA-256), m - сообщение, n - длина модуля RSA, sk, pk - ключи перестановки RSA, k_0, k_1 - фиксированные константы. Опишем шифр RSA-OAEP.

1. Шифрование.

- $len(m) \leq n - k_0 - k_1$; $m' = m||pad$, где $pad = 01000..0$, $len(pad) \geq k_1$.
- $r \xleftarrow{R} \{0,1\}^{k_0}$
- $s = m' \oplus G(r)$
- $t = r \oplus H(s)$
- $c' = s||t$
- $c = RSA(pk, c')$ - шифротекст.

2. Расшифрование. Алгоритм очевиден (исходя из корректности шифра). Если после расшифрования *pad* не верен, сообщение не принимается.

Теорема 12.2. Если перестановка RSA - односторонняя функция с секретом, G, H - случайные оракулы, то шифр RSA-OAEP стойкий к атаке с выбранным шифротекстом.

Без доказательства. (TODO: поставить ссылку.) Заметим, что используется рандомизированное шифрование.

Этот шифр может использоваться как для асинхронной, однонаправленной передачи данных, и так и для передачи первичного ключа симметричного шифра. Его реализации могут быть уязвимы к атакам по побочным каналам. Например, в реальном протоколе отвергнуть расшифрованное сообщение можно не только при неверном *pad*, но и при некоторых других условиях. В этих случаях время работы протокола должно быть одинаковым.

12.8 Сравнение длин ключей симметричных и асимметричных криптосистем

Асимметричные криптосистемы основаны на задачах, для которых существуют субэкспоненциальные алгоритмы решения. Современные симметричные криптосистемы имеют только экспоненциальные алгоритмы поиска ключа. Поэтому при одинаковой стойкости ко взлому их длины ключей и, как следствие, скорость работы существенно отличаются (см. табл. 7).

Симм. шифр	Размер модуля RSA	Порядок группы точек эллиптич. кривой
64	512	-
80	1024	160
128	3072	256
256 (AES)	15360	512

Таблица 7: Длины ключей различных криптосистем, обеспечивающие равную стойкость ко взлому. По материалам [4].

Криптосистемы на группе точек эллиптической кривой не имеют известных субэкспоненциальных алгоритмов взлома, поэтому имеют меньшую длину ключа и,

вследствие этого, работают быстрее классических асимметричных аналогов. По этой причине происходит постепенный переход к асимметричным криптосистемам на эллиптических кривых. Эти криптосистемы не вошли в данный курс. Любознательный читатель может ознакомиться с ними по книгам.

13 Электронно-цифровая подпись

Основная цель раздела: описать ЭЦП и условия ее стойкости к созданию подделки, а также исследовать некоторые ее реализации.

13.1 Общие сведения

Определение 13.1. Система электронно-цифровой подписи (ЭЦП) (digital signature) - это набор алгоритмов $G(), S(\cdot,\cdot), V(\cdot,\cdot,\cdot)$:

- $G()$ создает пару ключей: публичный ключ, секретный ключ - (pk, sk)
- подпись сообщения $\sigma = S(sk, m)$
- результат проверки подписи $v = V(pk, m, s) \in \{0, 1\}$
- $\forall (pk, sk)\ V(pk, m, S(sk, m)) = 1$

Публичный ключ pk известен всем, поэтому *каждый* может проверить подпись сообщения.

Код аутентификации и ЭЦП похожи, но имеют важные различия.

Код аутентификации:

- обеспечивает целостность сообщения при передаче
- проверяется приватно
- могут создать обе стороны

ЭЦП:

- обеспечивает целостность сообщения при передаче
- проверяется любым желающим
- может создать только владелец секретного ключа
- обеспечивает невозможность отказа от авторства документа

Для наглядности, сравнение кода аутентификации и ЭЦП приведено в табл. 8.

Опишем эксперимент SIG - попытку создания поддельной подписи.

Код аутентификации	ЭЦП
A,B знают k_{AB} A создает $t_{AB} = MAC(k_{AB}, m)$ B проверяет целостность m	A знает (pk_A, sk_A); B знает pk_A A создает $\sigma_A = S(sk_A, m)$ B проверяет целостность m
$\forall m'$ B может создать $MAC(k_{AB}, m)$	$\forall m'$ B не может создать $S(sk_A, m')$
отправка m двум получателям: $t_{AB} := MAC(k_{AB}, m)$ $t_{AC} := MAC(k_{AC}, m)$	отправка m двум получателям: $\sigma_A := S(sk_A, m)$
	Арбитр тоже знает pk_A B может предъявить арбитру пару (s, m)

Таблица 8: Сравнение кода аутентификации и ЭЦП.

1. $(pk, sk) = G(n)$, где n - параметр длины ключа.
2. Система сообщает pk злоумышленнику A и обеспечивает ему доступ к черному ящику $S(sk, \cdot)$, который создает подписи любых сообщений.
3. A получает подписи сообщений $m_1, ..., m_q$ по своему выбору.
4. A выбирает $m \neq m_i$, самостоятельно создает $\sigma(m)$.
5. $b = V(pk, m, \sigma)$ - результат проверки подписи системой.

Определение 13.2. Алгоритм ЭЦП (G, S, V) называется *стойким к созданию поддельной подписи* (existentially unforgeable), если $\forall A \in$ BPP в эксперименте SIG $P(b = 1) < \varepsilon(n)$, где n - параметр длины ключа.

На практике используется конкретное значение ε, например 2^{-80}.

13.2 Подпись “RSA из учебника”

Алгоритм подписи:

1. Создание ключей: $(N, e, d) = G()$. e - секретный ключ, (N, d) - открытый ключ.
2. Сообщение $m \in \mathbb{Z}_N^*$.
3. Подпись $\sigma = m^e \in \mathbb{Z}_N^*$.
4. Проверка $V(d, m, \sigma) : \ m \stackrel{?}{=} \sigma^d \in \mathbb{Z}_N^*$.

Корректность подписи очевидна.

Уязвимости подписи “RSA из учебника”.

1. Возможно создание новой верной пары (сообщение, подпись), если значение сообщения не фиксировано, создается вместе с подписью. Пусть злоумышленник знает (N, d). Он возьмет $\forall x \in \mathbb{Z}_n$. Тогда $m = x^d \bmod N$. (m, x) - верная пара (сообщение, подпись), т.к. $x^d \equiv m \bmod N$. $\Rightarrow$ подпись "RSA из учебника" не стойкая к созданию подделки и не должна использоваться.
2. Коммутативное свойство подписей. Если $m = m_1 \cdot m_2$ и известны подписи σ_1, σ_2 для m_1, m_2, то $\sigma(m) = \sigma_1 \cdot \sigma_2 \bmod N$ Проверка: $\sigma^d = (\sigma_1\sigma_2)^d = m_1^{de} m_2^{de} = m_1 m_2 = m \bmod N$. $\Rightarrow$ если стали известны подписи для t сообщений, можно создать подписи для $2^t - t$ новых сообщений.

13.3 Подпись "хэшированная RSA"

Алгоритм подписи:

1. Создание ключей: $(N, e, d) = G()$. e - секретный ключ, (N, d) - открытый ключ.
2. Пусть $H : \{0,1\}^* \to \mathbb{Z}_N^*$ - криптостойкая хэш функция. Подпись $\sigma = (H(m))^e \bmod N$
3. Проверка $V(d, m, \sigma) : \ \sigma^d \bmod N \stackrel{?}{=} H(m)$.

Корректность подписи очевидна.

Теорема 13.1. Подпись "хэшированная RSA" стойкая к созданию поддельной подписи в модели со случайным оракулом.

Без доказательства. (TODO: поставить ссылку.)

13.4 ЭЦП по Эль-Гамалю

Пусть H - криптостойкая хэш функция, p - большое простое число, g - $\forall$ генератор группы $\mathbb{Z}_p^*$, сообщение m - любой элемент $\mathbb{Z}_p^*$. Используется тот факт, что вычислить дискретный логарифм - сложная задача.

1. Создание ключа.

- $x \stackrel{R}{\leftarrow} \{1, ..., p-1\}$.
- $y = g^x \bmod p$.
- (p, g, y) - открытый ключ, x - секретный ключ.

2. Создание подписи.

- $k \stackrel{R}{\leftarrow} \mathbb{Z}_{p-1}^*$. Значение k - случайное, секретное, никогда не повторяется для данного набора (p, g, y, x), иначе можно узнать x.

- $r = g^k \bmod p$.
- $s = (H(m) - xr)k^{-1} \bmod (p-1)$.
- Если $s = 0$, выбрать другое k.
- $(r, s) = (g^k \bmod p, s)$ - подпись для m.

3. Проверка подписи.

- $0 < r < p$.
- $0 < s < p - 1$.
- $g^{H(m)} \stackrel{?}{=} y^r r^s \bmod p$.

4. Корректность алгоритма.

- $H(m) \equiv xr + sk \bmod (p-1)$.
- $y^r r^s \bmod p \equiv g^{xr} g^{ks} \equiv g^{H(m)} \bmod p$ - верно.

Утверждение 13.1. Без использования хэш функции ЭЦП по Эль-Гамалю не стойкая к созданию новой верной подписи.

Доказательство. По условию, алгоритм $s(m) = (\mathbf{m} - xr)k^{-1} \bmod (p-1)$. Создадим новую верную пару сообщение, подпись.

Цель: $y^r r^s \equiv g^m \bmod p$. Пусть $k \stackrel{R}{\leftarrow} \mathbb{Z}^*_{p-1}$. Выберем одновременно r, s, m. Избавимся от y^r в левой части: хотим $r^s \equiv y^{-r} z \bmod p$ для некоторого z. Пусть $r = yg^k \bmod p$, $s = -r \bmod p - 1$. Тогда $y^r r^s \equiv y^r (yg^k)^{-r} \equiv g^{-kr} \bmod p - 1$.

Выберем сообщение $m \equiv -kr \bmod p - 1$. Тогда пара $(r, -r \bmod p - 1)$ - верная подпись для $m \equiv -kr \bmod (p-1)$, где $r \equiv yg^k \bmod p$, для произвольного фиксированного $k \in \mathbb{Z}^*_{p-1}$. □

Теорема 13.2 (О стойкости ЭЦП по Эль-Гамалю)**.** Если злоумышленник может создать новую подпись по Эль-Гамалю в модели со случайным оракулом с не пренебрежимо малой вероятностью, то задача дискретного логарифма может быть решена за полиномиальное время.

Без доказательства. (TODO: поставить ссылку.)

Замечание. Таким образом, свели решение задачи дискретного логарифма к атаке на ЭЦП. Считается, что дискретный логарифм - трудная задача.

Пример 13.1. Создадим и проверим подпись по Эль-Гамалю.
Пусть $p = 11, g = 2, x = 8$. $y = g^x \bmod p = 2^8 \bmod 11 = 3$. Открытый ключ: $(11, 2, 3)$. Подпишем сообщение m такое, что $H(m) = 5$.

Выберем случайное $k = 9$, НОД$(9, 11 - 1) = 1$.

$a = g^k \bmod p = 2^9 \bmod 11 = 6$

$5 = (8 * 6 + 9 * b) \bmod 10 \ \Rightarrow b = 3 \ \ (ExtGCD)$

Подпись: пара $a = 6, b = 3$.

Проверка: $y^a a^b \bmod \ p \stackrel{?}{=} g^{H(m)} \bmod \ p$, т.е.

$3^6 \cdot 6^3 \bmod 11 \stackrel{?}{=} 2^5 \bmod 11$ - верно.

13.5 ЭЦП DSA

DSA - стандарт цифровой подписи США. (Стандарты FIPS 186, 186-1, 186-2, 186-3 (2009)) В ее основе - подпись по Эль-Гамалю.

Утверждение 13.2. Пусть p, q - простые, $p - 1 = qn, \ n \in \mathrm{N}, \ g^q \equiv 1 \bmod p$. Тогда $g^{x+y} \bmod p = g^{(x+y) \bmod q} \bmod p$, $g^{x \cdot y} \bmod p = g^{x \cdot y \bmod q} \bmod p$

Доказательство. $g^{x+y} \bmod p = g^{(x+y) \bmod q + qn} \bmod p \ \stackrel{g^q = 1}{=} \ g^{(x+y) \bmod q} \bmod p$. Аналогично для произведения. □

1. Создание ключа.

- p - простое число длины L бит (от 512 до 3072),
- q - простое число длины N бит (от 160 до 256), делитель $p - 1$.
- Для создания ключа нужен $\forall$ элемент $g \in \mathbb{Z}_p^*$ порядка q. Найдем его. Выберем $\forall h \in \mathbb{Z}_p^*$. Если $h^{(p-1)/q} \bmod p \neq 1$, тогда $g = h^{(p-1)/q} \bmod p$ имеет порядок q. Иначе - взять другой элемент h.
- $x \stackrel{R}{\leftarrow} \{1, ..., q - 1\}$ - секретный ключ,
- $y \equiv g^x \bmod \ p$. (p, q, g, y) - открытый ключ.

2. Создание подписи.

- $k \stackrel{R}{\leftarrow} \{1, ..., q - 1\}$
- $r \equiv (g^k \bmod p) \bmod q$
- $s \equiv (H(m) + xr)k^{-1} \bmod q$
- Подпись сообщения - это пара (r, s).

Значение k - случайное, секретное, не повторяется для данной пары ключей, иначе можно узнать секретный ключ x.

3. Проверка подписи.

- $u_1 \equiv (H(m) \cdot s^{-1}) \bmod q$
- $u_2 \equiv (rs^{-1}) \bmod q$

- $v \equiv (g^{u_1} \cdot y^{u_2} \bmod p) \bmod q$.
- $v \stackrel{?}{=} r$

4. Корректность алгоритма.

- $k \equiv (H(m) + xr)s^{-1} \bmod q$
- Т.к. порядок элемента $g \in Z_p^*$ равен q, то
- $g^k \equiv g^{H(m)s^{-1}} g^{xrs^{-1}} \equiv g^{H(m)s^{-1}} y^{rs^{-1}} \equiv g^{u_1} y^{u_2} \bmod p$
- $r \equiv (g^k \bmod p) \bmod q \equiv g^{u_1} y^{u_2} \bmod p \bmod q \equiv v$ - верно.

Задача 13.1. Пусть значение k известно и фиксировано. Найти x.

Теорема 13.3 (О стойкости подписи DSA к созданию подделки.)**.** Если злоумышленник может создать новую верную подпись DSA в модели со случайным оракулом с не пренебрежимо малой вероятностью, то задача дискретного логарифма в группе $\mathbb{Z}_p^*$ может быть решена за полиномиальное время.

Без доказательства. (TODO: поставить ссылку.)

Следствие. Необходимые условия стойкости подписи DSA и Эль-Гамаля:

- k - выбирается случайно, равномерно, не повторяется, держится в секрете
- H - криптостойкая хэш функция
- задача дискретного логарифма в группе $\mathbb{Z}_p^*$ не имеет полиномиального алгоритма решения.

Часть IV

Управление ключами

14 Протокол Нидхэма-Шредера

Основная цель раздела: описать протокол создания общих ключей пользователей для симметричных криптосистем. Ключ создается для пары пользователей только по запросу одного из них. В протоколе участвует доверенный центр.

Пусть n пользователей хотят общаться каждый с каждым, используя симметричный шифр. Тривиальное решение - создать постоянный ключ для каждой пары пользователей, всего $n(n-1)/2$ ключей. Квадратичный рост числа ключей, добавление пользователей затруднено.

Рассмотрим протокол создания временного общего ключа по требованию для двух пользователей с участием доверенного центра, вариант для симметричной криптосистемы.

Обозначим:

- A, B - Алиса и Боб; их уникальные имена известны всем.
- K_{XY} - симметричный ключ, известный только X и Y.
- N_A, N_B - nonce, генерируемые A и B.
- $\{M\}_{K_{XY}}$ - сообщение M, зашифрованное ключом K_{XY}. Используется шифр, обеспечивающий заверенное шифрование.
- $X \to Y : M$ - передача сообщения M от X к Y.

Пусть S - доверенный сервер. При создании пользователей A и B S выдает им ключи для связи с собой: K_{AS}, K_{BS}. Позже Алиса инициирует коммуникацию с Бобом, хочет иметь защищенный сеанс связи. Боб хочет быть при этом уверен, что к нему обратилась именно Алиса.

Протокол Нидхема-Шредера, первая версия:

1. $A \to S : \{A, B, N_A\}$ - открытый текст: “Я A, хочу связаться с B”.
2. $S \to A : \{N_A, K_{AB}, B, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$. S создает сессионный ключ для A, B, упаковывает его и отправляет A.
3. $A \to B : \{K_{AB}, A\}_{K_{BS}}$. A пересылает зашифрованный ключ K_{AB} к B, который (и только он) может расшифровать это сообщение. Никто, кроме S, не может зашифровать сообщение ключом K_{BS} так, чтобы B его принял. Т.о. B удостоверяется, что A получила этот ключ от S.
4. $B \to A : \{N_B\}_{K_{AB}}$. B стремится проверить, знает ли A ключ K_{AB}.
5. $A \to B : \{N_B+1\}_{K_{AB}}$. A производит фиксированную операцию с N_B, показывая B, что она все еще в сети и помнит этот сессионный ключ.

Этот протокол уязвим к атаке воспроизведения: если сессионный ключ K_{AB} стал известен злоумышленнику E, и он запомнил сообщение $\{K_{AB}, A\}_{K_{BS}}$, E начинает коммуникацию с шага 3, выдавая себя за A:

$E \to B : \{K_{AB}, A\}_{K_{BS}}$.

B примет это сообщение, т.о. установив сеанс связи с E (а не A) с ключом K_{AB}.

Защитимся от атаки воспроизведения, добавив и изменив несколько сообщений.

00. $A \to B : A$ - A связывается с S: “Я A, хочу связаться с тобой.”

0. $B \to A : \{A, \mathbf{N'_B}\}_{K_{BS}}$. B отвечает nonce, зашифрованным своим ключом K_{BS}.

1'. $A \to S : \{A, B, N_A, \{A, \mathbf{N'_B}\}_{K_{BS}}\}$. A связывается с S: “Я A, хочу связаться с B”. Отправляет этот открытый текст и сообщение от B.

2'. $S \to A : \{N_A, K_{AB}, B, \{K_{AB}, A, \mathbf{N'_B}\}_{K_{BS}}\}_{K_{AS}}$. S создает сессионный ключ K_{AB}.

3'. $A \to B : \{K_{AB}, A, \mathbf{N'_B}\}_{K_{BS}}$. Добавление nonce $\mathbf{N'_B}$ не дает возможность E переслать повторно это сообщение с скомпрометированным ключом K_{AB}: это значение N'_B уже использовано и не ожидается. Данные зашифрованы шифром с заверенным шифротекстом с ключом K_{BS}, поэтому E не может создать новое сообщение, которое будет принято B.

Следующие шаги без изменений:

4. $B \to A : \{N_B\}_{K_{AB}}$.

5. $A \to B : \{N_B + 1\}_{K_{AB}}$.

15 Протокол Kerberos

Основная цель раздела: описать протокол Kerberos контроля доступа к ресурсам в гетерогеных сетях.

15.1 Описание протокола

Kerberos - это протокол контроля доступа к ресурсам в гетерогенных сетях. В его основе - протокол Нидхема-Шредера с несколькими существенными изменениями:

- уменьшено количество сообщений между клиентом и сервером аутентификации.
- введен TGT (Ticket Granting Ticket, билет для получения билета) - концепция, позволяющая пользователям авторизоваться для доступа на несколько сервисов один раз.
- вместо *nonce* используется метка времени.

Центральные серверы Kerberos:

- сервер аутентификации (AS) и

- сервер выдачи билетов (TGS - Ticket Granting Server).

Цели создания Kerberos:

- оптимизировать использование пароля пользователя. Пусть программа, запущенная от моего имени, авторизуется для доступа к сервисам системы. Каждая такая программа будет хранить мой пароль и являться точкой для атаки. Система Kerberos позволяет программе не хранить мой пароль. Т.о. исключается одна из основных возможностей утечки пароля пользователя.
- обеспечить защиту от подслушивания, модификации сообщения и атаки воспроизведения.
- от других атак стандарт Kerberos не защищает.

Пример 15.1. Использование системы Kerberos: Windows 2000 и более поздние версии, а также все версии ОС Unix и гетерогенные сети.

Обозначения:

AS - сервер авторизации (authentication server)

TGS - сервер выдачи билетов (ticket granting server)

C - программа-клиент (client)

S - сервис системы (service provider)

A - сетевой адрес клиента (client's network address)

V - срок действия билета (validity period for a ticket)

t - текущее время (timestamp)

K_X - постоянный секретный ключ X (т.е. $K_{X,AS}$)

$T_{X,Y}$ - билет, содержащий информацию о том, что клиент X может пользоваться сервисом Y

$A_{X,Y}$ - токен авторизации клиента X для сервиса Y

$K_{X,Y}$ - сессионный ключ для X,Y

$\{m\}K_X$ - сообщение m, зашифрованное ключом K_X

Билет $T_{X,Y} = Y, \{X, A, V, t, K_{X,Y}\}K_Y$

Его создает TGS или AS, с новой меткой времени t.

Токен авторизации X для Y: $A_{X,Y} = \{X, t, [key]\}K_{X,Y}$

Его создает X, с новой меткой времени t. key - сессионный ключ для X, Y, опциональное поле.

Описание протокола (см. рис. 28):

1. $C \to AS : \{C, TGS, t\}$

   Запрос к AS на билет для доступа к TGS. Открытый текст. AS проверяет, что время клиента верное ($\pm$ 5 минут).

2. $AS \to C : \{T_{C,TGS}\}K_{TGS}, \{K_{C,TGS}\}K_C$

   AS возвращает билет (TGT) для доступа к TGS. Время жизни TGT 8-10 часов. Потом надо получить новый.

3. $C \to TGS : \{A_{C,S}\}K_{C,TGS}, \{T_{C,TGS}\}K_{TGS}$

   Запрос билета для связи с сервером S.

4. $TGS \to C : \{K_{C,S}, S, V\}K_{C,TGS}, \{T_{C,S}\}K_S$

   Возвращает Server Ticket $T_{C,S}$, который имеет срок действия, и сессионный ключ $K_{C,S}$.

5. $C \to S : \{A'_{C,S}\}K_{C,S}, \{T_{C,S}\}K_S$

   Предъявляет Server Ticket; шифрует метку времени $\Rightarrow$ знает ключ.

6. $S \to C : \{t_{A'} + 1\}K_{C,S}$

   Возвращает инкрементированную метку времени из $A'_{C,S} \Rightarrow$ знает ключ.

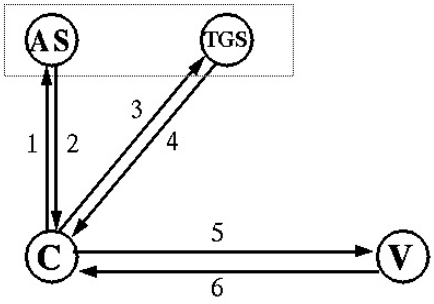

Рис. 28: Протокол контроля доступа Kerberos.

Чтобы связаться с другим сервером S', C повторит шаги 3-6. Шаги 1-2 надо повторить (получить новый билет для доступа к TGS) по истечении срока действия TGT.

Проверка метки времени подразумевает, что часы всех машин в сети синхронизованы в пределах нескольких минут. Если отличие метки времени от текущего времени выходит за эти пределы, запрос отвергается как попытка атаки воспроизведения.

TGS также хранит базу токенов авторизации, чтобы исключить атаки воспроизведения со все еще действительной меткой времени. Повторный запрос с тем же токеном авторизации и меткой времени будет проигнорирован.

C и S могут зашифровать свою сессию ключом $K_{C,S}$ или ключом key из токена авторизации $A_{c,s}$. (Разные версии протокола.) Оба этих ключа известны только им, т.е. их использование является свидетельством авторства сообщения.

Уязвимости протокола Kerberos:

- чтобы исключить атаку воспроизведения в течение временного окна после метки времени, каждый сервер должен хранить все действительные билеты своих клиентов. Если их много, то это проблема.
- если хосту может быть передано по сети неправильное время, можно воспроизвести старые сообщения. Т.е. требуется безопасный сетевой протокол точного времени.

Опциональное улучшение протокола:

- авторизация клиента перед AS с использованием ЭЦП сообщения и сертификата пользователя с использованием асимметричной криптосистемы.

15.2 Домены Kerberos

Домены (realms) Kerberos нужны в большой распределенной сети. Например, если компания имеет подразделения в разных странах.

Опишем протокол получения доступа к ресурсу в другом домене. Клиент должен знать, в каком домене находится этот ресурс. Основное отличие в том, что клиенту нужно дополнительно получить билет для доступа к TGS другого домена.

Обозначения: $r()$ - домен. K_{c-r} - междоменный ключ. TGS2 - сервер выдачи билетов другого домена.

Шаги 1-2 протокола Kerberos (получение TGT) - без изменений. Далее:

3. $C \to TGS : \{A_{C,TGS2}\}K_{C,TGS}, \{T_{C,TGS}\}K_{TGS}$

3'. $TGS \to C : \{K_{C,TGS2}, S, V\}K_{C,TGS}, \{T_{C,TGS2}, r(TGS)\}K_{c-r}$ - получение билета для доступа к TGS2

3". $C \to TGS2 : \{A'_{C,S}\}K_{C,TGS2}, \{T_{C,TGS2}, r(TGS)\}K_{c-r}$

4. $TGS2 \to C : \{K_{C,S}, S, V\}K_{C,TGS2}, \{T_{C,S}\}K_S$ - получение билета для доступа к ресурсу S происходит так же, как внутри одного домена.

Шаги 5-6 без изменений.

В протоколе Kerberos V4 каждые два домена имеют ключ для связи друг с другом $\Rightarrow$ квадратичный рост числа ключей. В протоколе Kerberos V5 - древовидная структура доменов. Например, следующие пары доменов могут иметь междоменные

ключи:

graphics.cs.mit.edu $\leftrightarrow$ cs.mit.edu

parallel.cs.mit.edu $\leftrightarrow$ cs.mit.edu

cs.mit.edu $\leftrightarrow$ mit.edu

mit.edu $\leftrightarrow$ biology.mit.edu

mit.edu $\leftrightarrow$ chemistry.mit.edu

В билете записывается путь, пройденный по дереву. Он проверяется на входе в домен.

16 Протокол Диффи-Хеллмана

Основная цель раздела: описать протокол создания общего секретного ключа для двух пользователей без участия доверенного центра.

16.1 Описание протокола

Цель протокола Диффи-Хеллмана - создать сессионный ключ симметричного шифра без участия доверенного центра.

Пусть p - большое простое число, $g \in \mathbb{Z}_p^*$ - порождающий элемент (генератор) группы. Цель - создать общий ключ для пользователей A, B - элемент группы $\mathbb{Z}_p^*$.

Задача DHP: даны группа $\mathbb{Z}_p^*$ и $g,\ g^a,\ g^b \in \mathbb{Z}_p^*$. Чему равен g^{ab}?

Гипотеза Диффи-Хеллмана: задача DHP не имеет полиномиального алгоритма решения. Не опровергнута за несколько десятилетий. Для групп общего вида доказано, что задача DHP так же сложна, как задача поиска дискретного логарифма (найти x по g^x). Лучший известный алгоритм решения - субэкспоненциальный.

Пусть A,B - два пользователя. Протокол Диффи-Хеллмана (DH):

1. A выбирает $a \xleftarrow{R} \mathbb{Z}_p^*$
2. $A \to B : g^a$, "A''
3. B выбирает $b \xleftarrow{R} \mathbb{Z}_p^*$
4. $B \to A : g^b$, "B''

В результате A, B имеют общий ключ k_{AB}: $(g^b)^a = g^{ab} = k_{AB} = (g^a)^b$.

Если злоумышленник E может разрывать канал связи между A и B, пропускать все данные через себя и изменять их произвольным образом, это - атака посредника (man in the middle attack). Протокол DH уязвим к атаке посредника, см. табл. 9. В результате A и E устанавливают соединение с одним ключом, E и B - с

A		E		B
$a \xleftarrow{R} \mathbb{Z}_p^*$	$\xrightarrow{g^a}$			
		$v \xleftarrow{R} \mathbb{Z}_p^*$	$\xrightarrow{g^v}$	
			$\xleftarrow{g^b}$	$b \xleftarrow{R} \mathbb{Z}_p^*$
	$\xleftarrow{g^w}$	$w \xleftarrow{R} \mathbb{Z}_p^*$		
g^{aw}	$\leftrightarrow$	$g^{aw} \mid g^{bv}$	$\leftrightarrow$	g^{bv}

Таблица 9: Атака посредника на протокол Диффи-Хеллмана.

другим. При этом A и B ничего не знают о существовании E.

Поэтому протокол DH на практике, как правило, "обернут" в какой-либо другой сетевой протокол, обеспечивающий защиту от атаки посредника.

Вариант системы - хранить значения g^a, g^b в "телефонном справочнике" - тоже уязвим к атаке посредника. Если же сообщения заверены ЭЦП, которую можно проверить независимо от посредника, то атака посредника не проходит.

16.2 Использование порождающего элемента группы

Обоснуем необходимость использования в протоколе Диффи-Хеллмана порождающего элемента группы.

Пусть p большое, но $|\langle g \rangle| = 10^6$. Тогда ключ g^{ab} - один из 10^6 элементов. Его легко найти перебором.

Кто выбирает p и g? Эти параметры общие для всех абонентов сети, в которой работает протокол DH. Поэтому каждый из пользователей должен проверить, что p - простое и g - порождающий элемент Z_p^*.

В $\mathbb{Z}_p^*$ $\varphi(p-1)$ порождающих элементов. Пусть $p_1, ..., p_n$ - простые множители $p-1$. Чтобы проверить, является ли g генератором $\mathbb{Z}_p^*$, вычислим все $g^{(p-1)/p_i},\ i = 1..n$ и сравним с 1. Если есть единица, то g - не генератор.

Пример 16.1. $p = 11$, $p - 1 = 10 = 2 * 5$.

$2^2 = 4 \not\equiv 1 \bmod 11$.

$2^5 = 32 \not\equiv 1 \bmod 11$

2 - генератор $\mathbb{Z}_{11}^*$.

$3^2 = 9 \not\equiv 1 \bmod 11$.

$3^5 = 243 \equiv 1 \bmod 11$.

3 - не генератор $\mathbb{Z}_{11}^*$.

16.3 Уязвимость протокола при наличии большого числа малых делителей у порядка группы

Если B не проверяет порядок присланного элемента, возможна атака на протокол DH. В реальной реализации протокола IPSec такой проверки не было!

Пусть d - малый делитель $(p-1)$. Злоумышленник присылает элемент h : $|\langle h \rangle| = d$. Теперь сессионный ключ $k = h^b$ пользователя B может иметь лишь d значений и полностью определяется значениями h и $(b \bmod d)$. Исходящее от B сообщение можно взломать перебором, найдя таким образом k и значение $(b \bmod d)$.

Если у $(p-1)$ много относительно малых делителей, то эту атаку можно повторять и получить набор $(b \bmod d_1), ..., (b \bmod d_t)$. По китайской теореме об остатках, можно найти $b \bmod (d_1 \cdot ... \cdot d_t)$. Если $(d_1 \cdot ... \cdot d_t)$ велико, злоумышленник получит большой объем информации о постоянном секретном ключе b пользователя B.

Защита от этой атаки: порядок любой подгруппы является делителем числа $(p-1)$, и обратно, $\forall d$ - делителя $(p-1)$ $\exists$ подгруппа $\mathbb{Z}_p^*$ порядка d. Если хотим, чтобы не было подгрупп малых размеров, надо выбирать простое p так, чтобы у $(p-1)$ не было малых делителей, кроме числа 2. Используем надежные простые числа $p = 2q + 1$ и подгруппу квадратичных вычетов в Z_p^*. Для надежного простого числа каждый элемент этой подгруппы - ее генератор.

16.4 Использование надежных простых чисел

Определение 16.1. *Надежное простое число* (safe prime) - это простое число вида $p = 2q + 1$, где q - также простое.

В этом случае у $\mathbb{Z}_p^*$ будут только 4 подгруппы:

- {1} - порядка 1,
- {1, (p-1)} - порядка 2,
- подгруппа порядка q.
- подгруппа порядка 2q, равная $\mathbb{Z}_p^*$.

Т.е. нет малых подгрупп.

Недостаток группы Z_p^*: при использовании всех возможных генераторов группы $\mathbb{Z}_p^*$ в протоколе DH раскрывается один бит ключа. Половина элементов $\mathbb{Z}_p^*$ являются квадратичными вычетами, половина - нет. Можно быстро проверить (вычислить символ Лежандра), является ли число квадратом по модулю p, не находя корня из числа.

Очевидно, генератор $g \in Z_p^*$ - не квадратичный вычет. Тогда при четном x $g^x \in QR(p)$, при нечетном $g^x \in QNR(p)$. Это может легко проверить злоумышленник и найти младший бит числа x.

Поэтому надо использовать только квадратичные вычеты по модулю p. $QR(p)$ - подгруппа Z_p^* порядка q. q - простое, поэтому вложенных подгрупп нет. Значит, всякий элемент $QR(p)$ имеет порядок q.

Протокол с использованием надежных простых чисел:

- Выберите пару простых $p, q:\ p = 2q+1$. (Например, перебором простого числа q.)
- Выберите $a \overset{R}{\leftarrow} \{2, ..., (p-2)\}$.
- $g = a^2 \bmod p$. Надо: $|\langle g \rangle| = q$. Т.о., g подходит $\Leftrightarrow g \neq 1,\ g \neq p-1$ и $g^q = 1 \bmod p$. Иначе выберите другое a и проверьте снова.

  Найденный элемент $g \in QR(p)$, причем он - генератор этой подгруппы.

Каждый раз, когда B получает элемент h, который (предположительно) является степенью g, он должен проверить, так ли это. Если p - надежное простое число, это проверить просто: $\left(\frac{h}{p}\right) \overset{?}{=} 1$.

16.5 Использование подгрупп меньшего размера.

При использовании надежных простых чисел протокол DH работает медленно: если длина p - n бит, то длина q - $n-1$ бит. При возведении в степень все показатели степеней приводятся по $\bmod q$ и будут иметь длину до $n-1$. При больших p это влияет на скорость работы.

Решение: использовать подгруппы меньшего размера.

Выберем q - 256-битовое простое число. Затем ищем N из некоторого наперед заданного диапазона такое, что $p = Nq + 1$ - простое. Длина простого p - несколько тысяч бит. N - четное.

Найдем элемент $g \in Z_p^*$ порядка q так же, как и до этого: $a \overset{R}{\leftarrow} Z_p^*$, $g = a^N$. Если $g \neq 1$, $g \neq p-1$, $g^q = 1$, то он подходит, иначе выберем другой элемент a.

Полученный набор (p, q, g) пригоден для использования в протоколе DH.

Задача 16.1. Почему g - квадратичный вычет?

$\forall s \in \langle g \rangle$ справедливо $s^q = 1$. Поэтому $s^e = s^{e \bmod q}$. Это сокращает объем вычислений, по сравнению со случаем, когда $p = 2q+1$. Чем больше значение N, тем больше разница. Выбранное заранее значение q гарантирует, что размер подгруппы достаточно велик.

Замечание. Так же как раньше, необходимо проверять, что присланный элемент $s \in \langle g \rangle$.

17 Инфраструктура открытого ключа

17.1 Обшее описание

Инфраструктура открытого ключа (public key infrastructure, PKI) - реально использующаяся система, с помощью которой можно определить, кому принадлежит тот или иной ключ. Административная, а не математическая/сложностная.

Пользователь A генерирует пару (pk_A, sk_A) и передает pk_A центру сертификации (ЦС). ЦС проверяет, что A тот, за кого он себя выдает, затем подписывает ключ pk_A своей ЭЦП (создает сертификат) и возвращает его A.

Определение 17.1. *Сертификат* (certificate) пользователя A - это:

- сообщение "pk_A - публичный ключ A"
- дополнительная информация, например, права пользователя A
- все эти данные подписаны ЭЦП ЦС.

Чтобы получать сообщения от B, A отсылает ему свой публичный ключ и сертификат. B знает открытый ключ ЦС, проверяет сертификат и убеждается, что это действительно ключ A. В идеале, существует единый ЦС, всемирная инфраструктура открытого ключа. Это невозможно.

Пример 17.1.

1. VPN. IT-отдел компании - ЦС, предоставляет сотрудникам компании сертификаты.
2. Электронные платежные поручения. Банк заверяет открытые ключи своих клиентов.

3. Ассоциация кредитных карт: клиент банка A совершает покупку по кредитке в точке, обслуживающейся в банке B. Банк A должен рассчитаться с банком B. Для этого они должны идентифицировать друг друга. ЦС заверяет ключи каждого банка.

Многоуровневые сертификаты.

В приведенном примере банков (пользователей) слишком много, поэтому главный ЦС создает региональные отделения, заверяет их ключи, добавляя в сертификат описание: “предъявитель данного ключа - региональное отделение X. Оно может удостоверять этим ключом ключи других пользователей”.

После этого сертификат открытого ключа пользователя состоит из двух частей: сертификата ЦС, авторизующего ключ регионального отделения X, и сертификата отделения о том, что это - ключ данного пользователя.

Получается цепочка сертификатов $\Rightarrow$ больше вычислений и потенциальных мест для атаки. Вариант свертывания: получив двухуровневый сертификат, клиент отсылает его ЦС. Тот возвращает одноуровневый сертификат, подписанный ЦС. На практике свертывание сертификатов не используется.

Некоторые параметры сертификата:

- дата начала действия
- дата конца действия
- кем выдан
- кому выдан
- разрешенные владельцу сертификата действия
- техническая информация: значение открытого ключа, тип алгоритма открытого ключа, и т.д.
- прочие данные

Самый распространенный формат сертификатов - X.509 v3.

17.2 Проблемы инфраструктуры открытого ключа

1. Клиенты должны доверять корневому ЦС. Сертификаты ЦС подписаны их предъявителями (self-signed certificate): издатель совпадает с субъектом. Пример корневого ЦС: GeoTrust Global CA. В любом веб браузере (или в ОС) предустановлен список корневых сертификатов, которым можно доверять. Поэтому

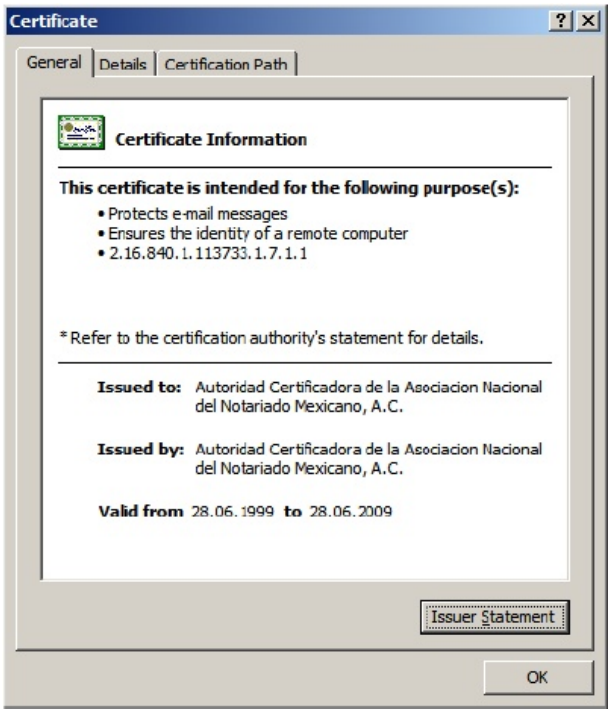

Рис. 29: Информация о сертификате.

они не могут быть отозваны. В результате проверка любого сертификата проходит оффлайн, обеспечивая защиту от атаки посредника. И нужно "надежно" распространить лишь сертификаты корневых ЦС. (Например, вместе с операционной системой.)

2. В сертификат можно упаковать разрешения (запрещения) на очень большое число разнородных действий: открытие файлов, доступ к базе данных и т.д. Поэтому реализация системы доступа, основанная на сертификатах, получается сложной и далекой от идеальной. (Сообщения о передаче полномочий и т.д.)
3. Отзыв сертификатов. Пусть клиент скомпрометировал или потерял свой секретный ключ, или сотрудник компании уволился. Варианты решения проблемы отзыва сертификата:
   - Список отзыва сертификатов ⇒ проверка сертификата - онлайн. Каждый член PKI должен иметь доступ к центральной БД с этим списком, получать его целиком. Это плохо, т.к. точка атаки - не только взлом, но и DoS. Если используется распределенная БД, то репликация требует ресурсов. Кроме того, как часто реплицировать?
   - Online Certificate Status Protocol. Протокол запроса статуса сертификата,

и получения ответа. Сообщения передаются открытым текстом. Меньше нагрузка, чем с БД отозванных сертификатов.

- Быстрое устаревание. Сертификат клиента действует от 10 мин. до 24 ч. Когда он почти устарел, клиент обращается в ЦС за новым сертификатом. Не требуется список отозванных сертификатов. Но для обновления сертификата клиент должен быть "постоянно" подключен к ЦС.
- Не PKI, а централизованный сервер ключей. Доверенный центр T, с которым может общаться каждый клиент. Если A хочет послать сообщение B, он получает ключ K_{AB} и его зашифрованную копию $E_{K_B}(K_{AB})$, оба подписаны центром T. A посылает B подписанный зашифрованный сессионный ключ $E_{K_B}(K_{AB})$ и они могут общаться. Например, так устроена система управления ключами Kerberos.

Замечание. для сравнительно небольших систем сложность PKI слишком велика, используются серверы ключей. Для большой, распределенной, слабо контролируемой системы используется PKI.

17.3 Жизненный цикл ключа

1. Создание. Пользователь A создает пару pk_A, sk_A.
2. Сертификация. ЦС сертифицирует публичный ключ A, наделяет того полномочиями.
3. Распространение pk_A.
4. Активное использование pk_A в транзакциях.
5. Пассивное использование. pk_A не применяется в новых транзакциях, но принимаются старые с его использованием. (Пример: email доходит не мгновенно.)
6. Окончание срока действия.

Часть V

Безопасные сетевые протоколы

18 Протокол TLS/SSL

Основная цель раздела: описать алгоритм создания новой сессии в протоколе TLS/SSL и дать пример практического использования исследованных ранее криптографических примитивов.

18.1 История TLS/SSL и общие сведения

Протокол Transport Layer Security / Secure Socket Layer находится на 6 уровне модели OSI. (См. табл. 10.) (Инициализация сессии TLS - на 5 уровне.) Он обеспечивает абстракцию уровня представления и шифрования данных для выше расположенных протоколов, в частности, для протоколов HTTP, FTP, SMTP.

Цели протокола:

- аутентификация сервера с помощью сертификата открытого ключа.
- аутентификация клиента с помощью сертификата открытого ключа (почти не используется).
- обеспечение конфиденциальности, целостности информации, передаваемой между приложениями.
- защита от атаки воспроизведения.

Развите TLS/SSL.

- SSL 1.0 взломан во время презентации в MIT. Уязвим к атаке воспроизведения, не обеспечивал целостность данных, использовал поточный шифр (1994).
- SSL 2.0 впервые реализован в Netscape Navigator 1.0 (1995).
- SSL 3.0 впервые проходил публичное обсуждение протокола. Предложен как стандарт IETF (1996). Работа не завершена.
- TLS 1.0 - стандарт Internet Engineering Task Force (1999).
- TLS 1.1 - стандарт IETF (2006). Обеспечил защиту от padding oracle attack, implicit IV attack (BEAST).
- TLS 1.2 - стандарт IETF (2008). Добавлены новые шифры и прочие технические улучшения.

Уровень модели OSI	Функции	Протоколы
7. Прикладной (application)	Доступ к сетевым службам	HTTP, POP3, FTP, SMTP, TELNET, RCP, PGP, **SSH**
6. Уровень представления (presentation)	Представление и шифрование данных	LPP, RDP, X25 PAD, **SSL/TLS**
5. Сеансовый (session)	Управление сеансом связи	L2TP, PPTP, NetBIOS, RPC
4. Транспортный (transport)	Связь между конечными пунктами и надежность	TCP, UDP
3. Сетевой (network)	Определение маршрута и логическая адресация	IP/IPv4/IPv6, **IPSec**, ICMP, RIP
2. Канальный (data link)	Физическая адресация	Ethernet, 802.11 wLAN, TokenRing
1. Физический (physical)	Работа с сигналами, представление двоичных данных	BlueTooth, IRDA, 802.11 WiFi

Таблица 10: Модель OSI и примеры протоколов.

18.2 Инициализация сессии TLS

Создание защищенного соединения TLS происходит путем обмена несколькими сообщениями (см. рис. 30).

1. Инициализация сессии, выбор метода обмена ключами, шифрования и хэш функции.
   - сообщение Client Hello содержит ClientHello.nonce, ID сессии (при восстановлении сессии), список поддерживаемых методов обмена ключами, шифров и хэш функций.
   - сообщение Server Hello содержит ServderHello.nonce, выбор шифра и хэш функции из предложенных.
2. Аутентификация сервера.
   - сообщение Server Certificate содержит открытый ключ сервера и тип шифра, заверенные сертификатом. Клиент может проверить сертификат у ЦС.

- сервер может потребовать аутентификацию клиента (обычно не требует), тогда тот должен прислать свой открытый ключ и сертификат.
- сообщение Server Hello Done

3. Передача Premaster Secret. Клиент создает случайное значение premaster_secret, шифрует его публичным ключом сервера и посылает серверу. Это сообщение также содержит уведомление о применении в дальнейшем симметричного шифрования с ключом на основе premaster_secret.
4. Создание master_secret. И клиент, и сервер вычисляют:
   master_secret = ПСФ(premaster_secret || “master secret” || ClientHello.nonce || ServerHello.nonce).
   ПСФ - это криптографическая хэш функция. По стандарту - это SHA-256.
5. Создание сессионных ключей. И клиент, и сервер вычисляют:
   key_schedule = ПСФ(master_secret || “key expansion” || ClientHello.nonce || ServerHello.nonce).
6. Клиент посылает зашифрованное симметричным шифром с первым сессионным ключом сообщение Client Finished.
7. Сервер расшифровывает сообщение клиента. Если сообщение расшифровано неверно, сессия не создается. Если верно, сервер подтверждает прием, отвечая зашифрованным симметричным шифром сообщением Server Finished.

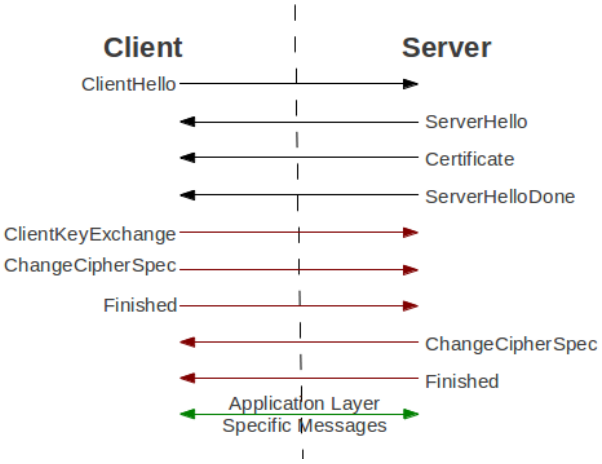

Рис. 30: Обмен сообщениями при инициализации сессии TLS.

После того как сессия установлена, данные передаются зашифрованными симметричным шифром. Структура пакетов сессии описана в секции “Протокол TLS/SSL после согласования ключей” раздела “Заверенное шифрование”.

18.3 Достоинства и недостатки TLS/SSL

Достоинства TLS/SSL:

- Обеспечивает конфиденциальность данных, контроль целостности и защиту от атаки воспроизведения.
- HTTPS сессии “знают” о том, что они используют протокол TLS. Клиенты уверены в том, что соединение безопасное.
- TLS реализован на уровне приложений, т.е. не в ядре операционной системы. Встроен в браузеры, не требует настройки на стороне клиента.
- Можно назначить разные права доступа разным приложениям.
- Возможна аутентификация пользователя, а не оборудования.
- Протокол TLS-VPN обеспечивает удаленный доступ в защищенную сеть из браузера внешней машины по протоколу HTTPS. Не нужен специальный софт, не требует настройки.
- Сессия закрывается явным образом при отключении клиента: сообщения TLS close, TCP fin.

Недостатки TLS/SSL:

- Для аутентификации сторон используется инфраструктура открытого ключа, которая требует много вычислительных ресурсов, т.к. использует асимметричные криптосистемы.
- Проблемы масштабирования и потребления памяти при наличии большого числа TCP соединений, т.к. каждый клиент устанавливает свое соединение с сервером.
- Медленное и ресурсоемкое восстановление сессии.

19 Протокол IPSec

Основная цель раздела: описать алгоритм создания новой сессии в протоколе IPSec и дать пример практического использования исследованных ранее криптографических примитивов.

19.1 История IPSec и общие сведения

Сначала вся сеть предприятия находилась в одном здании. Ограничение доступа к сети - физические замки. Если нужно соединить два соседних здания, протягивали выделенный кабель между ними. Позже появилась необходимость соединить сети на разных концах континента, сохранив конфиденциальность и целостность информации, и обеспечить контроль доступа к сети.

1993, протокол Software IP Encryption protocol *swIPe* - Columbia University и AT&T Bell Labs

1994, Gauntlet firewall - первый реальный VPN между двумя подсетями в США.

1993, протокол IP ESP - проект DARPA

1995, начало разработки открытого стандарта в IETF

1998, RFC 2401 - первый стандарт IPSec

2014: около 30 документов (стандартов) IETF.

Цели создания протокола IPSec:

- Конфиденциальность и/или целостность передачи информации между серверами
- Защита от воспроизведения
- Аутентификация источника данных
- Контроль доступа

Предоставить этот сервис для всех протоколов модели OSI, лежащих выше IP/IPSec

IPSec реализует два основных протокола:

- только контроль целостности, Authenticated Header (AH).
- шифрование и контроль целостности, Encapsulation Security Payload (ESP).

Каждый из протоколов имеет два режима:

- транспортный (точка-точка, защита содержимого IP датаграммы)
- туннельный (gateway-gateway, защита всей IP датаграммы)

Можно комбинировать протоколы AH и ESP, помещая один вовнутрь другого.

19.1.1 Протокол Authenticated Header

Пакет данных, передаваемых по протоколу Authenticated Header, содержит дополнительный заголовок для этого протокола. Поля заголовка Authenticated Header:

- Next Header - номер протокола следующего заголовка. Это протокол того же или следующего уровня OSI. Например, IP или TCP.
- Payload Length - длина этого заголовка AH минус 2.
- Resered - не используется, 0.
- SPI - security parameter index, уникальный локальный индекс ассоциации безопасности (АБ), описывающей все параметры соединения.
- Seq No - счетчик сообщений, отправленных с этой АБ.
- Authentication Data - метка кода аутентичности от всех неизменных частей IP заголовка, AH заголовка, и данных.

Используются раздельные АБ для входящих и исходящих сообщений. Их можно настроить вручную или автоматически (например, с помощью Internet Key Exchange protocol).

Создание пакета даных протокола AH в туннельном и транспортном режимах изображено на рис. 31.

19.1.2 Протокол Encapsulation Security Payload

В протоколе ESP Envelope пакет данных включает в себя:

1. ESP Header:
   - SPI - уникальный индекс АБ (см. ниже).
   - Seq No - счетчик сообщений, отправленных с этой АБ.
2. Payload - данные.
3. ESP Trailer:
   - Padding - для шифра, 0..255 байт.
   - Pad Length - число байт в Padding.
   - Next Header - номер протокола заголовка в Payload. Это протокол того же или следующего уровня OSI. Например, IP или TCP.
4. ESP Authentication Data - метка кода аутентичности, вычисленного по всему ESP Envelope.

Протокол ESP может использоваться в транспортном или туннельном режиме.

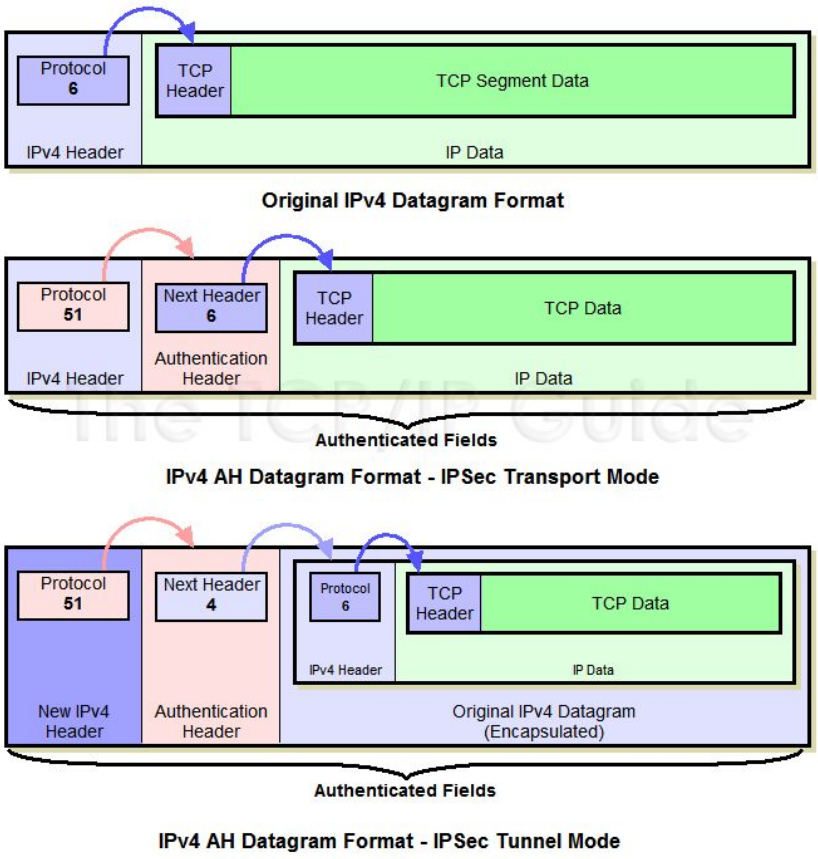

Рис. 31: Authenticated Header в туннельном и транспортном режимах.

Настройки протокола ESP определяют, какое использовать шифрование (или без него) и код аутентичности (или без него). Режим шифрования - всегда Encrypt-then-MAC.

VPN использует IPSec в туннельном режиме, протокол ESP с шифрованием и кодом аутентичности. Т.е. IP пакет целиком инкапсулируется, шифруется и защищается от изменений.

19.2 Инициализация сессии IPSec

Рассмотрим протокол IKE - internet key exchange.

Может использоваться один из следующих методов авторизации сторон:

1. Общий ключ (pre-shared key). Секретный ключ симметричного может быть задан вручную при настройке соединения.
2. Сертификат (открытый ключ, заверенный ЭЦП ЦС).

Определение 19.1. Протокол обеспечивает *совершенную прямую секретность* (perfect forward secrecy), если выполняются два условия:

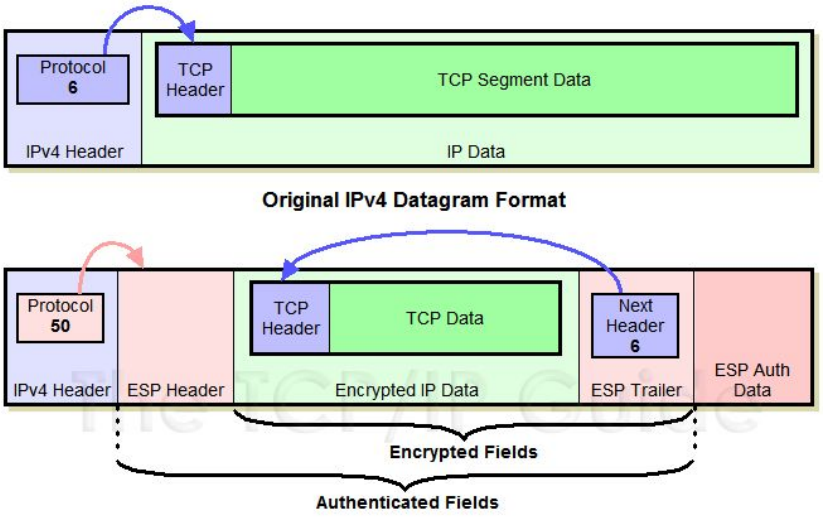

Рис. 32: Пакет ESP в транспортном режиме.

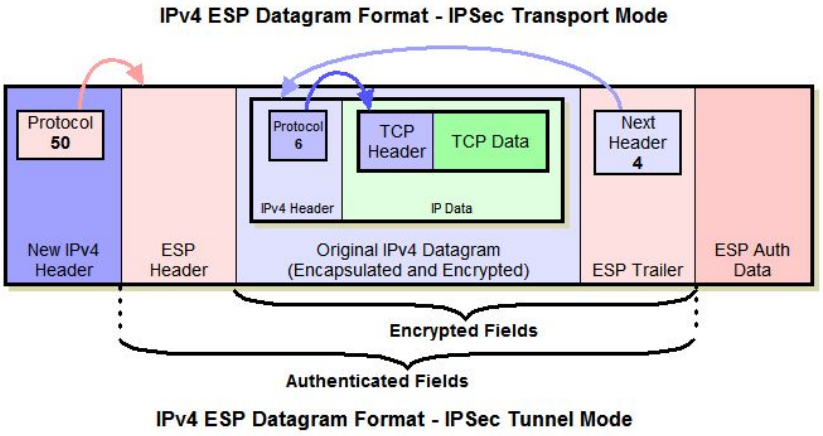

Рис. 33: Пакет ESP в туннельном режиме.

1. если один сессионный ключ скомпрометирован, ни первичный ключ, ни другие сессионные ключи не могут быть найдены.
2. если первичный ключ скомпрометирован, сессионные ключи не могут быть найдены.

Это одна из опций протоколов IPSec, TLS 1.2, SSH v2.

Инициализация сессии протокола IPSec проходит в две фазы:

1. Цель - взаимная авторизация и создание первичного ключа для ассоциации безопасности (АБ) фазы 2. Используется заданный метод авторизации и протокол Диффи-Хеллмана для создания первичного ключа. Целостность сообщений обеспечивается с помощью метки HMAC или ЭЦП.
2. Цель - создание сессионных ключей. Сессионный ключ - это временный ключ

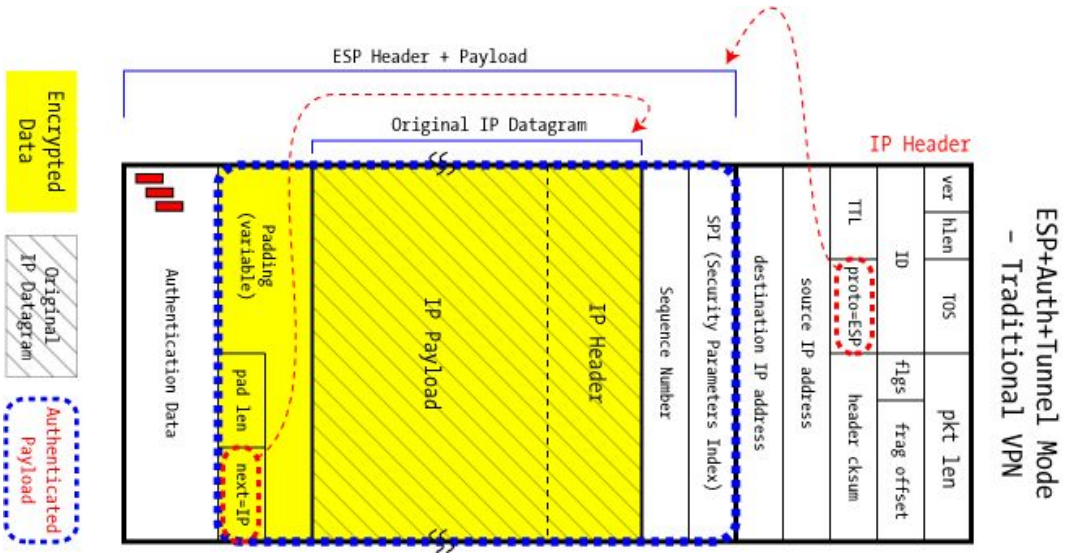

Рис. 34: Инкапсуляция данных в протоколе VPN.

для одной АБ. Они могут вычисляться по первичному ключу, который создан в фазе 1. Если требуется совершенная прямая секретность, на фазе 2 создаем новый секрет по протоколу Диффи-Хеллмана. АБ для фазы 2 установлена в результате фазы 1, защищает обмен данными. Когда сессионный ключ, созданный на фазе 2, устаревает, повторяется только фаза 2, не фаза 1.

19.3 Достоинства и недостатки IPSec

Достоинства IPSec:

- Обеспечивает конфиденциальность данных, контроль целостности и защиту от атаки воспроизведения.
- Работает на третьем уровне OSI. Предоставляет защищенный сетевой уровень для любого транспорта или приложения. Они могут и не знать об этом.
- Поддерживает два режима: транспортный (точка - точка) и туннельный (gateway - gateway)
- Большинство VPN используют IPSec.

Недостатки IPSec:

- Нужен специальный режим для работы за NAT, который приводит к усложнению протокола и конфигурации.
- Работает на третьем уровне OSI, поэтому изменения в протоколе - это изменения в ядре ОС.
- Обеспечивает аутентификацию устройства (хоста или gateway), а не пользова-

теля.

- Неявное начало, ведение и завершение сессии. Приложения не знают, предоставляется ли этот сервис.
- Нет контроля потери связи. (Отдан протоколам выше в стеке OSI.)
- Нет функции ограничения доступа для отельных приложений. Весь хост становится частью сети VPN.
- Запутанная конфигурация протоколов IPSec, IKE и др.

20 Утилита ssh

Основная цель раздела: описать утилиту ssh - наиболее распространенное средство для доступа на удаленные машины под ОС Unix.

20.1 История ssh и общие сведения

ssh - это утилита операционных систем семейства Unix, позднее портированная в другие ОС. Цель создания утилиты: обеспечить защищенный доступ на удаленную машину по сети для выполнения команд или передачи данных. Работает на седьмом уровне модели OSI, т.е. это - приложение.

Ранее для удаленного доступа использовалась утилита rsh, remote shell (1983):

- Все данные, всегда, включая логин и пароль, передавались в открытом виде.
- Данные не были зашифрованы и не было контроля целостности, поэтому злоумышленник мог встроиться в протокол как “клиент” или “сервер” и получить доступ к передаваемой информации.
- Хосты сторон не были аутентифицированы.

Первая версия утилиты ssh v1 (1995) обеспечивала шифрование данных. В качестве контроля целостности использовалась контрольная сумма CRC, которая вычислялась по зашифрованным данным. Поэтому злоумышленник мог добавить в канал свои сообщения. Прочие ошибки в реализации привели к тому, что была возможна авторизация злоумышленника.

Вторая версия утилиты - ssh v2 (1997), стандарт IETF с 2006.

Основные компоненты и возможности ssh:

- sshd - сервер ssh, обрабатывает запросы клиентов.

- ssh реализует удаленный терминал, или исполнение команды на удаленном хосте, или туннель для передачи данных (port forwarding).
- scp, sftp - утилиты для копирования файлов.
- ssh-keygen - утилита для создания ключей пользователя.

ssh обеспечивает:

- безопасный канал. Режим шифрования EaM обеспечивает защиту от подслушивания и контроль целостности данных;
- как следствие, защиту от атаки посредника;
- авторизацию сервера по публичному ключу. У клиента есть таблица соответствия IP адрес ↔ публичный ключ хоста. Так обеспечивается защита от подмены IP сервера;
- авторизация клиента возможна по паролю, публичному ключу или сертификату. Она происходит по защищенному каналу;
- использование номера сообщения обеспечивает защиту от атаки воспроизведения.

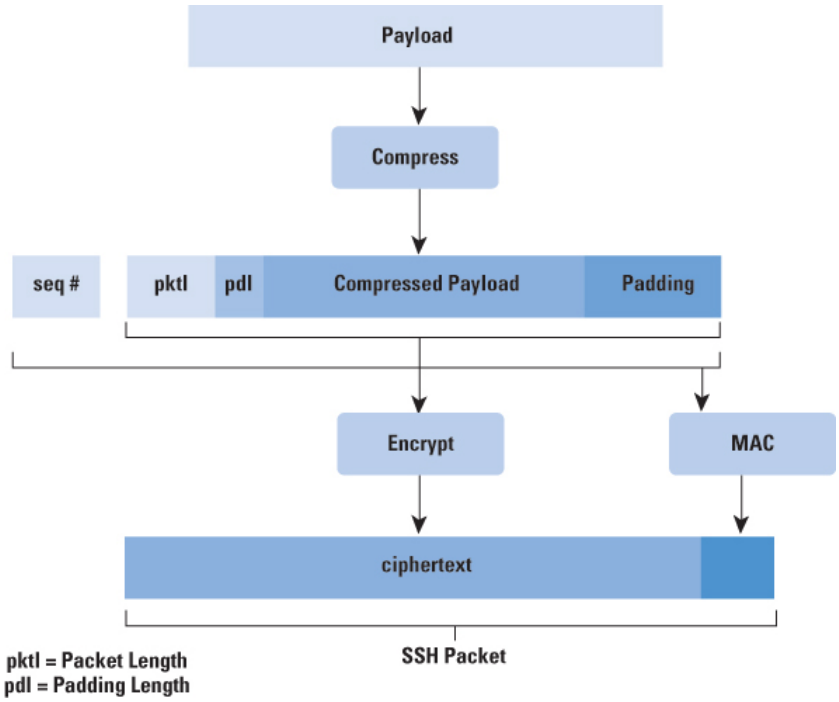

Рис. 35: Защита данных в протоколе ssh.

20.2 Инициализация сессии ssh

Обозначения:

C - клиент,

S - сервер,

P_X - открытый ключ X,

S_X - секретный ключ X,

Фаза 1. Начало сессии - аутентификация сервера.

1. C→S: Vc - версия SSH клиента.
2. S→C: Vs - версия SSH сервера.
3. C→S: Ic - список алгоритмов обмена ключами, список алгоритмов шифрования сессии.
4. S→C: Is - алгоритм публичного ключа хоста сервера, выбранный алгоритм обмена ключами, выбранный алгоритм шифрования сессии.
5. S→C: P_S, [CERT(P_S)]. Клиент проверяет P_S - публичный ключ хоста сервера. Если проверка пройдена, он авторизует сервер.

Локальные таблицы с публичными ключами удаленных хостов и публичные ключи клиентов, которым разрешен доступ на этот хост, хранятся в файлах в директории $\sim /.ssh$.

Файл authorized_keys2 содержит публичные ключи клиентов. Файл known_hosts - публичные ключи удаленных хостов. Файл id_rsa - секретный ключ пользователя для ЭЦП RSA. Файл id_rsa.pub - публичный ключ пользователя для ЭЦП RSA. Также могут быть файлы с ключами для других алгоритмов ЭЦП.

Безопасность хранения ключей в ОС *nix достигается тем, что право на запись, чтение и исполнение директории $\sim /.ssh$ и всех файлов в ней имеет только сам пользователь.

При первом соединении с удаленным хостом клиенту сообщают fingerprint - 128-битный хэш публичного ключа удаленного хоста. Его правильность можно проверить вручную или автоматически (как правило, это не делается).

Если ключ сервера не совпал с ключом, сохраненном в файле known_hosts, выдается сообщение об ошибке (изменился публичный ключ сервера) и соединение разрывается.

Фаза 2. Создание первичного ключа (по стандарту RFC 4432).

1. Клиент создает случайный первичный ключ (master_key), шифрует его и от-

правляет серверу:

C→S: RSA(P_S, master_key). Это обеспечивает совершенную прямую секретность.

2. Сервер расшифровывает сообщение и узнает master_key. Для контроля целостности обмена данными сервер вычисляет

   H = hash(Vc || Vs || Ic || Is || P_S || P_T || RSA(P_T, master_key), master_key)

   и передает его клиенту.

Алгоритмы создания мастер ключа:

- RFC 4253 (2006): протокол Диффи-Хеллмана. Нагружает ЦП обеих сторон.
- RFC 4432 (2006): RSA. Меньше грузит ЦП клиента (RFC 4432), нагружает ЦП сервера.
- RFC 5656 (2009): ECRSA. Еще меньше грузит ЦП обеих сторон.

Фаза 3. Создание сессионных ключей

По стандарту RFC 4253, сессионные ключи и векторы инициализации для первого сообщения вычисляются как значения хэш функции от первичного ключа, идентификатора сессии, значения H и служебных констант.

Сессионные ключи для симметричного шифрования и вычисления кода аутентичности различны для передачи данных в одну и другую стороны, всего их 4. Они меняются по времени или объему данных. По истечении срока действия сессионных ключей, они создаются заново, начиная с фазы 2.

Фаза 4. Авторизация клиента. Передаваемые данные шифруются и заверяются симметричным шифром и кодом аутентичности с ключами, созданными на фазе 3.

Методы авторизации клиента:

- публичный ключ P_C
- пароль
- сертификат (RFC 6187, 2011 год)

Авторизация по публичному ключу:

1. C→S: username.
2. Сервер создает случайное число r. S→C: E(P_C, r).
3. C→S: r.
4. S→C: Ack или Reject, [список других способов авторизации].

Если авторизация по открытому ключу не успешна, как правило, пользователь

может авторизоваться по паролю. После этого сессия связи установлена.

Часть VI

Специальные криптопротоколы

21 Доказательство без разглашения информации

Основная цель раздела: описать протокол, позволяющий доказать что-либо другому лицу, не сообщая доказательство и не добавляя никакой новой информации к тому, что другой знал априори. Подделать доказательство должно быть невозможно.

21.1 Интерактивное доказательство

Рассмотрим интерактивные доказательства.

Пример 21.1. Другой (V, verifier) не может различить два предмета, а я могу. Я (P, prover) хочу доказать ему этот факт.

Протокол: дать V эти два предмета. P выбирает один из них. V скрытно от P либо меняет предметы местами, либо нет, и предъявляет их P. P выбирает тот же предмет. Повторить это t раз. Если P не умеет различать предметы, вероятность, что он всегда укажет на один и тот же предмет, равна 2^{-t}.

Пусть L - некоторый язык, т.е. подмножество слов из $\{0,1\}^*$. Пусть $L \in \mathrm{NP}$, т.е. задача "принадлежит ли x L" из класса NP.

Определение 21.1. Пара алгоритмов (P, V) и правила их взаимодействия $(\leftrightarrow)$ являются *интерактивным доказательством* (interactive proof), если выполнены следующие свойства:

1. Полнота. $\forall x \in L$ существует способ доказать этот факт, т.е. $\exists y \in \{0,1\}^*$ - описание доказательства: $(P(x,y) \leftrightarrow V(x)) \mapsto 1$.

2. Непротиворечивость. $\forall x \notin L, \forall P \in \mathrm{PT}, \forall y \in \{0,1\}^*$ вероятность того, что $(P(x,y) \leftrightarrow V(x)) \mapsto 0$, равна $1 - \varepsilon(len(x))$.

3. $V \in$ PPT, $P \in$ PPT по $len(x)$.

21.2 Числа Блюма

Определение 21.2. *Простые числа Блюма* (Blume primes) - это простые числа вида $p = 4k + 3$, где $k \in$ N.

Утверждение 21.1. Пусть p - простое число Блюма. Тогда $-1 \in QNR(p)$.

Доказательство. $\forall a \in Z_p^*$, если $x^2 = a$, то $(-x)^2 = a$. Уравнение $x^2 = 1$ имеет не более 2 корней; 1, -1 - корни. Пусть уравнение $x^2 = -1$ имеет 2 корня в Z_p^*. Тогда множество корней 4 степени из 1 имеет мощность 4: $\{1, -1, i, -i\}$, где $i = \sqrt{-1}$, и является группой по умножению. При этом $|Z_p^*| = p-1$ не делится на 4. Противоречие с тем, что порядок любой подгруппы - делитель поряда группы. □

Определение 21.3. $n \in$ N - число Блюма, если $n = pq$, где p, q - простые числа Блюма.

Утверждение 21.2. $x^2 = y \bmod pq \Leftrightarrow x^2 = y \bmod p,\ x^2 = y \bmod q$,

Доказательство. $x^2 = y + pqk \Rightarrow x^2 = y \bmod p,\ x^2 = y \bmod q$. В обратную сторону - используем китайскую теорему об остатках. □

Определение 21.4. Символ Якоби числа a по модулю $n = pq$ - это $\left(\frac{a}{n}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{q}\right)$.

Исследуем свойства значений символа Якоби по модулю n, где $n = pq$ - число Блюма.

Т.к. по КТО $Z_p \times Z_q \simeq Z_{pq}$ то, если $y \in QR(n)$, в $Z_{pq}\exists$ ровно 4 корня из y: $(x_p, x_q), (-x_p, x_q), (x_p, -x_q), (-x_p, -x_q)$, из которых ровно один QR - пара $a = (x_p, x_q)$, где $x_p \in QR(p)$, $X_q \in QR(q)$. Очевидно, $\left(\frac{a}{n}\right) = 1$. Также среди этих 4 корней $\exists$ пара $b = (-x_p, -x_q)$, где $-x_p \in QNR(p)$, $-x_q \in QNR(q)$. Очевидно, $\left(\frac{b}{n}\right) = 1$. Для остальных двух корней символ Якоби равен -1. Т.е., ровно половина элементов Z_{pq}^* имеет $\left(\frac{a}{n}\right) = 1$, и ровно половина из них $\in QR(n)$. Заметим также, что $\left(\frac{-1}{n}\right) = 1$.

21.3 Протокол Файге, Фиата, Шамира

Все операции проводятся над элементами группы Z_n^*, где n - число Блюма.

$s \neq \pm 1$ - секретное значение. Его знает только P. $v = s^2$ - публичное значение. Размещено в справочнике как ключ P.

P доказывает V, что он знает $\sqrt{v}$.

- P выбирает $r \xleftarrow{R} Z_n^*$, $z \xleftarrow{R} \{-1, 1\}$ и вычисляет $x = z \cdot r^2 \bmod n$.
- P отправляет x к V.
- V выбирает $b \in \{0, 1\}$. V отправляет b к P.
- P вычисляет $y = rs^b \bmod n$. P отправляет y к V.
- V проверяет, что $y^2 = \pm xv^b \bmod n$.

  T.e. V просит P найти корень из xv или x.

  Для простоты, пусть $z = 1$.

  Необходимо доказать:

1. Полнота. (Очевидно.)
2. Непротиворечивость. Злоумышленник E не знает s.
   - Пусть он предполагает, что V пришлет $b = 0$. E выбирает r, вычисляет $x = r^2$, $y = r$.
   - Или он предполагает, что $b = 1$. Выбирает r, вычисляет $x = r^2 v^{-1}$, $y = r$.
   - Если E не угадал значение b, он не может верно ответить на запрос V.

   Шанс угадать равен 1/2. После t раундов - $1/2^t$.
3. Не разглашение информации. Что узнает злоумышленник, слушающий канал?
   - Пусть $b = 0$. $(x, y) = (r^2, r)$, где r - случайный элемент Z_n^*, $x = r^2$ - случайный элемент из $QR(n)$.
   - Пусть $b = 1$. $(x, y) = (r^2,\ rs)$. r - произвольный случайный элемент, значит rs - тоже произвольный случайный элемент.

   В обоих случаях $x = r^2$ - случайный элемент из $QR(n)$. $\Rightarrow$ случайный знак $z = \pm 1$ нужен. Без него V узнает, что присланное ему число $x \in QR(n)$. Это - новая информация. С использованием z $x \xleftarrow{R} \{a \in Z_n^* | \left(\frac{a}{n}\right) = 1\}$. Символ Лежандра V может эффективно вычислять самостоятельно, поэтому дополнительная информация не разглашается.

Допустимо ли повторное использование значения r? Предположим, что переданы $y_0 = r$ и $y_1 = rs$ при $b = 0$ и $b = 1$. $\Rightarrow y_0^2 \equiv x \bmod n$ и $x \equiv y_1^2 v^{-1} \bmod n$.
Решая эти уравнения относительно v, находим $v \equiv y_0^{-2} y_1^2 \bmod n$. $\Rightarrow$ находим секретное значение $\pm\sqrt{v} \equiv y_0^{-1} y_1 \bmod n$ без факторизации n.

Параллельная версия протокола.

Пусть s_i - секретные значения P. $v_i = s_i^2$ - публичные значения. Размещены в справочнике как ключ P.

- P выбирает $r \xleftarrow{R} Z_n^*$, $z \xleftarrow{R} \{-1, 1\}$ и вычисляет $x = z \cdot r^2 \bmod n$. Отправляет x к

V.

- V выбирает $b_1, ..., b_k$, $b_i \in \{0, 1\}$. V отправляет их P.
- P вычисляет $y = rs_1^{b_1}...s_k^{b_k} \bmod n$. P отправляет y к V. V проверяет, что $y^2 = \pm x v_1^{b_1}...v_k^{b_k} \bmod n$.

Вероятность имперсонализации после t раундов авторизации равна $1/2^{kt}$.

Пример 21.2. $n = 7 \cdot 11 = 77$. (В реальности длина n - 1024 бита.) $\varphi(7 \cdot 11) = 60$. $|QR(Z_{77}^*)| = 15 = 60/4$. $QR(77) = \{1, 4, 9, 15, 16, 23, ...\}$.

Публичный ключ $\{v_i\} = \{4, 9, 15, 23\}$, $\{v_i^{-1}\} = \{58, 60, 36, 67\}$.

Секретный ключ $\{v_i^{-1/2}\} = \{17, 26, 6, 12\}$.

1. A выбирает случайное значение $r = 9$, $z = +1$, $9^2 \equiv 4 \bmod 77$ и отправляет 4 к B.
2. B отправляет A: 1101
3. A вычисляет $z \cdot 9 \cdot (17^1 \cdot 26^1 \cdot 6^0 \cdot 12^1) \equiv 73 \bmod 77$ и отправляет 73 к B.
4. B проверяет $73^2 \cdot 4^1 \cdot 9^1 \cdot 15^0 \cdot 23^1 \equiv \pm 4 \bmod 77$ - верно.

Идентификация с помощью этого протокола.

Пусть I - информация о человеке A (уникальный ID). Арбитр T знает простые p, q и $n = pq$. Пусть H - криптографическая хэш-функция. Пусть $x = I||j$ - конкатенация I и некоторого небольшого числа j.

T находит перебором набор чисел {j} таких, что $H(I||j) \in QR(n)$. Он может каждую проверку делать быстро.

Следовательно, T имеет набор $v_j = H(I||j)$. Открытый ключ для A - информация I и набор значений {j}. Открытый ключ удостоверен подписью T. Секретный ключ A - корни из v_j.

После проведения авторизации (A перед B) по выше указанному протоколу, B удостоверяется, что арбитр T удостоверил факт того, что I и {j} принадлежат A, а именно, что T сообщил A корни из v_j.

22 Подбрасывание монеты по телефону

Основная цель раздела - описать протокол последовательной коммуникации двух сторон (без арбитра), который имитирует подбрасывание честной монеты и не дает ни одной стороне возможность обмануть другую сторону.

Опишем протокол подбрасывания монеты “по телефону”.

1. A выбирает случайные p, q - простые числа Блюма, вычисляет $n = pq$, отправляет n B.
2. B выбирает $x \overset{R}{\leftarrow} Z_n$, вычисляет $y \equiv x^2 \bmod n$, отправляет y A. $y \in QR(n)$.
3. A вычисляет $z_p \equiv y^{(p+1)/4} \equiv y^{1/2} \bmod p$, $z_q \equiv y^{(q+1)/4} \equiv y^{1/2} \bmod q$.

   A знает все 4 корня квадратных из $(y \bmod p, y \bmod q)$ в $Z_p \times Z_q$: это $(\pm z_p, \pm z_q)$, знаки независимы. По изоморфизму из КТО $\Rightarrow$ A знает корни из y в Z_n: обозначим их $(\pm w, \pm\tilde{w})$. В одну из этих пар входит принятое значение x : $y \equiv x^2 \bmod n$. A не знает, в какую.
4. A пытается угадать, в какой паре есть x, выбранный B. A выбирает одну из пар, например $\pm w$, отправляет B.
5. Если A не угадал, т.е. $x \not\equiv \pm w \bmod n$, B выиграл. Он это доказывает, присылая A факторизацию $n = pq$.

Обоснование: если A не угадал, B знает все 4 корня из y: $\pm w, \pm\tilde{w}$. Справедливо тождество $w^2 - \tilde{w}^2 \equiv 0 \bmod n \Rightarrow (w - \tilde{w})(w + \tilde{w}) \equiv 0 \bmod n \Rightarrow w - \tilde{w}$ кратно одному из p, q и НОД$(w - \tilde{w}, n) \in \{p, q\}$. (Иначе $w + \tilde{w}$ кратно $pq \Rightarrow w = -\tilde{w}$.) T.e. B узнал факторизацию n.

23 Разделение секрета

23.1 Схема интерполяционных полиномов Лагранжа

Это пороговая схема с параметрами (t, n): n - число долей, на которые был разделен секрет, t - число долей, которые необходимы, чтобы его восстановить. Автор схемы - Шамир (Shamir).

Пусть $M \in \mathbb{M}$ - секрет из множества допустимых секретов. Выберем простое $p : p > n$, $p > \max\limits_{M \in \mathbb{M}} M$.

Чтобы разделить секрет, создадим произвольный полином степени $t - 1$ над некоторым полем. Например, для $(3, n)$ схемы - полином степени 2. При этом свободный член - это всегда наш секрет M:
$F(x) = (ax^2 + bx + M) mod\ p$.

Значения a, b - случайные, секретные, после разделения секрета не нужны. Значение p известно всем.

Доли секрета получаются путем вычисления полинома в n точках: $K_i = F(x_i)$. Другими словами, первая доля может быть получена как $F(1)$, вторая $F(2)$ и т.д.

Для восстановления полинома степени $(t-1)$ по его значениям, необходимо и достаточно знать его значения в t точках.

Пример 23.1. Можно создать схему, когда секрет делится между 5 участниками, и любые 3 могут его восстановить.

Пример 23.2. Можно создать схему, когда секрет делится между N участниками, при этом первый участник может его восстановить, скооперировавшись с любым из остальных. Решение: например, первому сообщаем $t-1$ долей секрета, остальным по одной. Варьируя количество долей у каждого, а также t, можно, например, добиться, что участие первого необходимо для раскрытия секрета: он получает $t/2+1$ долю, остальные - по одной.

Пример 23.3. Пусть есть 2 группы людей, A и B. Хотим, чтобы набор любых двух людей из A и трех из B вместе могли восстановить секрет. Решение: полином - произведение линейной и квадратичной функций. Значения линейной функции в ряде точек сообщают людям из A, значения квадратичной - из B. Только собравшись вместе, они могут восстановить секрет.

23.2 Векторная схема Blakley

.

Секрет - точка в t-мерном пространстве. Каждая доля - уравнение гиперплоскости размерности $t-1$ или менее, содержащей эту точку. Пересечение определенного числа гиперплоскостей (не параллельных, не совпадающих) однозначно определяет точку.

Пример 23.4. 3 доли. 3-мерное пространство. Доля - уравнение плоскости. Две доли определяют прямую. Все три доли определяют точку.

Существуют и другие схемы.

Часть VII

Коды, исправляющие ошибки

Я РАССКАЗАЛ ЭТУ ЧАСТЬ НА ЛЕКЦИЯХ, Т.К. МНЕ НРАВИТСЯ ИДЕЯ СКРЫТОГО ПОЛУЧЕНИЯ ИНФОРМАЦИИ. ВОЗМОЖНО, В МЕТОДИЧКЕ ОНА СМОТРИТСЯ ЛИШНЕЙ И ЕЕ ЛУЧШЕ ИСКЛЮЧИТЬ?

24 Блоковые линейные коды

Три дальнейших раздела построены следующим образом. Даются основные понятия о двух классах кодов, исправляющих ошибки: линейных блоковых кодах и локально декодируемых кодах. Последние используются для построения криптографического протокола скрытого получения информации. Основная цель этого раздела - дать базовые сведения о блоковых линейных кодах, исправляющих ошибки.

Пусть бинарная информация передается по каналу передачи данных, в котором существует помеха. Т.е. при передаче данных возможна аддитивная ошибка. Вероятность ошибки в каждом бите $p < 1/2$. Ошибки независимы.

Для исправления ошибки потребуется передавать избыточную информацию - кодировать исходные данные. Кодируется либо всё сообщение целиком, либо оно разбивается на блоки и каждый блок кодируется отдельно. Размер блока - любой.

При описании кодов, исправляющих ошибки, все векторы - это строки. Обозначим множество $B = \{0, 1\}$.

Определение 24.1. *Расстояние Хэмминга* между бинарными векторами $\forall x, y \in B^n$ $\rho(x, y) = \sum\limits_{i=1}^{n} |x_i - y_i|$.

Определение 24.2. *Вес* вектора $x \in B^n$ - это $w(x) = \rho(x, 0^n)$, число единичных координат x.

Определение 24.3. *Двоичный блоковый код* (binary block code) - это отображение $B^k \to C \subset B^n$, $n > k$. При этом k/n - *скорость передачи* кода.

Определение 24.4. $d = \min\limits_{c_1, c_2 \in C} \rho(c_1, c_2)$ - кодовое расстояние (code distance).

Очевидно, код с расстоянием d замечает $d-1$ ошибку и исправляет $[(d-1)/2]$ ошибок.

Пример 24.1. простейший код, исправляющий одну ошибку:
$0 \to 000, 1 \to 111$.

Декодирование также идет поблочно. Считаем, что число ошибок в блоке не превосходит $[(d-1)/2]$. Пусть $y \in B^n$ - принятый вектор. Цель - найти кодовое слово $x : y = x \oplus e : x = \arg\min\limits_{x \in C} w(e)$.

Способы декодирования рассмотрим далее.

Определение 24.5. *Линейным блоковым кодом длины* n называется линейное подпространство C линейного пространства B^n.

Замечание. $0 \in C$.

Способы задания подпространства: через его базис или через базис ортогонального подпространства.

Определение 24.6. Пусть H - двоичная матрица размера $(n-k) \times n$. *Линейным блоковым кодом длины* n *с проверочной матрицей* H (check matrix) называется множество $C = \{x \in B^n | Hx^T = 0\}$.

Параметры линейного блокового кода:

- n - длина кода
- k - размерность подпространства C, размерность кода
- кодовое расстояние d линейного кода C равно $\min\limits_{x \in C, x \neq 0} w(x)$.

Код с этими параметрами называется $[n, k, d]$ кодом.

Теорема 24.1. Код C с проверочной матрицей H имеет кодовое расстояние $d \Leftrightarrow \forall d-1$ столбцов матрицы H линейно независимы, но $\exists\, d$ линейно зависимых столбцов.

Доказательство. $0 \in C \Rightarrow \forall y \in B^n : 1 \leq w(y) \leq d-1$ $y \notin C$, т.е. $Hy^T \neq 0$, любые $d-1$ столбцов H линейно независимы. Без ограничения общности, $\exists x \in C : w(x) = d$. Тогда верно $Hx^T = 0$. $\Rightarrow$ в матрице H $\exists\, d$ линейно зависимых столбцов. $\square$

Определение 24.7. *Порождающей матрицей* (generating matrix) линейного $[n, k, d]$ кода C называется матрица G размера $k \times n$, строками которой являются базисные векторы линейного пространства C. С помощью порождающей матрицы код можно представить в виде $C = \{x = aG | \forall a \in B^k\}$.

T.e. кодирование - суть умножение справа на матрицу G.

Задача 24.1. Пусть G, H - порождающая и проверочная матрицы кода C. Доказать, что $HG^T = GH^T = 0$.

Некоторые способы декодирования линейных кодов:

- декодирование к ближайшему кодовому слову по таблицам для B^n на практике не возможно из-за экспоненциального объема данных;
- мажоритарное декодирование - путем записи нескольких систем линейных уравнений для битов кодового слова и выбора значений i-го бита из них: если большинство систем дает ответ a, то i-й бит кодового слова равен a. Это медленный способ.
- декодирование с помощью синдрома.

Определение 24.8. Пусть $[n, k, d]$ код C имеет проверочную матрицу H. Пусть y - любой вектор длины n. *Синдромом* (syndrome) вектора y называется вектор $S(y) = Hy^T$.

Если $y \in C$, то $S(y) = 0$. Пусть $y = x \oplus e$, где $x \in C$, а e - вектор ошибок. Тогда $S(y) = He^T = \sum\limits_{i:e_i=1} H_i$ - сумма некоторых столбцов матрицы H.

Пусть из канала принят вектор $y = x \oplus e$. Его синдром $S(y) = S(e) = He^T$. Очевидно, $\forall e_1, e_2 : w(e_1), w(e_2) \leq t\ S(e_1) \neq S(e_2) \Leftrightarrow$ код может исправлять t ошибок.

Пусть $\forall y\ w(S(y)) < t = [(d-1)/2)]$. Вычислим синдромы $\forall e \in B^n : w(e) \leq t$. Результаты занесем в таблицу, сортированную по значению синдрома. Длина таблицы: $\sum\limits_{i=0}^{t} C_n^i$. Теперь, если принят вектор y, то вычисляем его синдром и находим в таблице вектор e, которому соответствует этот синдром. Тогда $x = e \oplus y$. T.e. декодирование идет в два шага: $y \to e \to x$.

Теорема 24.2 (Граница Синглтона)**.** Пусть C - $[n, k, d]$-код. Тогда $n - k \geq d - 1$.

Доказательство. Любую матрицу G размера $n \times k$ ранга k можно элементарными преобразованиями привести к виду $(I|M)$, где I - единичная матрица $k \times k$.

Тогда любое кодовое слово состоит из исходного слова с приписанными к нему проверочными символами. Вес кодового слова, в котором только один бит исходного слова не равен нулю, не превосходит $n - k + 1$. $\Rightarrow d \leq n - k + 1$. □

Более подробно теория линейных кодов изложена в [17], гл. 9, параграф 1.

25 Локально декодируемые коды

Основная цель раздела - кратко описать коды, обеспечивающие исправление определенной доли ошибок при кодировании сообщения целиком и возможность быстрого вычисления любого бита исходного сообщения.

Блоковые коды устроены так, что избыточность данных позволяет исправлять ошибки в блоке данных. Доступ к произвольному фрагменту длинного многоблочного сообщения происходит путем декодирования определенного блока.

Недостаток блоковых кодов - плохая устойчивость к концентрированным ошибкам. Пусть с помощью $[n, k, d]$ кода передается сообщение длиной несколько блоков. Пусть все блоки переданы без ошибок, а при передаче одного из них количество ошибок превысило $[\frac{d-1}{2}]$. Тогда этот блок будет декодирован неправильно. При этом количество ошибок может быть в произвольное количество раз меньше, чем длина сообщения.

Можно избавиться от этого недостатка, кодируя всё сообщение как один длинный блок кода, исправляющего ошибки. Тогда получим другой недостаток - невозможность доступа к произвольным данным без декодирования всего сообщения.

Так называемые локально декодируемые коды (ЛДК) обеспечивают эффективный доступ к произвольным данным и более высокую устойчивость к концентрированным ошибкам, чем у кодов с небольшими блоками. Декодирование одного бита сообщения происходит по данным о небольшом количестве случайно выбранных бит кодового слова. Цена - больший объем вычислений и меньшая скорость передачи данных, чем у классических блоковых кодов, исправляющих ошибки.

Рассмотрим некоторые локально декодируемые коды.

Определение 25.1. *Локально декодируемый код* (locally decodable code) $C \subset B^n$ с параметрами (r, δ, ε) - это взаимно однозначное отображение $B^k \to C$ со следующими свойствами. Пусть принятое слово $y = C(x) + e$ имеет до δn ошибок, т.е. δ - максимальная доля ошибок. Тогда значение любого бита x_i, $1 < i < k$, исходного сообщения может быть верно декодировано с вероятностью $(1 - \varepsilon)$ рандомизированной процедурой декодирования, которая использует значения каких-либо r бит принятого слова y.

Замечание. Декодирование рандомизировано только относительно случайных чисел, генерируемых на стороне декодера. (Т.е. верно для всякого искажения кодового слова.)

25.1 Код Адамара

Обозначим $[n]$ - это множество $\{1, .., n\}$.

Определение 25.2. *Код Адамара* (Hadamard code) с параметрами $(2, \delta, 2\delta)$ кодирует k-битовые сообщения в кодовые слова длиной 2^k по следующему правилу: каждый бит кодового слова равен $XOR_{i \in S}(x_i)$ над одним из подмножеств S множества $[k]$.

Декодирование происходит следующим образом. Пусть y - искаженное кодовое слово x. На вход декодера подается пара y, i. Декодер выбирает случайным образом, равновероятно, множество $S \subseteq [k]$ и получает значения y_q, y_t в позициях, соответствующих S и $S \oplus \{i\}$. δ - вероятность искажения одного бита кодового слова. Значит, вероятность того, что оба запроса попали в неискаженные биты, равна $(1 - 2\delta)$.

Таким образом, код Адамара обеспечивает быстрое локальное декодирование (использует 2 бита принятого слова), но имеет экспоненциальную длину кодового слова, т.е. нагрузка на сервер БД по вычислениям и по памяти - экспоненциальная.

25.2 Код Рида - Маллера

Код Рида-Маллера - это q-арный код: каждая буква сообщения и кодового слова может принимать одно из q значений. Код определяется 3 параметрами:

- размер алфавита $q = p^n$ - степень простого числа. Используется поле $GF(q)$.
- степень полинома f от многих переменных над $GF(q)$ - число $d < q - 1$.
- число переменных полинома n.

Рассмотрим код Рида-Маллера как локально декодируемый код.

Пусть $m = (x_1, .., x_k)$ - сообщение. Оно имеет длину $k = C^d_{n+d}$ символов из F_q. C^d_{n+d} - это количество разных одночленов степени от 0 до d от n переменных. Пусть $W = (w_1, .., w_k)$ - некоторые фиксированные векторы из F^n_q. Фиксируем полином: $f_m \in F_q[z_1, .., z_n]$ степени не более d такой, что $\forall i \in \{1, .., k\}$ $f(w_i) = x_i$. При определенном выборе W такой полином $\exists$ $\forall m \in F^k_q$. Коэффициенты полинома - это решение системы линейных уравнений $f_m(w_i) = x_i,\ i = 1, .., k$.

Кодирование - это отображение: $(x_1, ..., x_k) \to (f_m(a)\ \forall a \in F_q^n)$. Длина кодового слова равна q^n.

Декодирование: по номеру бита i и искаженному не более чем в $\delta \cdot q^n$ позициях кодовому слову, состоящему из значений полинома f_m в точках векторного пространства F_q^n, надо найти значение f_m в точке w_i.

Возьмем случайный вектор $v \in F_q^n$. Определим одномерное подпространство $L = \{w_i + \lambda v | \lambda \in F_q\}$. Таким образом, мы провели через точку w_i случайную прямую L.

На L полином f_m становится полиномом одной переменной λ, степень f_m по прежнему не более d. Значит, коэффициенты полинома $f_m(\lambda)$ можно однозначно восстановить по значения в $d+1$ точке.

Возьмем (потенциально искаженные) значения полинома f в случайно выбранной $d+1$ точке $L \backslash \{w_i\}$. Каждый запрос декодера идет в случайную, независимо выбранную точку. Какова вероятность того, что все $d+1$ точек будут без искажений? Это $1-(d+1)\cdot\delta$. $\Rightarrow$ код Рида-Маллера - это $(d+1, \delta, (d+1)\cdot\delta)$ локально декодируемый код.

25.3 Семейства локально декодируемых кодов

Коды с низкой длиной (сложностью) запроса. $r = const$ или $r < log(k)$. Такие коды используются в теоретической криптографии для скрытого получения информации. Скорость передачи информации низкая и уменьшается с увеличением длины сообщения, поэтому они пока не нашли практического применения. Например, код Адамара, некоторые коды Рида-Маллера и другие.

Коды с высокой длиной (сложностью) запроса. $r = k^\epsilon$ для некоторого $\epsilon > 0$. У этих кодов длина кодового слова пропорциональна длине сообщения (это коды с положительной скоростью передачи информации, positive rate codes). Начиная с 2010 года они используются в некоторых приложениях для передачи и хранения данных. Например, код Рида-Маллера с числом переменных $k_{RM} = 1/\epsilon$, δ, $r = \Theta(\delta)$ - локально декодируемый код с параметром $r = k^\epsilon$. А также мультипликативные коды.

Введением в теорию локально декодируемых кодов может служить статья [18]. Подробнее эта область освещена в книге [19].

26 Скрытое получение информации

Основная цель раздела: описать протоколы, которые при некоторых предположениях о базе данных (БД) позволяют получать информацию из БД, не сообщая ее владельцу, какая именно информация была запрошена.

Рассмотрим получение информации из БД. Владелец БД может мониторить запросы пользователей и узнать, чем интересовался отдельный пользователь. Цель - создать протокол, который не позволить это узнать. (Практических приложений пока нет.)

Определение 26.1. Схема *скрытого получения информации* (СПИ) (private information retrieval) заключается в следующем:

- БД - строка X из n бит.
- $S_1, .., S_k$ - реплицированные, не общающиеся друг с другом сервера БД.
- i - номер бита в строке X, значение которого хотим узнать.
- Выберем несколько случайных чисел ("бросим монету"), запросим сервера, вычислим значение X_i по ответам серверов. Каждый запрос случаен, не зависит от i. $\Rightarrow$ каждый сервер не получает никакой информации об i. Запросы к БД - не обязательно запросы о значениях некоторых битов. Это могут быть запросы о вычислении определенных функций от некоторых битов, например, XOR.

Исторически, основной параметр эффективности схемы скрытого получения информации - максимальная сложность в смысле числа переданных по каналу бит. Максимум берется по всем вариантам значений строки X и значения генератора случайных чисел. На практике, объемы вычислений и памяти также являются ограничивающими факторами.

26.1 Вычислительно стойкие схемы СПИ

Рассмотрим вычислительно стойкие схемы СПИ, т.е. стойкие против злоумышленника, использующего алгоритмы не более чем полиномиальной сложности. Их свойства:

- владелец БД должен решить сложную задачу, чтобы узнать, что запрашивал пользователь

- не требуют репликации БД
- для работы схемы требуется большой объем вычислений на сервере БД

Опишем схему, построенную на основе проблемы распознавания квадратичных вычетов по модулю $m = p_1 p_2$, где p_1, p_2 - простые.

Пусть БД - это строка X длины $n = s^2$, хранится в виде квадратной матрицы (x_{ij}).

Клиент хочет получить значение некоторого бита x_{ij}. Он выбирает произвольное большое число $m = p_1 p_2$, где p_1, p_2 - простые. Затем создает $s-1$ квадратичных вычетов $\{a_t \in Z_m^* | 1 < t < s, t \neq j\}$ и один квадратичный невычет $b_j \in Z_m^*$. Клиент передает серверу m и вектор $\{a_1, .., b_j, .., a_s\}$. Тот принимает этот вектор как набор $u_1, .., u_s$.

Сервер возвращает набор $\pi_1, .., \pi_s$: $\pi_i = \prod\limits_{k=1..s} u_t^{x_{ik}} \; mod \; m$, $i = 1..s$. Заметим, что если $x_{ij} = 0$, то в произведении π_i нет квадратичных невычетов, иначе - есть один невычет. Клиент вычисляет x_{ij} следующим образом: если сервер вернул $\pi_i \in QR(m)$, то $x_{ij} = 0$, иначе $x_{ij} = 1$. Сложность по объему переданных данных: $O(\sqrt{n})$. При каждом запросе клиента серверу приходится проводить довольно сложные вычисления, их объем - полином от размера базы данных n.

26.2 Абсолютно стойкие схемы СПИ

Рассмотрим схемы СПИ, стойкие с информационно-теоретической точки зрения. Т.е. владелец БД, даже имея неограниченные ресурсы, не должен суметь получить никакой информации о запросе пользователя.

В случае одного сервера это невозможно, единственный способ - получение всей БД. Но существует эффективный протокол для реплицированных, не обменивающихся информацией серверов. Каждый отдельный сервер не получает информацию о том, что интересно пользователю.

Все современные схемы СПИ строятся следующим образом: строится ЛДК и конвертируется в схему СПИ. Очевидно, набор r индексов, используемых в запросах при декодировании ЛДК, не должен быть постоянно из некоторой окрестности какого-либо бита принятого слова.

Определение 26.2. ЛДК, у которых распределение запросов является равномерным, называются *абсолютно гладкими* (perfectly smooth).

Построим r-серверную схему СПИ по (r, δ, ε) абсолютно гладкому ЛДК.

Пусть C - абсолютно гладкий ЛДК, кодирующий слова длины k в кодовые слова длины n. Предварительные вычисления: серверы $S_1, .., S_r$ кодируют БД x в n-битовое кодовое слово $C(x)$.

Клиент выбирает набор из r случайных запросов $q_1, .., q_r$ таких, что он может вычислить x_i по $C(x)_{q_1}, .., C(x)_{q_r}$. Отправляет по одному запросу к каждому серверу. Получает ответы: $C(x)_{q_j}$ и вычисляет по ним значение x_i.

Каждое значение q_j - реализация равномерно распределенной случайной величины, поэтому серверы не получают никакую информацию о запросе клиента.

Введением в теорию скрытого получения информации может служить статья [20]. Более подробно связь между локально декодируемыми кодами и СПИ освещена в [21].

Литература

[1] J.Katz and Y.Lindell. *Introduction to Modern Cryptography, Second Edition.* 2014.

[2] Рябко и Фионов. *Криптографические методы защиты информации.* Москва, 2005.

[3] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography.* 1st. Boca Raton, FL, USA: CRC Press, Inc., 1996. ISBN: 0849385237.

[4] Dan Boneh. *Cryptography I.* 2014. URL: `https://www.coursera.org/course/crypto`.

[5] *Letter Frequency.* URL: `http://en.wikipedia.org/wiki/Letter_frequency`.

[6] Goldreich O. *Foundations of Cryptography (Basic Tools).* Cambridge University Press, 2004. ISBN: 0-511-04120-9.

[7] Chang D. and Nandi M. *A Short Proof of the PRP/PRF Switching Lemma.* 2008. URL: `http://eprint.iacr.org/2008/078`.

[8] Duong Hieu Phan and Pointcheval D. "About the security of ciphers (semantic security and pseudo-random permutations)". In: *Proceedings of Selected Areas in Cryptography (SAC'04), Volume 3357 of LNCS.* Springer-Verlag, 2004, pp. 182–197.

[9] Jeff Atwood. *The Danger of Naivete*. 2007. URL: http://blog.codinghorror.com/the-danger-of-naivete/.

[10] Жуков А.Е. *Системы блочного шифрования*. 2006.

[11] Biham E. and Shamir A. "Differential Cryptanalysis of DES-like Cryptosystems". In: *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '90. London, UK, UK: Springer-Verlag, 1991, pp. 2–21. ISBN: 3-540-54508-5. URL: http://dl.acm.org/citation.cfm?id=646755.705229.

[12] Bellare M., Canetti R., and Krawczyk H. "Keying Hash Functions for Message Authentication". In: *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '96. London, UK, UK: Springer-Verlag, 1996, pp. 1–15. ISBN: 3-540-61512-1.

[13] Mihir Bellare and Chanathip Namprempre. "Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm". In: *J. Cryptol.* 21.4 (Sept. 2008), pp. 469–491. ISSN: 0933-2790. DOI: 10.1007/s00145-008-9026-x. URL: http://dx.doi.org/10.1007/s00145-008-9026-x.

[14] Rene Schoof. *Four primality testing algorithms*. 2008. URL: http://arxiv.org/abs/0801.3840.

[15] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. "PRIMES is in P". In: *Ann. of Math* 2 (2002), pp. 781–793.

[16] Yiannis Tsiounis and Moti Yung. "On the Security of ElGamal Based Encryption". In: *Public Key Cryptography, First International Workshop on Practice and Theory in Public Key Cryptography, PKC '98, Pacifico Yokohama, Japan, February 5-6, 1998, Proceedings*. 1998, pp. 117–134. DOI: 10.1007/BFb0054019. URL: http://dx.doi.org/10.1007/BFb0054019.

[17] Р. Лидл и Г. Нидеррайтер. *Конечные поля. В 2-х томах. Том 2*. Москва, «Мир», 1988.

[18] Sergey Yekhanin. "Locally Decodable Codes: A Brief Survey". In: *Coding and Cryptology - Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings*. 2011, pp. 273–282. DOI: 10.1007/978-3-642-20901-7_18. URL: http://dx.doi.org/10.1007/978-3-642-20901-7_18.

[19] Sergey Yekhanin. "Locally Decodable Codes". In: *Foundations and Trends in Theoretical Computer Science* 6.3 (2012), pp. 139–255. DOI: 10.1561/0400000030. URL: http://dx.doi.org/10.1561/0400000030.

[20] Sergey Yekhanin. "Private information retrieval". In: *Commun. ACM* 53.4 (2010), pp. 68–73. DOI: 10.1145/1721654.1721674. URL: http://doi.acm.org/10.1145/1721654.1721674.

[21] Sergey Yekhanin. *Locally Decodable Codes and Private Information Retrieval Schemes*. Information Security and Cryptography. Springer, 2010. ISBN: 978-3-642-14357-1. DOI: 10.1007/978-3-642-14358-8. URL: http://dx.doi.org/10.1007/978-3-642-14358-8.