Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Томский государственный университет систем управления и радиоэлектроники»

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

Утверждаю:
Зав. каф. КИБЭВС
д–р техн. наук, профессор
_____ А. А. Шелупанов
«__» _____ 2014 г.

Методические указания для выполнения практических и самостоятельных работ

«Криптографические методы защиты информации»

для студентов направлений подготовки
10.00.00 Информационная безопасность

Разработчик:
Доцент каф. КИБЭВС
канд. техн. наук
_____ О. О. Евсютин
«__» _____ 2014 г.

Томск 2014

Содержание курса

Введение 3
Глава 1. Основные понятия и определения 4
Глава 2. Математические основы криптографии 8
2.1 Алгебраические структуры. Группы 8
2.2. Кольца 12
2.3. Поля 16
2.4. Элементы теории чисел 17
Глава 3. Историческая криптография 20
Глава 4. Современные симметричные шифры 25
Глава 5. Криптография с открытым ключом 40
Глава 6. Хеширование 49
Глава 7. Коды аутентичности сообщений. Электронная подпись 55
Глава 8. Управление ключами. Распределение симметричных ключей 60
Глава 9. Инфраструктура открытого ключа 61
Список литературы 64
Вопросы для самоконтроля 65
Задачи 66

Введение

Целью дисциплины «Криптографические методы защиты информации» является формирование у студентов общих представлений о криптографических методах защиты информации, о применении криптографических методов защиты информации для решения отдельных задач обеспечения информационной безопасности и об основных принципах, лежащих в основе функционирования криптографических средств защиты информации.

Задачи изучения дисциплины:

- дать представление о криптографических методах защиты информации;
- изучить математические основы современной криптографии;
- изучить современные стандарты симметричного шифрования;
- изучить основные криптографические алгоритмы с открытым ключом;
- изучить криптографические функции хеширования;
- сформировать умение применять полученные знания для компьютерной реализации криптографических алгоритмов.

Практические занятия

№ п/п	Тематика практических занятий (семинаров)	Трудоемкость (час.)
1.	Алгебраические структуры. Группы. Циклические группы.	2
2.	Кольца, кольца классов вычетов.	2
3.	Конечные поля, поля Галуа.	2
4.	Теоретико-числовые алгоритмы, используемые в криптографии	2
5.	Простейшие шифры и их криптоанализ.	4
6.	Современные симметричные шифры	2
7.	Защита индивидуального задания по теме «Программная реализация простейшего шифра»	2
8.	Протокол Диффи-Хеллмана	2
9.	Криптосистема RSA	2
10.	Защита индивидуального задания по теме «Программная реализация симметричного шифра»	2
11.	Криптосистема Эль-Гамаля	2
12.	Криптосистема Рабина	2
13.	Защита индивидуального задания по теме «Программная реализация асимметричного шифра»	2

Самостоятельная работа

№ п/п	Виды самостоятельной работы *(детализация)*	Трудоемкость (час.)	Контроль выполнения работы
1.	Проработка лекционного материала	14	Устный опрос на лекциях
2.	Подготовка к практическим занятиям	10	Устный опрос на практических занятиях
3.	Выполнение индивидуальных заданий	20	Защита отчетов по индивидуальным заданиям

Глава 1. Основные понятия и определения

Комплекс мер по обеспечению информационной безопасности включает в себя следующие составляющие: организационную, правовую, инженерно-техническую, программно-аппаратную и криптографическую (специальную). Последней из перечисленных составляющих посвящена дисциплина «Криптографические методы защиты информации» (КМЗИ).

Целью первой лекции по данной дисциплине является рассмотрение основных понятий, относящихся к криптографическим методам защиты информации, а также тех задач из области обеспечения информационной безопасности, которые могут быть решены с их помощью.

Для начала введем основные определения.

Криптография — это наука, занимающаяся поиском и исследованием математических методов преобразования информации с целью ее защиты.

Криптоанализ — это наука, занимающаяся исследованием методов получения доступа к зашифрованной информации без знания секретного ключа. Кроме того, под криптоанализом понимается любая попытка найти уязвимость в криптографическом алгоритме или протоколе.

Отрасль математики, включающая в себя криптографию и криптоанализ, называется *криптологией*.

Криптографические методы защиты информации можно разделить на три основных направления: *шифрование*, *хеширование* (*хеширование*) и *электронную подпись*. Каждое из данных направлений отвечает за решение одной или нескольких задач обеспечения информационной безопасности, поэтому на практике все указанные методы используются в комплексе.

Теперь перечислим основные задачи, решением которых занимается криптография, давая там, где это необходимо, краткие пояснения.

1. Обеспечение *конфиденциальности (секретности)* — обеспечение невозможности несанкционированного ознакомления с содержанием информации.

2. Обеспечение *целостности* — обеспечение невозможности несанкционированного изменения информации.

3. Обеспечение *аутентификации (в широком смысле)* — подтверждение подлинности сторон и самой информации.

4. Обеспечение возможности подтверждения авторства и невозможности отказа от него.

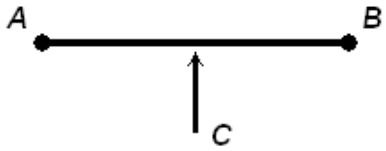
Есть и другие задачи, дополняющие перечисленные, но они являются основными.

Теперь рассмотрим методы, с помощью которых достигается решение этих и других задач.

Шифрование

Шифрование отвечает за обеспечение конфиденциальности, то есть секретности передаваемых и/или хранимых данных, что означает невозможность их несанкционированного прочтения.

На рисунке 1 представлена простейшая схема информационного обмена, в котором участвуют две стороны, традиционно обозначаемые как *A* и *B*, и присутствует третья сторона – злоумышленник *C*.



Канал передачи информации

Такой канал передачи информации называется открытым или незащищенным, так как противник может получить к нему доступ. Для обеспечения конфиденциальности информации ее необходимо передавать в зашифрованном виде.

В данной ситуации мы оперируем следующими понятиями.

Открытый текст (*исходный текст*, *сообщение*) – подлежащие защите данные.

Шифртекст (*криптограмма*) – защищенные данные, полученные посредством зашифрования открытого теста.

Зашифрование – преобразование открытого текста в шифртекст.

Расшифрование – обратное преобразование шифртекста в исходный текст.

Здесь следует сделать небольшое отступление. В литературе процесс преобразования шифртекста в исходный текст часто называют *дешифрованием*. Это неверное использование терминологии, поскольку дешифрование – это понятие из области криптоанализа, попытка получения исходной информации без знания секретного ключа. Если же речь идет о законном пользователе, используется термин расшифрование.

Кроме того, говоря о терминологии, следует обратить внимание еще на один момент. Иногда шифрование данных называют *кодированием*, что в принципе неверно, так как кодирование – это преобразование данных от одного вида в другой, представление информации в более удобной в данной ситуации форме. И защиту информации кодирование не подразумевает.

Таким образом, шифрование как криптографический метод защиты информации относится к совокупности процессов зашифрования и расшифрования, или, по-другому, означает использование некоторого шифра для обеспечения конфиденциальности данных.

Данное понятие является ключевым, как в рассматриваемом направлении, так и в криптографии в целом.

Шифром (*криптосистемой*) называют совокупность криптографических алгоритмов зашифрования и расшифрования вместе с алгоритмами генерации соответствующих ключей, где *ключ* – это секретный элемент данных, знание которого позволяет законному пользователю осуществлять зашифование и расшифрование данных, и незнание которого не позволяет злоумышленнику прочитать зашифрованные данные.

Рассмотрим математическую модель шифра, для чего введем следующие обозначения.

A – алфавит, то есть конечное множество используемых для кодирования информации знаков. В качестве примера алфавита можно привести русский алфавит Z_{33} (32 буквы русского алфавита, исключая “ё”, и пробел) или двоичный алфавит $Z_2 = \{0,1\}$.

X – множество исходных текстов.

Y – множество шифртекстов.

В общем случае $X = Y = \bigcup_{i=1}^{l} A^i$, то есть каждый элемент множества открытых текстов и множества шифрованных текстов представляет собой некоторую последовательность символов алфавита.

K – множество ключей.

E_k – алгоритм зашифрования на ключе k.

$E = \{E_k \mid k \in K\}$ – множество преобразований открытого текста в шифртекст при всех возможных значениях ключа.

$E(x) = \{E_k(x) \mid k \in K\}$ – множество преобразований открытого текста *x* в шифртекст при всех возможных значениях ключа.

$E_k(x) = \{E_k(x) \mid x \in X\}$ – преобразование открытого текста *x* в шифртекст при заданном значении ключа *k*.

$D_k : E_k(x) \to x$ – обратное преобразование шифртекста в открытый текст при заданном значении ключа *k*.

$D = \{D_k \mid k \in K\}$ – множество обратных преобразований шифртекста в открытый текст при всех возможных значениях ключа.

Таким образом, математическая модель шифра включает в себя множество всех возможных открытых текстов, множество соответствующих им шифртекстов, ключевое множество, множество преобразований открытого текста в шифртекст и множество обратных преобразований шифртекста в открытый текст.

$\Sigma = (X, Y, K, E, D)$ – математическая модель шифра.

Существуют различные классификации алгоритмов шифрования. Если классификацию проводить по тому, какие ключи используются при шифровании и расшифровании информации, то можно выделить криптосистемы двух типов:

1) Симметричные криптосистемы — для шифрования и расшифрования используется один и тот же ключ.

2) Асимметричные криптосистемы (криптосистемы с открытым ключом) — ключи шифрования и расшифрования различны.

Хеширование

Хеширование – это преобразование входной битовой строки произвольной длины в выходную битовую строку фиксированной длины. Соответственно, хеш-функция – это функция, принимающая на вход строку битов произвольной длины и возвращающая на выходе строку битов заданной длины. Значение хеш-функции называют хеш-значением, хеш-кодом, сверткой, а также дайджестом сообщения.

Смысл хеширования заключается в получении характеристического признака входных данных, по которому эти данные можно впоследствии идентифицировать. Таким образом, хеширование предназначено для обеспечения контроля целостности передаваемых и/или хранимых данных.

Хеш-функции, использующиеся в криптографии должны быть однонаправленными и защищенными от коллизий.

Однонаправленная хеш-функция защищена в вычислительном отношении от восстановления прообразов. Это означает, что по элементу *Y* из множества значений хеш-функции *h* невозможно подобрать такое значение *x* из множества определения, при котором $h(x) = Y$.

Коллизией (а также повторением или столкновением) по отношению к функциям хеширования называют ситуацию, когда для двух различных значений m_1 и m_2 $h(m_1) = h(m_2)$. Очевидно, что любая хеш-функция обладает бесконечным числом коллизий, поскольку представляет собой отображение бесконечного множества битовых строк произвольной длины в конечное множество битовых строк заданной длины. Защищенность от коллизий заключается в том, что хотя коллизии и существуют, их вычислительно невозможно обнаружить.

Основной принцип проектирования хеш-функции заключается в том, что ее действие должно сопровождаться лавинным эффектом – это означает, что небольшое изменение входных данных приводит к значительному изменению выходных данных.

Рассмотрим два типа атак на целостность данных, которые можно отразить, используя хеширование.

1) Имитация.

Злоумышленник может перехватить сообщение пользователя A, составить новое сообщение, вычислить его хеш-значение и отправить результат пользователю B. Чтобы отразить такую атаку, пользователи A и B используют известный только им секретный ключ аутентификации, не зная которого, злоумышленник для данного открытого текста M не может вычислить $H_k(M) = N$.

Следует добавить, что нарушение целостности может быть как умышленным, так и случайным, произошедшим, например, вследствие аппаратного сбоя или помех на канале связи. Криптография рассматривает первый случай – целенаправленные попытки изменения передаваемой информации, поскольку для защиты от случайных изменений можно использовать менее сложные средства.

2) Подмена.

Злоумышленник, зная данный открытый текст M, и зная $H_k(M) = N$, может попытаться подобрать другой текст M_1 для которого $H_k(M_1) = N$. Однако хеш-функции строятся таким образом, что осуществить подбор вычислительно невозможно.

Электронная подпись

Согласно федеральному закону «Об электронной подписи» электронная подпись – это информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом связана с такой информацией и которая используется для определения лица, подписывающего информацию.

Электронная подпись является аналогом рукописной подписи и обладает теми же свойствами для электронных документов, что и рукописная подпись для бумажных.

1) Подпись достоверна. Она убеждает получателя документа в том, что подписавший сознательно подписал документ.

2) Подпись неподдельна. Она доказывает, что именно подписавший, и никто иной, сознательно подписал документ.

3) Подпись не может быть использована повторно. Она является частью документа, жулик не сможет перенести подпись на другой документ.

4) Подписанный документ нельзя изменить. После того, как документ подписан, его невозможно изменить.

5) От подписи не возможно отречься. Подпись и документ материальны. Подписавший не сможет впоследствии утверждать, что он не подписывал документ.

Электронная подпись предназначается для идентификации лица, подписавшего электронный документ, а также позволяет осуществлять контроль целостности данных, гарантировать невозможность отказа от авторства или, напротив, гарантированно подтверждать авторство.

Глава 2. Математические основы криптографии

2.1 Алгебраические структуры. Группы

Множества.

Данная лекция является первой в небольшом цикле, охватывающем математические основы современной криптографии. Дальнейшее изложение предполагает знакомство с основами теории множеств и разделом дискретной математики, рассматривающим бинарные отношения, поэтому для начала давайте вспомним понятие множества и основные операции над множествами. Что касается бинарных отношений, то к данному понятию мы обратимся несколько позже, когда оно нам понадобится.

Итак, что мы называем множеством? Можно сформулировать различные определения, поэтому предлагаю остановиться на следующем: множество — это совокупность объектов, объединенных неким общим признаком. Например, множество действительных чисел, или множество студентов в аудитории.

Здесь нелишним будет вспомнить стандартные обозначения числовых множеств, а именно: N — множество натуральных чисел, Q — множество рациональных чисел, R — множество действительных чисел, C — множество комплексных чисел.

Перечисленные обозначения будут использоваться в дальнейшем.

Алгебраические операции.

Введем базовые понятия и определения.

Определение. Будем говорить, что на множестве X задана алгебраическая операция $*$, если любой упорядоченной паре элементов $x, y \in X$ поставлен в соответствие однозначно определенный элемент $z \in X$, который обозначается $z = x * y$. Множество X с заданной на нем алгебраической операцией $*$ будем называть алгебраической структурой и обозначать $(X; *)$.

Из определения следует, что характеристическими признаками алгебраической операции являются всюдуопределенность, однозначность и замкнутость. В качестве примера алгебраической операции можно привести операцию сложения, заданную на множестве целых чисел, тогда соответствующую алгебраическую структуру обозначим $(Z; +)$. В свою очередь, операция деления, заданная на том же множестве, алгебраической операцией не является, поскольку не выполняются признаки всюдуопределенности и замкнутости.

Однако обратите внимание, что не нужно ассоциировать операцию $*$ с известными вам арифметическими операциями сложения и умножения, поскольку таким образом мы обозначаем некоторую абсолютно абстрактную операцию, которая может быть задана в том числе и таблично. Приведем пример для множества $X = \{a, b, c\}$.

$*$	a	b	c
a	a	c	a
b	b	a	a
c	c	b	a

Теперь рассмотрим основные свойства, которыми может обладать та или иная алгебраическая операция.

1. Ассоциативность.

Определение. Будем говорить, что алгебраическая операция $*$, заданная на множестве X, имеет свойство *ассоциативности* (является *ассоциативной*), если для любой тройки элементов $x, y, z \in X$ выполнятся соотношение $x * (y * z) = (x * y) * z$. При этом алгебраическая структура, построенная на основе ассоциативной алгебраической операции, называется *полугруппой*.

2. Существование нейтрального элемента

Определение. Пусть $(X;*)$ есть алгебраическая структура. Элемент $e \in X$ называется нейтральным элементом, если для любого $x \in X$ выполняется $e * x = x * e = x$.

Определение. Полугруппа с нейтральным элементом называется моноидом.

3. Свойство квазигруппы

Определение. Алгебраическая структура $(X;*)$ называется квазигруппой, если для любых $a, b \in X$ однозначно разрешимы уравнения $a * x = b$ и $y * a = b$.

4. Существование обратимых элементов.

Определение. Пусть $(X;*)$ есть алгебраическая структура с нейтральным элементом $e \in X$. Будем говорить, что элемент $y \in X$ является обратным элементом для элемента $x \in X$, если $x * y = y * x = e$. Обратный элемент для $x \in X$, называемого в этом случае обратимым, будем обозначать x^{-1}.

5. Коммутативность

Определение. Алгебраическая операция $*$ на множестве X называется коммутативной, если для любых элементов $a, b \in X$ справедливо равенство $a * b = b * a$.

Группы.

После того, как было введено понятие алгебраической операции и рассмотрены ее возможные свойства, рассмотрим особый класс алгебраических структур, называемых группами. С такого рода алгебраическими структурами мы столкнемся неоднократно при изучении различных криптографических алгоритмов.

Определение. Алгебраическая структура $(G;*)$ называется группой, если она одновременно является полугруппой и квазигруппой.

Для краткости группу $(G;*)$ будем обозначать просто G, а групповую операцию называть умножением, используя мультипликативную форму записи, то есть $(G;\cdot)$.

Теперь сформулируем определение группы, эквивалентное введенному ранее, но раскрывающее некоторые другие свойства данной алгебраической структуры.

Определение. Алгебраическая структура $(G;\cdot)$ является группой, если выполняются следующие условия:

1) заданная операция умножения является ассоциативной;

2) относительно заданной операции умножения в G существует нейтральный элемент, обозначаемый 1_G и называемый единицей группы;

3) каждый элемент в G является обратимым относительно заданной операции умножения.

Если групповая операция является коммутативной, то такая группа называется коммутативной или *абелевой*. Для абелевых групп принято использовать аддитивную форму записи, когда групповая операция записывается как сложение, нейтральный элемент обозначается как 0, а элемент, обратный элементу группы a, обозначается как $-a$.

В качестве примера группы можно привести множество целых чисел с заданной на нем операцией сложения. В свою очередь, по операции умножения данное множество группу не образует.

Теперь снова обратимся к элементам теории множеств. Как вам известно, с понятием множества тесно связано понятие подмножества, которое неформально можно определить как множество, все элементы которого являются также и элементами исходного множества. Аналогичным образом в группе могут выделяться подгруппы. Однако не каждое подмножество множества, на котором задана группа, является подгруппой.

Определение. Подмножество H группы G называется подгруппой, если H само является группой относительно той же операции, что и G. Данный факт обозначается следующим образом: $H \le G$.

Почему произвольно взятое подмножество множества, на котором задана группа, не обязательно является подгруппой данной группы? Дело в том, что групповая операция, вводимая не для всех элементов исходного множества, может лишиться признака замкнутости и, таким образом, выбранное подмножество не только не будет являться группой, но и в принципе не будет алгебраической структурой. Например, рассмотрим все ту же группу целых чисел по сложению $(Z; +)$.

Определить, является ли некоторое подмножество группы ее подгруппой, можно с помощью критерия подгруппы, который сформулирован ниже в виде теоремы.

Теорема. Подмножество H группы G есть подгруппа в G тогда и только тогда, когда выполняется условие: для любых $x, y \in H$ выполнено $xy^{-1} \in H$.

Примем данную теорему без доказательства.

Целочисленные степени элементов группы.

Введем понятие целочисленной степени элемента группы.

Определение. Пусть G — группа и $g \in G$ — произвольный элемент. Для любого целого числа n определим n-ую степень элемента g следующим образом:

$$\begin{cases} \underbrace{g \cdot g \cdot \ldots \cdot g}_{n}, & \text{если } n > 0, \\ \underbrace{g^{-1} \cdot g^{-1} \cdot \ldots \cdot g^{-1}}_{n}, & \text{если } n < 0, \\ 1, & \text{если } n = 0. \end{cases}$$

Обратите внимание, что представленное определение подразумевает под целочисленной степенью элемента группы g^n не показатель степени n, а результат возведения элемента g в данную степень.

Целочисленные степени элементов группы обладают следующими двумя свойствами: для любых целых чисел m и n выполняется $g^m \cdot g^n = g^{m+n}$ и $\left(g^m\right)^n = g^{mn}$.

Введем новое обозначение.

Пусть G — группа и $g \in G$ — произвольный элемент. Обозначим через $\langle g \rangle$ множество всевозможных целочисленных степеней элемента g, т. е. $\langle g \rangle = \left\{\ldots, g^{-2}, g^{-1}, g^0 = 1_G, g^1, g^2, \ldots\right\}$.

Согласно критерию подгруппы данное множество представляет собой подгруппу группы G. Будем говорить, что $\langle g \rangle$ есть подгруппа группы G, порожденная элементом g, а сам элемент g будем называть образующим подгруппы $\langle g \rangle$.

Циклические группы.

Отсюда приходим к следующему понятию.

Определение. Группа G называется циклической группой, если $G = \langle g \rangle$ для некоторого $g \in G$, который называется образующим циклической группы G.

Циклические группы бывают двух типов.

Если все целочисленные степени образующего элемента g в циклической группе $\langle g \rangle$ различны, то в этом случае говорят, что $\langle g \rangle$ есть бесконечная циклическая группа.

Если же некоторые целочисленные степени элемента g с различными показателями степени совпадают, т. е. $g^m = g^n$ для некоторых целых чисел $m \neq n$, то группа $\langle g \rangle$ является конечной циклической группой.

Это означает, что последовательно возводя образующий такой группы в различные целочисленные степени, мы получим все возможные элементы группы и завершим данный процесс получением единицы группы, после чего цикл может быть повторен.

Наименьшее натуральное число k такое, что $g^k = 1_G$, назовем порядком элемента g (это обозначается как $O(g) = k$), а группу $\langle g \rangle$, в свою очередь, назовем конечной циклической группой порядка k. Соответственно, порядок бесконечной циклической группы равен бесконечности.

Очевидно, что порядок циклической группы совпадает с количеством различных элементов в ней, т. е. ее мощностью. Таким образом, порядок циклической группы с одной стороны равен порядку ее образующего, с другой — ее мощности. $|G| = O(g)$.

Поясним введенные понятия на примерах.

Пример. Множество целых чисел по сложению является бесконечной циклической группой с образующим 1.

Пример. Рассмотрим циклическую группу G, состоящую из целочисленных степеней матрицы $g = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$. Покажем, что порядок данной циклической группы равен 4.

Теперь введем еще ряд определений и теорем, которые также примем без доказательств.

Определение. Пусть G — группа и $g \in G$ — произвольный элемент. Назовем подгруппу $\langle g \rangle$ группы G циклической подгруппой, порожденной элементом g. Порядок данной подгруппы равен порядку элемента g.

Следующая теорема имеет достаточно важное значение для криптографии.

Теорема (теорема Лагранжа). Пусть G есть конечная группа, $|G| = n$. Если H есть подгруппа группы G и $|H| = k$, то для некоторого натурального числа s имеем $n = k \cdot s$.

Следствием из данной теоремы является то, что для любого элемента $g \in G$ имеет место $g^n = 1$. Другими словами, любой элемент группы, возведенный в степень, равную порядку данной группы, дает единицу группы.

Теорема. Пусть $G = \langle x \rangle$ есть циклическая группа порядка n. Элемент $g = x^k \in G$ является образующим группы G тогда и только тогда, когда выполняется условие $\text{НОД}(k, n) = 1$.

Теорема. Любая подгруппа конечной циклической группы G является циклической группой.

Следующая теорема является обратной к теореме Лагранжа.

Теорема. Пусть $G = \langle x \rangle$ есть циклическая группа порядка n. Если d есть делитель n, то существует и единственная подгруппа H группы G такая, что $|H| = d$.

Сформулированные сейчас теоремы понадобятся нам несколько позднее, когда мы будем рассматривать широко используемые в криптографии циклические группы определенного типа.

Группы подстановок.

Теперь рассмотрим один из известных классов групп — группы подстановок, называемые также симметрическими группами. Для этого введем следующее определение.

Определение. Пусть $N_k = \{1, 2, \ldots, k\}$ — отрезок натурального ряда чисел. Назовем подстановкой на множестве N_k произвольную биекцию на N_k.

Здесь напомню, что биекцией (биективным отображением) называется такое отображение $f : X \to Y$, которое обладает двумя следующими свойствами:

1) любым двум отличным между собой элементам из области определения X ставятся в соответствие также отличные между собой элементы из области значений Y (свойство инъективности), т. е. $\forall\, x_1, x_2 \in X$, где $x_1 \neq x_2$, выполняется $f(x_1) \neq f(x_2)$;

2) для любого элемента $y \in Y$ найдется такой элемент $x \in X$, что $f : x \mapsto y$ (свойство сюръективности).

Практический смысл использования биективных отображений в криптографии заключается в том, что такие отображения являются обратимыми, что позволяет использовать их для построения криптосистем, поскольку, как вам уже известно, расшифрование представляет собой обращение зашифрования.

Подстановка записывается в виде таблицы из двух строк, в которой верхняя строка представляет собой отрезок натурального ряда чисел $N_k = \{1, 2, \ldots, k\}$, а нижняя — некоторую перестановку элементов верхней строки. Собственно, в этом и проявляется биективность такого отображения.

Пример. Подстановка $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 2 & 5 & 3 \end{pmatrix}$ является биекцией на N_5, которая отображает 1 в 4, 2 в 1, 3 в 2, 4 в 5 и 5 в 3.

Теперь рассмотрим множество всевозможных подстановок на N_k и зададим на данном множестве алгебраическую операцию умножения как композицию подстановок. Известно, что композиция отображений ассоциативна, следовательно заданная алгебраическая операция является ассоциативной. При этом во множестве подстановок на N_k существует нейтральный элемент — так называемая тождественная подстановка $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$. И, наконец, каждая подстановка является обратимой: для ее обращения достаточно поменять местами верхнюю и нижнюю строки, и упорядочить столбцы по возрастанию элементов верхней строки.

Таким образом, множество подстановок на N_k относительно умножения образует группу, которая называется симметрической группой k-ой степени S_k.

2.2. Кольца

Кольца.

Рассмотрев понятие группы, перейдем к алгебраической структуре более сложного типа, называемой кольцом.

Дадим определение.

Определение. Будем говорить, что на множестве K задана структура кольца (или короче: K есть кольцо), если на множестве K заданы две алгебраические операции, которые принято обозначать + (сложение) и · (умножение), причём выполняются следующие свойства:

1) $(K; +)$ есть абелева группа;

2) $(K;\cdot)$ есть полугруппа;

3) имеет место двоякая дистрибутивность умножения относительно сложения, т. е. для любых $a, b, c \in K$ выполнятся $\begin{cases} a(b+c) = ab + ac \\ (a+b)c = ac + bc \end{cases}$.

Нейтральный элемент абелевой группы $(K;+)$, которую принято называть аддитивной группой кольца K, принято обозначать 0_K или просто 0 и называть нулём кольца K.

Полугруппу $(K;\cdot)$ принято называть мультипликативной полугруппой кольца K.

Если в мультипликативной полугруппе $(K;\cdot)$ кольца K есть нейтральный элемент, то его принято обозначать 1_K или просто 1 и называть единицей кольца K. В этом случае говорят, что K есть кольцо с единицей.

Если умножение в кольце K коммутативно, то говорят, что K — коммутативное кольцо.

Пример. Примером кольца является множество целых чисел относительно арифметических операций сложения и умножения. Вообще говоря, понятие кольца возникло как обобщение свойств целых чисел. В связи с этим множество целых чисел называют также кольцом целых чисел. Будем считать данные понятия взаимозаменяемыми.

Для колец с единицей имеет смысл следующее определение.

Определение. Пусть K — кольцо с единицей. Будем говорить, что элемент $u \in K$ является обратимым элементом, если существует элемент $u^{-1} \in K$, такой, что $u \cdot u^{-1} = u^{-1} \cdot u = 1$.

Множество обратимых элементов кольца с единицей K, обозначаемое K^*, является группой относительно умножения.

Обратите внимание, что когда речь идет об обратимости элементов кольца, обычно подразумевается обратимость относительно операции умножения, поскольку относительно сложения все элементы кольца обратимы по определению.

Пример. В кольце целых чисел группа обратимых элементов имеет вид $Z^* = \{-1, 1\}$.

Теперь вспомним понятие подгруппы, вводимое для группы, и введем аналогичное понятие для кольца.

Определение. Подмножество L кольца K называется подкольцом, если L является кольцом относительно операций сложения и умножения в кольце K. Данный факт обозначается следующим образом: $L \le K$.

Сформулируем теорему, дающую критерий кольца.

Теорема. Подмножество L кольца K является подкольцом в K тогда и только тогда, когда выполняются следующие условия:

1) для любых $x, y \in L$ имеем $x - y \in L$,

2) для любых $x, y \in L$ имеем $x \cdot y \in L$.

Кольца классов вычетов.

Теперь рассмотрим класс колец, нашедший широкое применение в криптографии, — кольца классов вычетов. Но прежде вспомним понятия бинарного отношения и отношения эквивалентности.

Определение. Бинарным отношением ρ между множествами X и Y называется произвольное подмножество декартового произведения X и Y, $\rho \subseteq X \times Y$.

Другими словами, любое бинарное отношение ρ между X и Y есть некоторое множество упорядоченных пар (x, y), где $x \in X$, $y \in Y$. Тот факт, что $(x, y) \in \rho$ будем

записывать так: $x\rho y$ и читать — «x находится в отношении ρ к y». Если $X = Y$, то будем говорить, что ρ есть бинарное отношение на множестве X.

Введем следующее определение.

Определение. Бинарное отношение ρ на множестве X называется отношением эквивалентности, если выполнены следующие три условия для любых $x, y, z \in X$:

1) рефлексивность, когда $x\rho x$;

2) симметричность, когда из $x\rho y$ следует $y\rho x$;

3) транзитивность, когда из $x\rho y$ и $y\rho z$ следует $x\rho z$.

Определение. Пусть ρ есть отношение эквивалентности на множестве X. Для любого $x \in X$ обозначим $\bar{x} = \{y \in X \mid y\rho x\}$ и назовем $\bar{x}$ классом эквивалентности элемента $\bar{x}$.

Неформально говоря, когда на множестве вводится отношение эквивалентности, это означает, что множество разбивается на некоторое количество непересекающихся подмножеств — классов, причем все элементы отдельно взятого класса обладают неким общим признаком, т. е. некоторым образом эквиваленты друг другу, отсюда и классы эквивалентности.

Теперь вернемся к кольцам классов вычетов.

Зададим на множестве целых чисел Z бинарное отношение ρ_n для произвольного натурального числа n следующим образом: для любых $a, b \in Z$ пусть $a\rho_n b \Leftrightarrow n \mid (a-b)$.

Очевидно, что любые целые a и b, находящиеся в отношении ρ_n друг к другу, имеют одинаковые остатки при делении на n. Будем обозначать этот факт следующим образом: $a \equiv b (\mathrm{mod}\, n)$ (a сравнимо с b по модулю n).

Можно показать, что введенное бинарное отношение является отношением эквивалентности, поскольку выполняются условия рефлексивности, симметричности и транзитивности.

Тогда множество целых чисел может быть разбито на классы эквивалентности относительно ρ_n. Множество таких классов эквивалентности обозначим Z_n и назовем их классами вычетов по модулю n.

Целые числа, принадлежащие одному классу вычетов по модулю n, имеют одинаковый остаток при делении на n, т. е. класс вычетов $\bar{a}$ описывается как $\bar{a} = \{x \in Z \mid x = kn + a, k \in Z\}$. При этом обозначается класс вычетов по своему наименьшему представителю. Поскольку для каждого натурального n остатками при делении на него могут являться целые числа 0, …, $n-1$, общее число которых составляет n, то множество Z_n состоит из n элементов.

Пример. Возьмем множество Z_4.

Таким образом, мы определили разбиение множества целых чисел на подмножества, называемые классами вычетов и состоящие из целых чисел, имеющих одинаковые остатки при делении на n, для любого натурального n.

Зададим на данном множестве структуру кольца.

Для любых классов вычетов $\bar{a}, \bar{b} \in Z_n$ пусть $\bar{a} + \bar{b} = \overline{a+b}$ и $\bar{a} \cdot \bar{b} = \overline{ab}$. Относительно сложения множество Z_n образует абелеву группу, относительно умножения — полугруппу, кроме того имеет место двоякая дистрибутивность умножения классов вычетов относительно их сложения. Наличие данных свойств у операций сложения и умножения классов вычетов обусловлено тем, что ими обладают операции сложения и умножения целых чисел, являющихся представителями этих классов.

Таким образом, мы пришли к понятию кольца классов вычетов по модулю n, которое обозначим $(Z_n; +; \cdot)$ или просто Z_n. Отметим, что для любого n кольцо Z_n является коммутативным кольцом с единицей.

Теперь рассмотрим некоторые вопросы исследования конкретных колец классов вычетов. Причем для нас с точки зрения криптографии наибольший интерес представляют свойства групп обратимых элементов данных колец. Группу обратимых элементов кольца классов вычетов по модулю n обозначим Z_n^*.

Теорема. Ненулевой элемент кольца классов вычетов по модулю n $\bar{0} \neq \bar{s} \in Z_n$ является обратимым элементом тогда и только тогда, когда $\text{НОД}(s, n) = 1$.

Исследование группы обратимых элементов кольца классов вычетов по модулю n позволяет выяснить, в числе прочего, является ли данная группа циклической и какие подгруппы в ней можно выделить. Ответ на данные вопросы сводится к нахождению порядков составляющих группу Z_n^* элементов.

Кольца многочленов

Рассмотрим еще один класс колец — кольца многочленов.

Определение. Пусть K — коммутативное кольцо с единицей. Назовём многочленом над кольцом K произвольную конечную последовательность элементов $(a_0, a_1, \ldots, a_n)$ кольца K. Если $p = (a_0, a_1, \ldots, a_n)$ — многочлен и $a_n \neq 0$, то n называется степенью многочлена p ($\deg(p) = n$), a_n называется старшим коэффициентом многочлена p, причём если $a_n = 1$, то многочлен называется нормированным. Многочлен нулевой степени $p = (a_0)$ называется константой.

Введем символ X как многочлен первой степени вида $X = (0, 1)$, и обозначим $K[X]$ множество всех многочленов над кольцом *K*. В качестве операций сложения и умножения в данном множестве примем операции сложения и умножения многочленов, реализуемые так же, как аналогичные операции для обычных многочленов. Относительно данных операций множество $K[X]$ является коммутативным кольцом с единицей. Это утверждение оставим без доказательства.

Теперь введем определение, которое пригодится нам в дальнейшем.

Определение. Если $u, v \in K[X]$, такие, что $u = q \cdot v$, причём $\deg(v) < \deg(u)$, то многочлен v называется собственным делителем многочлена *u*. Будем говорить, что многочлен *u* неприводим, если u не имеет собственных делителей, причём $\deg(v) \geq 1$.

Очевидно, что многочлены степени 1 неприводимы.

Обратите внимание, что неприводимый многочлен в кольце многочленов является аналогом простого числа в кольце целых чисел.

Теперь рассмотрим операцию деления с остатком, заданную на множестве $K[X]$ и реализуемую так же, как аналогичная операция для обычных многочленов. Выбрав произвольный многочлен p, отличный от нулевого, и разделив на него с остатком все элементы множества $K[X]$, мы приходим к разбиению множества многочленов $K[X]$ (кольца многочленов) на классы эквивалентности, объединяющие в себе многочлены, имеющие одинаковый остаток при делении на p. Данное множество классов эквивалентности образует кольцо относительно операций сложения и умножения по аналогии с кольцом классов вычетов, получаемым при разбиении на классы эквивалентности множества целых чисел. Назовем такое кольцо кольцом многочленных вычетов.

2.3. Поля

Поля

Можно увидеть, что рассматриваемые нами алгебраические структуры образуют определенную иерархию, выстраиваемую от простого к сложному. Продолжим рассмотрение данной иерархии алгебраических структур и введем новое понятие.

Определение. Кольцо K, в котором все ненулевые элементы образуют абелеву группу относительно умножения, называется полем.

Из определения следует, что группа обратимых элементов K^* поля K состоит из всех ненулевых элементов поля K и называется мультипликативной группой поля K.

Заметим, что кольцо K является полем тогда и только тогда, когда K есть коммутативное кольцо с единицей, в котором каждый ненулевой элемент обратим.

Пример. В качестве примера поля можно привести множество действительных чисел с заданными на нем операциями сложения и умножения.

Введем определение подполя.

Определение. Подмножество L поля K называется подполем, если L само есть поле относительно операций поля K.

Конечные поля

Для криптографии представляют интерес так называемые конечные поля. Введем определение.

Определение. Поле, построенное на основе множества, состоящего из конечного числа элементов, называется конечным полем

Самым простым примером конечного поля является кольцо классов вычетов по модулю простого числа. Сформулируем теорему.

Теорема. Кольцо Z_n является полем тогда и только тогда, когда $n = p$ есть простое число.

Не вводя строгого, поясним справедливость данной теоремы. Ранее нами была сформулирована теорема, дающая критерий обратимости произвольного элемента кольца классов вычетов. Согласно данному критерию ненулевой элемент кольца классов вычетов по модулю n $\bar{0} \neq \bar{s} \in Z_n$ является обратимым элементом тогда и только тогда, когда $\text{НОД}(s, n) = 1$. Нетрудно заметить, что, если n является простым числом, то все целые числа от 1 до $n-1$ взаимно просты с ним, и таким образом, соответствующие классы вычетов являются обратимыми.

Такого рода поля обозначаются F_p.

Следующий факт имеет большое значение в криптографии.

Теорема. Мультипликативная группа Z_p^* поля Z_p, где p — простое число, является циклической группой порядка $p-1$.

Определение. Пусть F есть конечное поле и α есть образующий группы F^*, т.е. $F^* = \alpha$. Элемент $\alpha \in F^*$ назовём примитивным элементом поля F.

Теперь рассмотрим более сложный класс конечных полей, называемый полями Галуа по имени исследовавшего их французского математика.

Определение. Полем Галуа назовем поле F_{p^n}, полученное расширением поля F_p посредством неприводимого многочлена $f(x) \in F_p[x]$ степени n. Мощность поля Галуа составляет p^n, то есть $\left|F_{p^n}\right| = p^n$.

Теперь введем некоторые пояснения. Элементами поля Галуа являются многочлены, принадлежащие множеству многочленов над полем F_p, степень которых не превышает n. Напомню, что принадлежность многочлена множеству многочленов над полем F_p означает, что коэффициенты при степенях данного многочлена являются элементами поля F_p. Другими словами, поле Галуа F_{p^n} состоит из всевозможных остатков при делении многочленов, заданных над полем F_p, на неприводимый многочлен $f(x) \in F_p[x]$ степени n.

Очевидна аналогия между полем Галуа и кольцом классов вычетов по модулю простого числа.

Рассмотрим на конкретном примере алгоритм построения поля Галуа как расширения поля F_p посредством неприводимого многочлена $f(x) \in F_p[x]$ степени n.

Пример. Пусть $p = 3$ и $n = 2$. Возьмем в качестве неприводимого многочлена многочлен $f(x) = 2x^2 + x + 1$. Проверим, что данный многочлен действительно является неприводимым. Для малых значений p это можно сделать простым перебором значений элементов поля F_p: если ни одно из них не является корнем многочлена $f(x)$, то он является неприводимым.

Теперь запишем все элементы поля F_{p^n}.

Необходимо определить операции сложения и умножения элементов данного поля. Сделаем это в виде таблиц.

Таблица сложения строится легко и просто.

Построение таблицы умножения осуществляется несколько сложнее. Как только при перемножении двух элементов поля Галуа, степень которых меньше n, появляется многочлен степени n и более, его необходимо привести по модулю $f(x)$. Сделать это можно поделив данный многочлен на $f(x)$, и взяв остаток от деления.

Однако, проще воспользоваться другим подходом. Приравняв $f(x)$ к нулю, выразим старшую степень x^n. Данное значение будем подставлять каждый раз, как у нас появится x^n.

После того как таблица умножения была построена, определим образующий мультипликативной группы. Сделать это можно с помощью таблицы умножения.

2.4. Элементы теории чисел

Теперь пришло время рассмотреть некоторые элементы теории чисел, используемые в современной криптографии. Как известно, теория чисел — это раздел математики, изучающий целые числа. Рассмотрим основные моменты, которые пригодятся нам в дальнейшем изложении.

Наибольший общий делитель

Для начала вспомним понятие наибольшего общего делителя (НОД) и некоторые ему сопутствующие.

Определение. Пусть $a, b \in Z$. Назовём наибольшим общим делителем целых чисел a и b такое целое число $d \geq 1$, которое удовлетворяет следующим условиям:

1) d есть общий делитель a и b;

2) если $d' \in Z$ есть любой общий делитель a и b, то d делится на $d' \in Z$.

Наибольший общий делитель чисел a и b принято обозначать НОД(a,b). Если НОД$(a,b)=1$, то a и b называются взаимно простыми числами.

Определение. Целое число p, делители которого исчерпываются числами ±1 и ±p, называется простым числом.

Обычно принято в качестве простых чисел рассматривать положительные простые числа, которые больше единицы.

Сформулируем теорему, называемую основной теоремой арифметики.

Теорема (основная теорема арифметики).

Каждое натуральное число $n > 1$ может быть записано в виде произведения простых чисел, не обязательно различных, а именно:

$n = p_1 p_2 \ldots p_k$,

причём эта запись единственна с точностью до порядка сомножителей.

Возникает вопрос: каким образом вычисляется НОД двух целых чисел? Можно разложить каждое из чисел на простые множители, и таким образом прийти к ответу. Но задача разложения целого числа на простые множители (задача факторизации) является трудной вычислительной задачей и решать с её помощью более лёгкую задачу нахождения НОД не имеет смысла.

Поэтому рассмотрим эффективные алгоритмы вычисления НОД целых чисел, которые не требуют умения факторизовать числа.

Рассмотрим вначале деление с остатком в кольце целых чисел.

Теорема. Для заданных $a, b \in \mathbb{Z}$, $b > 0$ существуют $q, r \in \mathbb{Z}$, такие, что $a = bq + r$, $0 \le r < b$.

Теорема. В кольце $\mathbb{Z}$ для любых двух целых чисел a и b существует d = НОД(a, b). Более того, существуют целые числа u, v, такие, что $au + bv = d$.

Данной запись называется целочисленной линейной комбинацией a и b.

Рассмотрим алгоритм, предназначенный для вычисления НОД целых чисел.

Алгоритм Евклида.

Вход: целые числа $a \ge b > 0$.

Выход: d = НОД(a, b).

Шаг 1. Пока $b \ne 0$, выполнить следующее:

Шаг 1.1. Вычисляем $r \leftarrow a \bmod b$, $a \leftarrow b$, $b \leftarrow r$.

Шаг 2. Возврат (a).

Рассмотрим пример вычисления НОД с помощью данного алгоритма.

Пример. a = 4864, b = 3458.

Если, кроме НОД(a, b), нужно найти также и целочисленную линейную комбинацию a и b, равную НОД(a, b), то применяется расширенный алгоритм Евклида, который выглядит следующим образом.

Расширенный алгоритм Евклида.

Вход: целые числа $a \ge b > 0$.

Выход: d = НОД(a, b) и целые x, y, такие, что $ax + by = d$.

Шаг 1. Полагаем $x_2 \leftarrow 1$, $x_1 \leftarrow 0$, $y_2 \leftarrow 0$, $y_1 \leftarrow 1$.

Шаг 2. Пока b > 0, выполнить следующее:

Шаг 2.1. q ←[a/b], r ← a − qb, x ← x2 − qx1, y ← y2 − qy1;

Шаг 2.2. a ← b, b ← r, x2 ← x1, x1 ← x, y2 ← y1, y1 ← y.

Шаг 3. d ← a, x ← x2, y ← y2 и возврат (d, x, y).

(Здесь [a] — целая часть числа a).

Пример работы расширенного алгоритма Евклида мы рассмотрим чуть позже. Отметим, что в криптографических приложениях расширенный алгоритм Евклида часто применяется для нахождения обратного элемента по модулю натурального числа. Алгоритм для этого выглядит следующим образом.

Алгоритм вычисления модулярного обратного элемента.

Вход: n > a > 0 — целые числа.

Выход: $a^{-1} \bmod n$.

Шаг 1. Используя расширенный алгоритм Евклида, находим целые числа x, y, такие, что nx + ay = d = НОД(a, n).

Шаг 2. Если d > 1, то $a^{-1} \bmod n$ не существует.

Шаг 3. Если d = 1, то ay ≡ 1(mod n) и, следовательно, $y = a^{-1} \bmod n$, тогда возврат (y).

Вот теперь рассмотрим пример.

Пример. Найти $13^{-1} \bmod 267$.

Сравнение первой степени с одним неизвестным.

Ещё одно применение расширенного алгоритма Евклида — решение сравнений первой степени.

Определение. Назовём сравнением первой степени с одним неизвестным сравнение вида $ax + b \equiv 0 (\bmod m)$.

Удобно записывать сравнения первой степени с учетом их свойств в виде $ax \equiv b (\bmod m)$. Заметим, что решение сравнений первой степени в кольце $\mathbb{Z}$ равносильно решению уравнений в кольце $\mathbb{Z}_m$.

Следующая теорема даёт ответ на вопрос о решениях сравнений первой степени с одним неизвестным.

Теорема. Сравнение вида $ax \equiv b (\bmod m)$ имеет следующие решения:

1) если НОД(a, m) = 1, то множество решений — класс вычетов $a^{-1} \cdot b \in \mathbb{Z}m$;

2) если НОД(a, m) = d >1 и d ∤ b, то сравнение не имеет решений;

3) если НОД(a, m) = d >1 и d делит b, то d классов вычетов по модулю m являются решениями сравнения, причём эти d классов образуют один класс вычетов по модулю m/d.

Функция Эйлера.

Рассмотрим еще некоторые теоретико-числовые понятия, применяющиеся в криптографии.

Определение. Функцией Эйлера ϕ(m) называется число натуральных чисел, не превосходящих m и взаимно простых с m.

Теорема. Пусть НОД(k, l) = 1, тогда $\phi(k \cdot l) = \phi(k) \cdot \phi(l)$.

Теорема. Если $m = p^k$, где p — простое число, то $\phi(p^k) = p^{k-1}(p - 1)$.

Теорема (теорема Эйлера). Пусть натуральное число a ∈ $\mathbb{Z}$n (то есть a ∈ {1, 2, …, n−1}). Если НОД(a, n) = 1, то имеет место сравнение $a^{\phi(n)} \equiv 1 (\bmod n)$.

Следствие (малая теорема Ферма). Если p — простое число, то для любого $0 \neq a \in \mathbb{Z}p$ имеем $a^{p-1} \equiv 1 (\bmod p)$.

Глава 3. Историческая криптография

Рассмотрев математические основы современной криптографии, перейдем непосредственно к соответствующим методам и алгоритмам, использующимся для решения задач обеспечения информационной безопасности. И начнем с небольшого исторического экскурса, в рамках которого рассмотрим некоторые исторические шифры, устаревшие и не использующиеся в настоящее время, однако построенные с соблюдением некоторых основных принципов, актуальных и для современной криптографии. Кроме того, знакомство с подобными шифрами поможет разделить надежное и ненадежное шифрование и выявить слабые места, которых не должно быть у современных криптографических алгоритмов.

Вообще говоря, криптография возникла очень давно, практически одновременно с письменностью. Как только появилась возможность сохранять слова в виде текста на материальных носителях информации, возникла необходимость защищать эту информацию от посторонних глаз. Правда, здесь стоит отметить тот факт, что когда умение читать и писать было уделом немногих избранных, отсутствовала такая уж острая необходимость проводить какие-либо дополнительные преобразования информации. В общем-то, точная дата появления криптографии не известна за давностью лет, но считается, что она зародилась в Древнем Египте — древнеегипетские жрецы таким образом защищали от непосвященных религиозные тексты.

Довольно много известно про древнегреческую криптографию, многие идеи, возникшие в ту эпоху, сохранились до наших дней, прежде перейдя по наследству к древнеримской цивилизации. С этого наследства мы, пожалуй, и начнем.

Однако, для начала введем классификацию исторических шифров, приняв в качестве классификационного признака способ организации криптографических преобразований. В этом случае все исторические шифры можно разделить на перечисленные ниже классы.

1. *Подстановочные шифры* (моноалфавитные и полиалфавитные).

Криптографическое преобразование заключается в замене символов открытого текста на другие (того же алфавита) по некоторому правилу.

2. *Перестановочные шифры*.

Криптографическое преобразование заключается в перестановке местами символов открытого текста по некоторому правилу.

3. *Блочные шифры*.

Открытый текст разбивается на блоки равной длины, одно и то же криптографическое преобразование применяется к каждому блоку.

4. *Шифры гаммирования*.

Гаммирование заключается в наложении на открытый текст некоторой псевдослучайной последовательности (гаммы), генерируемой на основе ключа шифрования. Под наложением гаммы на открытый текст обычно подразумевается сложение символов открытого текста с символами гаммы по модулю соответствующего алфавита. Однако в классических шифрах наложение гаммы может означать вычисление значений символов шифртекста на основе значений соответствующих символов открытого текста и гаммы по некоторому правилу.

Отметим, что это только одна из возможных классификаций исторических шифров, можно предложить и другие.

Подстановочный шифр

В качестве первого примера рассмотрим подстановочный шифр, называемый также шифром простой замены.

Опишем математическую модель данного шифра.

Каждой букве алфавита A мощностью m ($m=26$, если речь идет о латинском алфавите) ставится в соответствие число из диапазона $0\ldots m-1$. Другими словами, все символы алфавита нумеруются.

Множество возможных ключей шифра простой замены является симметрической группой степени m, то есть группой подстановок длины m: $K=S(A)=S_m$.

Открытый текст обозначим $x=(x_1,\ldots,x_l)$, где $x_i\in A$, $i=\overline{1,l}$, соответствующий шифртекст — $y=(y_1,\ldots,y_l)$.

Зашифрование открытого текста $x=(x_1,\ldots,x_l)$ на ключе $k\in K$ может быть записано как $E_k(x)=(k(x_1),\ldots,k(x_l))$, расшифрование шифртекста $y=(y_1,\ldots,y_l)$ на том же ключе — $D_k(y)=\left(k^{-1}(y_1),\ldots,k^{-1}(y_l)\right)$, где $k^{-1}\in K$ — подстановка, обратная k.

Проще говоря, при шифровании каждый символ текста заменяется на другой символ с помощью ключевой подстановки.

Известным частным случаем шифра простой замены является шифр Цезаря, названный так по имени использовавшего его всю жизнь древнеримского полководца. Данный шифр основан на использовании одного-единственного ключа — подстановки, полученной циклическим сдвигом элементов второй строки относительно первой на три позиции влево.

Аффинный шифр

Другим частным случаем шифра простой замены является аффинный шифр, основанный на так называемом аффинном преобразовании.

Рассмотрим математическую модель данного шифра, используя те же обозначения, что и ранее. Однако, теперь символы алфавита представим как элементы кольца классов вычетов Z_m.

Тогда можем записать $X=Y=\bigcup_{i=1}^{l}Z_m^i$. В качестве ключа аффинного шифра выступает пара значений $k=(\alpha,\beta)$, $\alpha\in Z_{26}^*$, $\beta\in Z_{26}$, соответственно ключевое пространство имеет вид $K=Z_{26}^*\times Z_{26}$.

Открытый текст и шифртекст обозначим соответственно $x=(x_1,\ldots,x_l)$ и $y=(y_1,\ldots,y_l)$, где $x_i\in Z_m$, $y_i\in Z_m$, $i=\overline{1,l}$.

Зашифрование отдельного символа открытого текста осуществляется по формуле $y_i=\alpha x_i+\beta$, $i=\overline{1,l}$, расшифрование — по формуле $x_i=(y_i-\beta)\alpha^{-1}$, $i=\overline{1,l}$.

Усилением аффинного шифра является аффинный рекуррентный шифр, когда для каждого символа открытого текста вычисляется новое ключевое значение на основе предыдущего. Для этого необходимо задать две ключевые пары $k_1=(\alpha_1,\beta_1)$, $k_2=(\alpha_2,\beta_2)$, и тогда ключевая пара для произвольного символа преобразуемой последовательности будет иметь вид $k_i=(\alpha_{i-1}\alpha_{i-2},\beta_{i-1}+\beta_{i-2})$, $i=\overline{3,l}$.

Перестановочный шифр

Следующий шифр, который мы рассмотрим — это перестановочный шифр.

В данном случае перед началом шифрования открытый текст разбивается на блоки одинаковой длины по l символов. В качестве ключа выступает подстановка аналогичной длины $k\in K=S_l$.

Отдельный блок открытого текста и соответствующий ему блок шифртекста обозначим $x=(x_1,\ldots,x_l)$ и $y=(y_1,\ldots,y_l)$ соответственно, где $x_i\in A$, $y_i\in A$, $i=\overline{1,l}$.

Зашифрование и расшифрование одного блока преобразуемой последовательности осуществляется по формулам $E_k(x) = (x_{k(1)}, \ldots, x_{k(l)})$ и $E_k(y) = (y_{k^{-1}(1)}, \ldots, y_{k^{-1}(l)})$ соответственно.

Шифр Хилла

Данный шифр построен на основе матричных преобразований.

Множество невырожденных квадратных матриц над кольцом классов вычетов по модулю *n* образует группу.

Открытый текст разбивается на блоки длиной n, и каждый блок представляется в виде n-мерного вектора.

$X = Y = (\mathrm{Z}_m)^n$.

Ключом является квадратная матрица размера n×n.

$K = GL_n(\mathrm{Z}_m)$.

$k = (k_{i,j}),\ i, j = \overline{1,n},\ k_{i,j} \in \mathrm{Z}_m$.

Эта матрица должна быть обратима в Z_m, чтобы была возможна операция расшифрования. Матрица будет являться обратимой только в том случае, если ее детерминант не равен нулю и не имеет общих делителей с основанием модуля.

$|k| \in \mathrm{Z}_m^*$.

Операция зашифрования заключается в том, что вектор, соответствующий блоку открытого текста, умножается на ключевую матрицу.

$x = (x_1, ..., x_n)^T$.

$$y = (y_1, ..., y_n)^T = E_k(X) = k\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

Для того, чтобы расшифровать шифртекст, необходимо, разбив его на блоки, представить каждый блок в виде вектора и умножить на обратную матрицу ключа.

В случае рекуррентного шифра Хилла для каждого блока открытого текста вычисляется новое ключевое значение на основе двух предыдущих.

$k_{i+1} = k_i k_{i-1}$.

$k_{i+1}^{-1} = k_{i-1}^{-1} k_i^{-1}$.

Шифры гаммирования

Рассмотрим класс симметричных криптосистем, криптографические преобразования в которых основываются на гаммировании.

Гаммирование заключается в наложении на открытый текст некоторой последовательности (гаммы), генерируемой на основе ключа шифрования. Под наложением гаммы на открытый текст обычно подразумевается сложение символов открытого текста с символами гаммы по модулю соответствующего алфавита. Однако в классических шифрах наложение гаммы может означать вычисление значений символов шифртекста на основе значений соответствующих символов открытого текста и гаммы по некоторому правилу.

Шифр на основе латинского квадрата

В качестве первого примера такого шифра мы рассмотрим шифр на основе латинского квадрата.

Пусть *A* — алфавит мощностью *n* (|*A*| = *n*).

Латинский квадрат — это квадратная матрица размера *n*×*n*, строками и столбцами которой являются некоторые перестановки алфавита *A*. Это означает, что в каждой строке и каждом столбце матрицы по одному разу встречаются все символы алфавита. При этом любой латинский квадрат является таблицей умножения квазигруппы.

Итак, зададим латинский квадрат на алфавите *A* мощностью *n*.

Алгоритмы в составе криптосистемы на основе латинского квадрата будут следующими:

1) Генерация ключей.

Задается гамма шифра — произвольная последовательность символов $\gamma_1\gamma_2\gamma_3...$, $\gamma_i \in A$, длина которой совпадает с длиной открытого текста.

2) Алгоритм зашифрования.

Гамма накладывается на открытый текст следующим образом: символ открытого текста соответствует столбцу латинского квадрата, символ гаммы — строке (либо наоборот), соответственно, значение символа шифртекста находится в ячейке латинского квадрата на пересечении этих столбца и строки.

$M = m_1m_2m_3...$

$\Gamma = \gamma_1\gamma_2\gamma_3...$

$---------$

$C = c_1c_2c_3...$

3) Алгоритм расшифрования.

При расшифровании та же гамма накладывается на шифртекст следующим образом: символ гаммы соответствует строке латинского квадрата, в этой строке ищется ячейка со значением, равным значению символа шифртекста, тогда символ открытого текста определяется столбцом, соответствующим найденной ячейке.

$C = c_1c_2c_3...$

$\Gamma = \gamma_1\gamma_2\gamma_3...$

$---------$

$M = m_1m_2m_3...$

Рассмотрим простой пример.

В качестве алфавита возьмем множество из трех элементов $A = \{a, b, c\}$. Зададим на этом множестве с помощью таблицы ассоциативную алгебраическую операцию, которую обозначим *.

*	*a*	*b*	*c*
a	*a*	*c*	*b*
b	*b*	*a*	*c*
c	*c*	*b*	*a*

Тогда функции шифрования соответствует выражение $\gamma_i * m_j = c_k$.

Шифр Виженера

Следующий шифр, который мы рассмотрим, это шифр Виженера.

Здесь, как и ранее мы рассматриваем некоторый алфавит *A* мощностью *n*. Этот алфавит заполняет первую строку латинского квадрата, а каждая последующая строка получается циклическим сдвигом влево предыдущей, то есть мы имеем группу вычетов по модулю *n* $(\mathrm{Z}_n;+)$.

Если вернуться к предыдущему примеру, где мы рассматривали алфавит из трех символов, то латинский квадрат будет выглядеть следующим образом.

*	*a*	*b*	*c*
a	*a*	*b*	*c*
b	*b*	*c*	*a*
c	*c*	*a*	*b*

Теперь запишем уравнения зашифрования и расшифрования.

$$\begin{cases} c_i = (m_i + \gamma_i) \bmod n \\ m_i = (c_i - \gamma_i) \bmod n \end{cases}$$

Преимущество шифра Виженера над рассмотренным ранее шифром на основе латинского квадрата заключается в том, что нет необходимости запоминать латинский квадрат, который теперь описывается двумя простыми уравнениями.

В классических шифрах на основе гаммирования в качестве ключа шифрования обычно использовалась короткая фраза, называемая лозунгом (паролем), которая циклически повторялась, формируя гамму.

Существует другой подход к формированию псевдослучайной ключевой последовательности – самоключ Виженера. Здесь в качестве начального ключа мы выбираем только один символ, к нему добавляем все символы открытого текста, за исключением последнего, и таким образом формируем гамму. Либо мы можем формировать гамму, добавляя к начальному символу поочередно символы шифртекста.

Шифр Вернама

В заключение данного раздела рассмотрим шифр гаммирования особого типа, называемый одноразовым шифр-блокнотом или шифром Вернама по имени его создателя. Мы еще не рассматривали вопросы стойкости криптографических алгоритмов, тем не менее сразу необходимо отметить, что при правильном использовании одноразовый шифр-блокнот обладает абсолютной стойкостью, несмотря на то, что относится к историческим шифрам и был создан в 1917 г.

Его описание крайне просто и может быть представлено без использования излишней математической формализации.

Если имеется некоторый открытый текст, составленный из символов алфавита *A*, то для его зашифрования необходимо задать гамму — последовательность символов того же алфавита, причем длина гаммы должна совпадать с длиной открытого текста. Видим, что в данном случае понятие пароля не имеет смысла, гамма и будет являться секретным ключом. Само зашифрование заключается в сложении символов открытого текста с соответствующими символами гаммы по модулю мощности алфавита, в результате чего получается шифртекст.

Основное требование, обеспечивающее упомянутую абсолютную стойкость данного шифра, звучит следующим образом: все возможные ключи являются равновероятными и ни один из них не может быть использован повторно. При соблюдении выдвинутого требования узнать открытый текст при неизвестном ключе нельзя даже с помощью полного перебора всех возможных значений ключа.

В дополнение отметим, что в современной криптографии под одноразовым шифр-блокнотом зачастую подразумевается вариант рассмотренного шифра для двоичного алфавита, когда наложение гаммы соответствует операции XOR над битовыми строками.

Некоторые вопросы стойкости криптографических алгоритмов, как исторических, так и современных, будут рассмотрены в соответствующем разделе.

Глава 4. Современные симметричные шифры

Общие сведения

Завершив изучение исторических шифров, мы приступим к изучению современных криптографических алгоритмов. И здесь необходимо отметить, что основное отличие между современными и историческими (классическими) шифрами, к последним из которых относятся все алгоритмы шифрования докомпьютерной эпохи (до 1960 г.), заключается в способе представления данных. Современные шифры обрабатывают данные в цифровом виде как битовые последовательности, природа которых не имеет значения, в то время как исторические шифры могли работать лишь с текстами, что накладывало определенные ограничения на использовавшийся математический аппарат.

До появления компьютерной криптографии данные, которые необходимо было защитить с помощью шифрования, практически всегда представляли собой тексты на естественном языке, обладающие высокой избыточностью. Именно высокая избыточность открытых текстов является причиной успешного применения различных статистических атак для взлома классических шифров. В связи с этим К. Шенноном были предложены два основных криптографических метода сокрытия избыточности открытого текста: перемешивание и рассеивание.

Перемешивание устраняет зависимость между открытым текстом и шифртекстом, затрудняет попытки найти в шифртексте избыточность и статистические закономерности. Перемешивание осуществляется с помощью подстановок (замен).

Рассеивание перераспределяет избыточность открытого текста, распространяя ее на весь шифртекст. Каждый бит открытого текста при шифровании должен повлиять на максимальное число других бит открытого текста. Простейшим способом рассеивания является перестановка.

По отдельности ни перемешивание, ни рассеивание не могут обеспечить надежного шифрования, поэтому классические подстановочные и перестановочные шифры неприменимы в современном мире, однако криптосистемы, в которых реализованы оба этих метода, обладают высокой криптостойкостью.

Таким образом, введенная ранее классификация исторических шифров оказывается неактуальной для шифров современных — современная криптография выделяет только два типа симметричных криптосистем: блочные и поточные (потоковые).

Блочные шифры обрабатывают открытый текст, разбивая его на блоки равного размера (обычно 64 бита и больше). Поточные шифры по своему устройству похожи на шифры гаммирования, рассмотренные нами на предыдущей лекции, — они преобразуют открытый текст в шифртекст последовательно по одному биту путем сложения битов открытого текста с битами ключевой последовательности по модулю 2. Но для формирования ключевой последовательности используется значительно более сложный математический аппарат, чем в шифрах гаммирования. Зацикливание начального ключевого значения для получения ключевой последовательности нужной длины или применение самоключей является недопустимым. Кроме того, отметим, что поточный шифр может быть построен на основе блочного шифра при использовании того в специальном режиме.

ГОСТ 28147-89

Изучение современных симметричных криптосистем мы начнем с российского стандарта шифрования ГОСТ 28147-89.

Данный стандарт описан в нормативном документе «ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования».

ГОСТ является блочным шифром, работающим с блоками длиной 64 бита. Ключ шифрования составляет 256 бит. Такая величина ключа более чем достаточна, чтобы полный перебор всего множества ключей был невозможен.

В ходе дальнейшего описания алгоритма шифрования будем использовать следующие обозначения.

КЗУ — ключевое запоминающее устройство, включающее 8 накопителей X_0, X_1, …, X_7 по 32 бита каждый.

$СМ_1$, $СМ_3$ — сумматоры, осуществляющие сложение по модулю 2^{32}.

$СМ_2$ — сумматор, осуществляющий сложение по модулю 2.

$СМ_4$ — сумматор, осуществляющий сложение по модулю $2^{32} – 1$.

$СМ_5$ — сумматор, осуществляющий сложение по модулю 2 без ограничения битовых строк.

N_1, N_2, N_3, N_4 — 32-разрядные накопители.

N_5, N_6 — накопители, хранящие константы.

K — блок замен.

R — блок циклического сдвига на 11 позиций влево (в сторону старшего разряда).

[+] — операция сложения по модулю 2^{32}.

$[+]^{|}$ — операция сложения по модулю $2^{32} –1$.

Блок замен представляет собой таблицу, состоящую из 8-ми строк и 16-ти столбцов, в ячейках которой находятся 4-битовые значения, которые можно представить в виде целых чисел от 0 до 15. Узлы блока замен (строки таблицы) в ряде источников называют S-блоками. Но мы это обозначение использовать не будем.

Таблица замен может использоваться при шифровании как дополнительный ключевой материал, но обычно она является параметром алгоритм шифрования, общим для определенных групп пользователей.

ГОСТ 28147-89 предназначен для работы в следующих режимах:

1) режим простой замены;

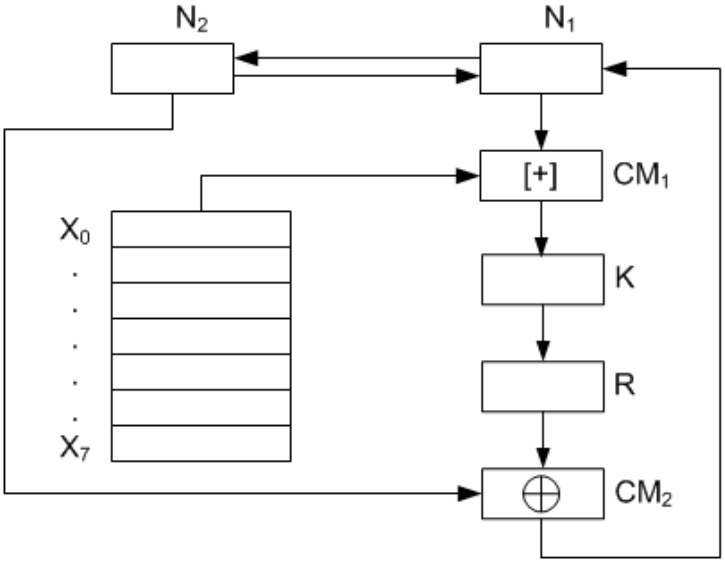
2) режим гаммирования;

3) режим гаммирования с обратной связью;

4) режим выработки имитовставки.

Базовым режимом работы ГОСТа является режим простой замены. В этом режиме блоки открытого текста последовательно и независимо друг от друга преобразуются в блоки шифртекста.

На схеме ниже представлено шифрование блока открытого текста в режиме простой замены.



Режим простой замены

Блок открытого текста длиной 64 бита обозначим T_O и разделим на две части по 32 бита. Биты правой части (младшие 32 бита) будем обозначать символом a с индексом, соответствующим номеру бита, биты левой части (старшие 32 бита) — символом b с индексом, соответствующим номеру бита.

$T_O = (b_1(0), b_2(0), \ldots, b_{32}(0); a_1(0), a_2(0), \ldots, a_{32}(0))$.

В приведенной записи значение в скобках после обозначения бита открытого текста означает номер итерации, то есть шага преобразования. Ноль означает неизмененные исходные данные, единица — данные после первой итерации и т. д.

В накопители N_1 и N_2 передаются соответственно младшие и старшие 32 бита блока открытого текста.

$(a_1(0), a_2(0), \ldots, a_{32}(0)) \rightarrow N_1$.

$(b_1(0), b_2(0), \ldots, b_{32}(0)) \rightarrow N_2$.

Содержимое накопителя N_1 складывается по модулю 2^{32} с элементом ключа. Полученное 32-битовое значение интерпретируется как массив из 8-ми 4-битовых блоков, значение каждого блока заменяется новым согласно таблице замен K. Замена блоков осуществляется следующим образом: новое значение блока находится в ячейке, которая стоит на пересечении строки с номером, равным номеру заменяемого блока, и столбца с номером, равным значению заменяемого блока.

Следующим шагом является циклический сдвиг результата предыдущего шага на 11 позиций влево. Затем полученное значение побитно складывается по модулю 2 с содержимым накопителя N_2.

После этого содержимое накопителя N_1 помещается в накопитель N_2, а значение, полученное в результате всех преобразований, помещается в накопитель N_1.

Перечисленные шаги повторяются 32 раза. Так как 256-битовый ключ разбивается на 8 32-битовых ключевых элементов, каждый из этих элементов используется по 4 раза. На шагах с 1-го по 8-ой, с 9-го по 16-ый и с 17-го по 24-ый последовательно используются ключевые элементы X_0, X_1, …, X_7. На шагах с 25-го по 32-ой ключевые элементы используются в противоположном порядке — X_7, X_6, …, X_0.

В результате всех преобразований мы получим блок шифртекста, который обозначим $T_{Ш}$.

В ходе обратного преобразования блока шифртекста в блок открытого текста выполняются те же самые преобразования, но ключевые элементы используются в обратном порядке. На шагах расшифрования с 1-го по 8-ой последовательно используются ключевые элементы X_7, X_6, …, X_0, а на шагах с 9-го по 16-ый, с 17-го по 24-ый и с 25-го по 32-ой — ключевые элементы X_0, X_1, …, X_7.

Теперь запишем уравнения зашифрования блока открытого текста, используя введенные ранее обозначения.

Для $j = \overline{1,24}$

$$\begin{cases} a(j) = R\left(K\left(a(j-1)[+]X_{(j-1)\bmod 8}\right)\right) \oplus b(j-1) \\ b(j) = a(j-1) \end{cases}$$

Для $j = \overline{25,31}$

$$\begin{cases} a(j) = R\left(K\left(a(j-1)[+]X_{32-j}\right)\right) \oplus b(j-1) \\ b(j) = a(j-1) \end{cases}$$

Для $j = 31$

$$\begin{cases} a(32) = a(31) \\ b(32) = R\left(K\left(a(31)[+]X_0\right)\right) \oplus b(31) \end{cases}$$

Уравнения расшифрования блока шифртекста запишутся аналогично.

Для $j = \overline{1,8}$

$$\begin{cases} a(32-j) = R\left(K\left(a(32-j+1)[+]X_{j-1}\right)\right) \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases}$$

Для $j = \overline{9,31}$

$$\begin{cases} a(32-j) = R\left(K\left(a(32-j+1)[+]X_{(32-j) \bmod 8}\right)\right) \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases}$$

Для $j = 32$

$$\begin{cases} a(0) = a(1) \\ b(0) = R\left(K\left(a(1)[+]X_0\right)\right) \oplus b(1) \end{cases}$$

В приведенных выражениях K и R — это функции, принимающие на вход и возвращающие на выходе 32-разрядное значение.

Следующий режим шифрования ГОСТ 28147-89, который мы рассмотрим — это режим гаммирования. Как уже говорилось неоднократно, гаммирование — это наложение на данные криптографической гаммы, вырабатываемой с помощью некоторого криптографического алгоритма.

Открытый текст, разбитый на 64-битовые блоки, обозначаемые $T_O^{(i)}$, зашифровывается в режиме гаммирования путем поразрядного сложения по модулю 2 с гаммой шифра, которая вырабатывается блоками по 64 бита.

Уравнение зашифрования в режиме гаммирования записывается следующим образом.

$$T_{Ш}^{(i)} = \Gamma_{Ш}^{(i)} \oplus T_O^{(i)} = E_K\left(Y_{i-1}[+]C_2, Z_{i-1}[+]' C_1\right) \oplus T_O^{(i)}$$

Здесь $T_{Ш}^{(i)}$ обозначает 64-битовый блок шифртекста, E_K — функцию шифрования в режиме простой замены, аргументами которой являются два 32-разрядных значения, C_1 и C_2 — константы, равные соответственно 01010104h и 01010101h.

Величины Y_i и Z_i определяются итерационно по мере вычисления гаммы следующим образом:

$(Y_0, Z_0) = E_K(S)$, где S — 64-разрядное значение, называемое синхропосылкой.

Для i = 1, 2, …,

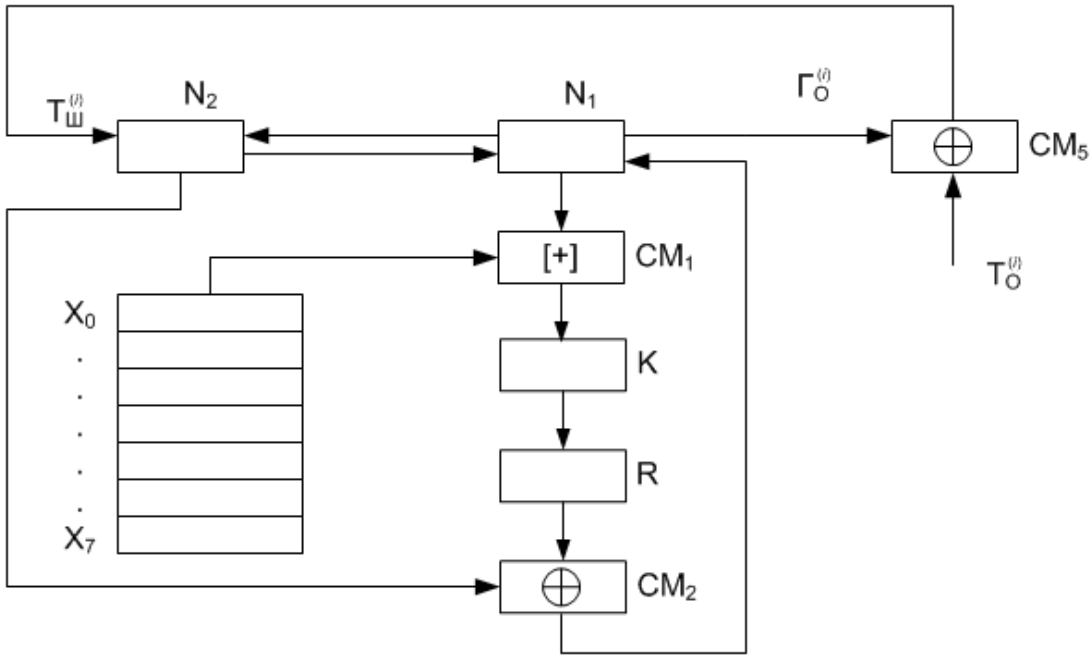
$(Y_i, Z_i) = \left(Y_{i-1}[+]C_2, Z_{i-1}[+]' C_1\right)$.

Уравнение расшифрования запишется аналогично.

$$T_O^{(i)} = \Gamma_{Ш}^{(i)} \oplus T_{Ш}^{(i)} = E_K\left(Y_{i-1}[+]C_2, Z_{i-1}[+]' C_1\right) \oplus T_{Ш}^{(i)}$$

Смысл режима гаммирования заключается в следующем. Используя простое рекуррентное соотношение, мы последовательно вычисляем для каждого блока открытого текста некоторое значение той же длины, что и блок. Зашифровывая вычисленные значения в режиме простой замены, мы получаем блоки гаммы, которые побитно складываем по модулю 2 с блоками открытого текста. Кроме того, мы должны задать начальное 64-битовое значение, синхропосылку, на основе которого будем вычислять все последующие значения.

Режим гаммирования с обратной связью очень похож на режим гаммирования и отличается от него только способом выработки гаммы. Схема шифрования в режиме гаммирования с обратной связью приведена на рисунке ниже.



Гаммирование с обратной связью

Сначала, как и в режиме гаммирования, необходимо задать 64-битовую синхропосылку, которая затем зашифровывается в режиме простой замены. Полученное значение является первым блоком гаммы, который накладывается на первый блок открытого текста путем побитного сложения по модулю 2.

Полученный в результате первый блок шифртекста зашифровывается в режиме простой замены и результат шифрования используется в качестве второго блока гаммы, то есть побитно складывается по модулю 2 со вторым блоком открытого текста.

Таким образом, каждый блок гаммы, за исключением первого, получается путем шифрования в режиме простой замены предыдущего блока шифртекста.

Запишем уравнения зашифрования в режиме гаммирования с обратной связью.

$T_{Ш}^{(1)} = Г_{Ш}^{(1)} \oplus T_O^{(1)} = E_K(S) \oplus T_O^{(1)}$.

Для i = 2, 3, …,

$T_{Ш}^{(i)} = Г_{Ш}^{(i)} \oplus T_O^{(i)} = E_K\left(T_{Ш}^{(i-1)}\right) \oplus T_O^{(i)}$.

Уравнения расшифрования в режиме гаммирования с обратной связью записываются аналогично.

$T_O^{(1)} = Г_{Ш}^{(1)} \oplus T_{Ш}^{(1)} = E_K(S) \oplus T_{Ш}^{(1)}$.

Для i = 2, 3, …,

$T_O^{(i)} = Г_{Ш}^{(i)} \oplus T_{Ш}^{(i)} = E_K\left(T_{Ш}^{(i-1)}\right) \oplus T_{Ш}^{(i)}$.

Рассмотренные нами три режима работы российского стандарта шифрования ГОСТ 28147-89 обеспечивают непосредственно шифрование данных, то есть служат для обеспечения конфиденциальности. Кроме того, в ГОСТе предусмотрен режим, предназначенный для решения задачи обеспечения целостности информации, — режим выработки имитовставки

Имитовставка — это контрольная комбинация, зависящая от открытого текста и секретного ключа, использующаяся для обнаружения всех случайных или преднамеренных изменений в открытом тексте.

Выработка имитовставки осуществляется следующим образом: открытый текст разбивается на блоки длиной 64 бита, первый блок зашифровывается 16-ю циклами режима простой замены, полученное значение побитно складывается по модулю 2 со вторым блоком открытого текста, результат зашифровывается 16 циклами режима

простой замены и побитно складывается по модулю 2 с третьим блоком открытого текста. Эти действия повторяются для всех блоков открытого текста, и в итоге мы получим 64-битовое значение. В качестве имитовставки можно взять любое количество бит этого значения, обычно берутся младшие 32 бита.

В режиме выработки имитовставки используется то же основное криптографическое преобразование, что и в режиме простой замены, но течении 16-ти циклов, а не 32-х. Длина ключа так же составляет 256 бит.

Имитовставка добавляется отправителем к защищаемым данным, которые могут передаваться и в открытом виде, если задача обеспечения конфиденциальности не ставится. Получатель, знающий секретный ключ, вычисляет имитовставку и сравнивает вычисленное значение с полученным. Если обнаруживается несовпадение, полученные данные отвергаются как ложные.

Таким образом, для потенциального злоумышленника практически неразрешимы следующие задачи: вычисление имитовставки для заданного открытого текста, и подбор открытого текста для заданного значения имитовставки. Имитовставка обеспечивает невозможность имитации и подмены данных, делает невозможными те атаки на целостность данных, о которых мы говорили на вводной лекции.

Теперь обсудим особенности каждого из рассмотренных режимов работы ГОСТа. Начнем, естественно, с режима простой замены.

Этот режим имеет следующие особенности, которые, скорее, можно назвать недостатками:

1. Так как блоки открытого текста зашифровываются последовательно и независимо друг от друга, то одинаковые блоки открытого текста преобразуются в одинаковые блоки шифртекста. В результате криптоаналитик получает определенную информацию о структуре открытого текста. Кроме того, зачастую открытый текст представляет собой документ, содержащий среди всего прочего стандартные заголовки, поля, адреса и т. п. И если криптоаналитику удастся определить таким образом значения некоторых блоков открытого текста, то в случае их повторения он может раскрыть часть зашифрованных данных. А это уже является уязвимостью.

2. В режиме простой замены на вход функции шифрования всегда подается 64-битовое значение, но открытый текст не всегда можно разделить на целое число 64-битовых блоков. Поэтому возникает вопрос: как дополнять последний блок открытого текста до длины 64 бита? Можно дополнить его нулевыми или единичными битами или некоторой битовой последовательностью, формируемой по заданному правилу. Но это дает дополнительную информацию для криптоаналитика.

Указанные недостатки режима простой замены устранены в режиме гаммирования. Для каждого блока открытого текста вырабатывается свой блок гаммы, поэтому одинаковые блоки открытого текста при шифровании преобразуются в разные блоки шифртекста. Проблема заполнения последнего блока открытого текста также решена, в этом случае можно использовать столько бит последнего блока гаммы, сколько необходимо.

При шифровании в режиме гаммирования стороны, участвующие в информационном обмене, должны согласовать значение синхропосылки, определяющей выработку гаммы. Кроме того, необходимо учесть, что вырабатываемая гамма зависит только от синхропосылки и ключа шифрования. Поэтому при шифровании на одном ключе нескольких сообщений, нужно каждый раз задавать новую синхропосылку.

Недостатком режима гаммирования является оторванность гаммы от открытого текста и шифртекста, так как они не участвуют в процессе ее формирования. Вследствие этого противник может внести изменения в шифртекст, так что при расшифровании изменится открытый текст. Так как при изменении одного бита шифртекста меняется только один бит открытого текста, то при определенных условиях эти изменения могут быть предсказуемыми и даже целенаправленными.

Для устранения этого недостатка используется режим гаммирования с обратной связью, в котором изменение одного бита шифртекста приведет к лавинному эффекту изменений в открытом тексте

Что касается выработки имитовставки, то мы уже не раз останавливались на задаче защиты целостности. Здесь можно добавить следующее: в случае одновременного шифрования и выработки имитовставки следует использовать разные ключи — ключ шифрования и ключ аутентификации.

DES

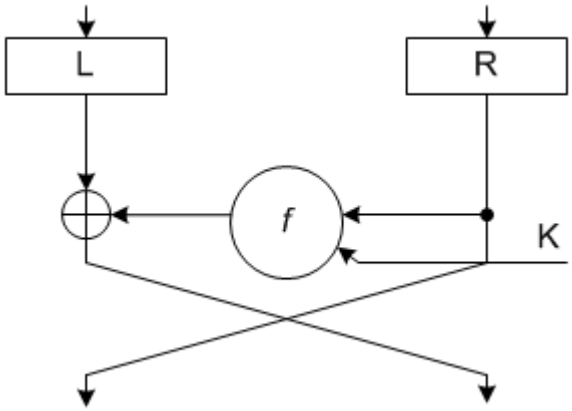
Изучение современных симметричных шифров мы начали с российского стандарта шифрования ГОСТ 28147-89. Теперь рассмотрим американские стандарты шифрования: существовавший до 1998 года Data Encryption Standard (DES) и сменивший его в 2001 году Advanced Encryption Standard (AES).

В 1972 году американское НБС (Национальное Бюро Стандартов), сейчас переименованное в НИСТ (Национальный Институт Стандартов и Технологий), выступило инициатором создания единого, стандартного криптографического алгоритма шифрования данных для частного сектора. Такой стандарт был необходим для обеспечения возможности согласованной работы криптографических средств от различных производителей. Было проведено два конкурса, победителем второго из которых стала разработка фирмы IBM, выполненная исследовательской группой под руководством Хорста Фейстеля.

В конце 1976 года DES был принят в качестве федерального стандарта и разрешен к использованию, в начале 1977 года было опубликовано официальное описание стандарта FIPS PUB 46.

DES представляет собой блочный шифр, работающий с 64-битовыми блоками. Длина ключа составляет 64 бита, но каждый восьмой бит используется для проверки четности, поэтому реальная длина ключа шифрования равна 56 битам.

Шифрование заключается в многократном повторении основного криптографического преобразования, называемого раундом или циклом. Схема одного цикла DES приведена на рисунке ниже.



Цикл DES

Блок открытого текста разбивается на две равные части по 32 бита, правая часть вместе с раундовым ключом подаются на вход нелинейной функции, значение на выходе этой функции поразрядно складывается по модулю 2 с левой частью блока. После этого исходное значение правой части блока записывается в левую часть, а результат предыдущих преобразований – в правую. Нелинейная функция осуществляет следующую последовательность действий над входным 32-битовым значением:

— перестановку с расширением до 48 бит;

— поразрядное сложение по модулю 2 с 48-битовым раундовым ключом;

— замену 8-ми 6-битовых блоков на 4-битовые блоки посредством таблиц замен, называемых S-блоками;

— перестановку с помощью P-блока.

Рассмотренный цикл повторяется 16 раз.

Такая схема шифрования, когда блок данных разбивается на две равные части, которые пошагово преобразуются с использованием некоторой нелинейной функции, использовалась при конструировании большого числа симметричных блочных шифров. Она носит название сети Фейстеля по имени ее создателя. Российский стандарт шифрования ГОСТ 28147-89 также построен на основе сети Фейстеля.

Более подробно данный шифр рассматривать не будем, поскольку действующим стандартом он уже не является.

AES

За годы, прошедшие с момента создания DES, он подвергался многочисленным исследованиям и попыткам криптоанализа. Были разработаны такие методы криптоанализа как дифференциальный и линейный, проведены успешные атаки на некоторые варианты DES с меньшим числом циклов. Но, в целом, все проведенные исследования подтвердили надежность DES. Поэтому главным его недостатком стала длина ключа. Быстрое развитие средств вычислительной техники уже в первой половине 90-х годов сделало возможным полный перебор всех ключей DES за реальное время.

В связи с этим в 1997 году НИСТ объявил конкурс на создание нового общенационального стандарта шифрования данных, который должен был придти на замену DES. На конкурс были представлены 15 алгоритмов симметричного шифрования из разных стран, из которых затем отобрали 5 финалистов. В итоге, лучшим был признан шифр Rijndael, разработанный двумя бельгийскими криптографами Винсентом Рэйменом и Йоаном Дайменом. В 2002 году Rijndael был принят в качестве улучшенного стандарта шифрования данных AES.

Одним из требований к новому стандарту была возможность работы с информационными блоками и ключами шифрования различной длины. Rijndael представляет собой шифр с переменной длиной блока и переменной длиной ключа, которые независимо могут приниматься равными 128, 192 и 256 бит. В свою очередь, AES представляет собой несколько усеченную версию Rijndael — для него длина блока фиксирована и составляет 128 бит, а длина ключа может изменяться.

AES является итерационным шифром, что означает, как уже было сказано, многократное повторение основного криптографического преобразования. При этом число раундов не фиксировано, а зависит от длины блока и длины ключа. Эта зависимость приведена в таблице ниже.

Количество раундов AES/Rijndael

Длина ключа \ Длина блока	128	192	256
128	10	12	14
192	12	12	14
256	14	14	14

На основе исходного ключа для каждого раунда шифрования с помощью специальной процедуры формируются раундовые ключи.

При шифровании блок открытого текста представляется в виде матрицы, состоящей из 4-х строк и числа столбцов, зависящего от длины блока (4-х, 6-ти и 8-ми, соответственно, для 128-, 192- и 256-битовых блоков). Элементами матрицы являются байты. Аналогично задается ключ шифрования.

Блок открытого текста A_i длиной 128 бит можно представить следующим образом.

$$A_i = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}.$$

Основное криптографическое преобразование состоит из следующих этапов, которые обычно описываются на С-подобном псевдокоде:

1) Замена байт в матрице A_i (ByteSub);
2) Циклический сдвиг строк (ShiftRow);
3) Перемешивание столбцов (MixColumn);
4) Добавление раундового ключа (AddRoundKey).

Теперь подробно рассмотрим каждый из этих этапов.

1. ByteSub

Каждый байт матрицы блока открытого текста a_{ij} представляется элементом поля Галуа F_{2^8}, построенного как расширение поля F_2 по корням неприводимого многочлена $f(x) = x^8 + x^4 + x^3 + x + 1$, и заменяется мультипликативно обратным элементом a_{ij}^{-1} в поле Галуа F_{2^8}. Байт со значением ‘00h’ отображается в себя.

После этого осуществляется аффинное преобразование $x(x_0x_1x_2x_3x_4x_5x_6x_7) \to y(y_0y_1y_2y_3y_4y_5y_6y_7)$ результата предыдущей замены по следующей формуле:

$$y = C \cdot x + C_1 (\mathrm{mod}\, x^8 + 1).$$

В матричном представлении аффинное преобразование имеет следующий вид:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}.$$

Оба рассмотренных преобразования можно объединить в одной таблице подстановки, определяющей замену для всех возможных значений байта.

Обратное преобразование для этапа замены байт состоит из обратного аффинного преобразования и взятия мультипликативно обратного элемента для каждого байта в поле F_{2^8}.

2) ShiftRow

На данном этапе строки матрицы циклически сдвигаются влево на различную величину. 0-я строка не сдвигается, а 1-я, 2-я и 3-я строки сдвигаются, соответственно, на величины C_1, C_2 и C_3. Значения этих величин зависят от размера блока и приведены в таблице ниже.

Величины сдвига

Величина сдвига / Длина блока	C_1	C_2	C_3
128	1	2	3
192	1	2	3

256	1	3	4

Обратное преобразование представляет собой циклический сдвиг строк матрицы вправо на то же число позиций.

3) MixColumn

Столбцы, всегда состоящие из 4-х байт, рассматриваются как полиномы 3-й степени вида $a(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0$ над полем F_{2^8} Перемешивание заключается в умножении полинома, соответствующего столбцу на константу $C(x) = '03'x^3 + '01'x^2 + '01'x + '02'$ по модулю $x^4 + 1$.

В матричном представлении данное преобразование имеет следующий вид:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}.$$

Обратное преобразование представляет собой умножение столбцов матрицы на полином $D(x)$ = ‘0Bh’x^3 + ‘0Dh’x^2 + ‘09h’x + ‘0Eh’, мультипликативно обратный к $C(x)$ по модулю $x^4 + 1$.

4) AddRoundKey

Последний этап основного криптографического преобразования заключается в поразрядном сложении по модулю 2 матрицы блока открытого текста и матрицы сеансового ключа. Благодаря операции XOR это преобразование обратно самому себе.

Процедура формирования сеансовых ключей AES заключается в развертывании исходного ключа, который может иметь длину 128, 192 или 256 бит. Длина сеансового ключа совпадает с длиной информационного блока.

Для формирования сеансовых ключей в AES рекомендуется использовать два разных алгоритма: первый для исходного ключа длиной 128 или 192 бита, второй для исходного ключа длиной 256 бит.

Алгоритм 1.

Исходный ключ рассматривается как одномерный массив 4-байтовых слов длиной n_k, где n_k – это число столбцов в матрице ключа, то есть преобразование в одномерный массив происходит считыванием по столбцам матричного представления ключа. Развернутый массив сеансовых ключей имеет размер $n_m(n_r + 1)$, где n_m – это число столбцов в матрице блока открытого текста, а n_r – число раундов.

Алгоритм развертывания ключа на С-подобном псевдокоде имеет следующий вид.

```
for (i=0; i<Nk; i++)
{
 W[i]=K[i];
}
for (i=Nk; i<Nm*(Nr + 1); i++)
{
 T=W[i-1];
 if (i%Nk==0)
 T=SubByte(RotByte(T) ^RC[i/Nk]);
 W[i]=W[i - Nk]^T;
}
```

K – это массив, содержащий исходный ключ, а W, соответственно, содержит развернутые сеансовые ключи. Элементами этих массивов являются 4-байтовые слова. Nk, Nm, Nr означают, соответственно, число столбцов в матрице исходного ключа, число столбцов в матрице информационного блока и число раундов основного криптографического преобразования.

SubByte – это функция, которая применяет блок замены основного криптографического преобразования AES к каждому байту входного слова, а RotByte – это циклический сдвиг слова на один байт влево.

RC – это массив раундовых констант, которые вычисляются с помощью выражения $RC[j] = x^j \left(\mathrm{mod}\, x^8 + x^4 + x^3 + x + 1\right)$.

Таким образом, первые n_k элементов массива развернутых раундовых ключей заполняются элементами исходного ключа. Каждый последующий *i*-ый элемент образуется путем поразрядного сложения по модулю 2 элемента с индексом $i - 1$ с элементом с индексом $i - n_k$. При этом для элементов с индексом, кратным n_k, элементы с индексом $i - 1$ перед операцией XOR подвергаются дополнительным преобразованиям.

Алгоритм 2.

Алгоритм развертывания ключа длиной 256 бит на С-подобном псевдокоде имеет следующий вид.

```
for (i=0; i<Nk; i++)
{
 W[i]=K[i];
}
for (i=Nk; i<Nm*(Nr + 1); i++)
{
 T=W[i-1];
 if (i%Nk==0)
 T=SubByte(RotByte(T) ^RC[i/Nk]);
 else
 if (i%Nk==4)
 T=SubByte(T);
 W[i]=W[i - Nk]^T;
}
```

Теперь с учетом всего вышесказанного запишем алгоритмы зашифрования и расшифрования AES также с помощью С-подобного псевдокода.

```
AddRoundKey(S,K[0]);
for (i=1; i<Nr; i++)
{
 ByteSub(S);
 ShiftRow(S);
 MixColumn(S);
 AddRoundKey(S,K[i]);
}
ByteSub(S);
ShiftRow(S);
AddRoundKey(S,K[Nr]);
```

Алгоритм расшифрования.

```
AddRoundKey(S,K[Nr]);
ShiftRow(S);
ByteSub(S);
for (i=Nr-1; i>0; i--)
{
 AddRoundKey(S,K[i]);
 MixColumn(S);
 ShiftRow(S);
 ByteSub(S);
}
AddRoundKey(S,K[0]);
```

RC6

Еще один симметричный блочный шифр, который мы рассмотрим, называется RC6. Этот шифр был разработан RSA Data Security специально для конкурса AES и оказался в первой пятерке финалистов.

Одной из особенностей RC6 является использование операции целочисленного умножения, что существенно увеличивает рассеивание информации при шифровании. Однако, операция умножения, медленно выполняемая на некотором оборудовании, затрудняет реализацию данного шифра на ряде аппаратных платформ.

RC6 является полностью параметризованным алгоритмом шифрования. Алгоритм с заданными параметрами обозначается как RC6-*w*/*r*/*b*, где *w* – длина машинного слова в битах, *r* – число раундов, *b* – длина ключа шифрования в байтах.

При шифровании открытый текст разбивается на блоки, состоящие из 4-х w-битовых слов, и преобразуется посредством 6-ти базовых операций.

Перечислим их:

$a + b$ – целочисленное сложение по модулю 2^w;

$a - b$ – целочисленное вычитание по модулю 2^w;

$a \oplus b$ – поразрядное сложение по модулю 2 *w*-битовых слов;

$a \times b$ – целочисленное умножение по модулю 2^w;

$a << b$ – циклический сдвиг w-битового слова влево на величину, заданную $\log_2 w$ младшими битами *b*;

$a >> b$ – циклический сдвиг w-битового слова вправо на величину, заданную $\log_2 w$ младшими битами *b*;

Рассмотрим алгоритмы зашифрования и расшифрования в общем виде.

Алгоритм зашифрования.

Исходный ключ, состоящий из *b* байт, формирует ключевую таблицу *S*[0; …; 2*r* + 3]

Блок открытого текста длиной 4*w* бит записывается в 4 *w*-битовых регистра *A*, *B*, *C*, *D*.

Содержимое указанных регистров подвергается следующим преобразованиям.

```
B=B + S[0];
D=D + S[1];
for (i=1; i<=r; i++)
{
 T=(B×(2B + 1))<<log2(w);
 U=(D×(2D + 1))<<log2(w);
 A=((A^T)<<U) + S[2i];
 C=((C^U)<<T) + S[2i+1];
 (A; B; C; D)=(B; C; D; A);
}
A=A + S[2r + 2];
C=C + S[2r + 3];
```

Алгоритм расшифрования.

Исходный ключ, состоящий из *b* байт, формирует ключевую таблицу *S*[0; …; 2*r* + 3]

Блок шифртекста длиной 4*w* бит записывается в 4 *w*-битовых регистра *A*, *B*, *C*, *D*.

Содержимое указанных регистров подвергается следующим преобразованиям.

```
C=C - S[2r + 3];
A=A - S[2r + 2];
for (i=r; i>=1; i--)
{
 (A; B; C; D)=(D; A; B; C);
 U=(D×(2D + 1))<<log2(w);
 T=(B×(2B + 1))<<log2(w);
```

```
 C=((C - S[2i+1])>>T)^U;
 A=((A – S[2i])>>U)^T
}
D=D - S[1];
B=B - S[0];
```

Режимы работы симметричных блочных шифров

Режим ECB (Electronik Code Book — электронная кодовая книга) является аналогом режима простой замены ГОСТ 28147-89.

Здесь и далее обозначим последовательность блоков открытого текста m_1, m_2, ..., m_l. Соответствующие блоки шифртекста будут получены в результате преобразования $c_i = E_k(m_i)$. Данный режим может потребовать дополнения последнего блока открытого текста.

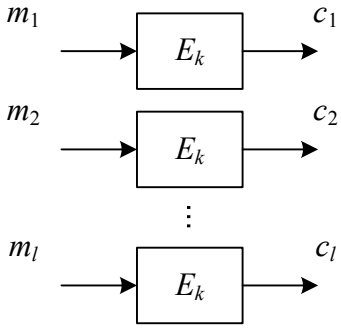


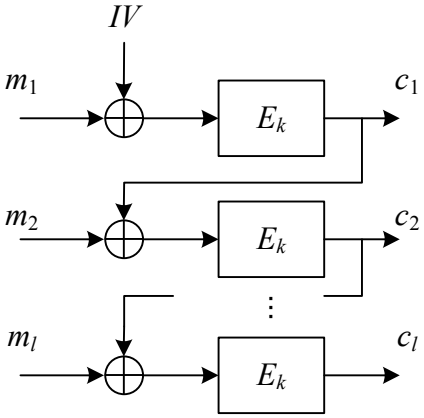
Рисунок — Зашифрование в режиме ECB

Недостатки:

1. Если $m_i = m_j$, то и $c_i = c_j$.
2. Уязвимость перед атакой на удаление блоков.
3. Уязвимость перед атакой на вставку блоков.

Режим CBC (Cipher Block Chaining — сцепление блоков шифра).

Данный режим также может потребовать дополнения последнего блока открытого текста. Зашифрование осуществляется по формулам $c_1 = E_k(m_1 \oplus IV)$, $c_i = E_k(m_i \oplus c_{i-1})$ для $i > 1$. Здесь IV — это вектор инициализации (аналог синхропосылки), который не является секретным и может передаваться в открытом виде как часть сообщения. Один из способов работы с вектором инициализации заключается в том, что он генерируется случайным образом и принимается в качестве первого блока шифртекста.



Зашифрование в режиме CBC

Режим OFB (Output Feedback — обратная связь по выходу).

Данный режим обращает блочный шифр в поточный. Для этого задается параметр режима, который обозначим j — он определяет количество битов ключевого потока, получаемых в результате одного зашифрования блочным шифром. При этом $1 \le j \le n$, где n — длина блока данных, которым оперирует блочный шифр. Зашифрование в режиме OFB реализуется по следующим формулам:

$x_1 = IV$;

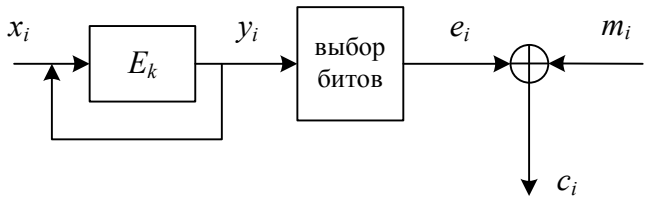
для $i \ge 1$

$y_i = E_k(x_i)$;

$e_i = j$ старших битов блока y_i;

$c_i = m_i \oplus e_i$;

$x_{i+1} = y_i$.



Зашифрование в режиме OFB

Можно увидеть, что режим обратной связи по выходу аналогичен режиму гаммирования ГОСТ 28147-89. Отличие заключается в более простом способе получения блоков ключевого потока в режиме OFB. Кроме того, в отечественном стандарте симметричного шифрования размер блока гаммы фиксирован и определяется длиной блока открытого текста.

Режим CFB (Cipher Feedback — обратная связь по шифртексту).

Данный режим также обращает блочный шифр в поточный с длиной блоков ключевого потока, равной j, и реализуется согласно следующим формулам:

$y_0 = IV$;

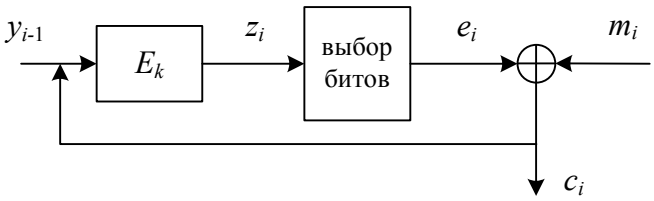
для $i \ge 1$

$z_i = E_k(y_{i-1})$;

$e_i = j$ старших битов блока z_i;

$c_i = m_i \oplus e_i$;

$y_i = c_i$.



Зашифрование в режиме CFB

Данный режим аналогичен режиму гаммирования с обратной связью ГОСТ 28147-89 с теми же отличиями, что и в предыдущем случае.

Глава 5. Криптография с открытым ключом

В 1976 году была опубликована статья Уитфилда Диффи и Мартина Хеллмана «Новые направления в криптографии», представившая концепцию криптографии с открытым ключом.

Криптосистемы с открытым ключом, также называемые асимметричными криптосистемами, используют два разных ключа: открытый ключ зашифрования и закрытый ключ расшифрования. Открытый ключ в общем случае доступен всем желающим, а закрытый ключ известен только законному владельцу. Оба ключа связаны между собой некоторой зависимостью. При этом данная зависимость такова, что, зная один ключ, вычислить другой практически невозможно.

Криптографические алгоритмы с открытым ключом используются для шифрования данных, для решения проблемы распределения секретных ключей, для выработки цифровой подписи.

С проблемой распределения ключей шифрования сталкиваются все симметричные криптосистемы. Стороны, участвующие в защищенном информационном обмене, должны каким-то образом предварительно получить общий секретный ключ, который нельзя передавать в открытом виде. Кроме того, необходимо обеспечить эффективное управление сеансовыми ключами, безопасное хранение долговременных ключей и т. д.

Все перечисленные проблемы настолько важны, что в современной криптографии управление ключами выделяют в отдельный раздел.

Существует несколько решений задачи предварительного распределения секретных ключей.

1) Физическое распределение. Ключи рассылаются с помощью курьера с той или иной степенью охраны.

2) Распределение с помощью протоколов с секретными ключами. Между пользователями и неким доверенным центром распределены долговременные ключи. Генерирование сеансовых ключей и обмен ими между пользователями происходит под управлением доверенного центра. Однако, долговременные ключи могут быть переданы только физическим путем.

3) Распределение с помощью протоколов с открытым ключом. Это наиболее важное приложение криптографии с открытым ключом, которое будет рассмотрено несколько позднее.

Базовым понятием в криптографии с открытым ключом является понятие однонаправленной или односторонней функции. Значение такой функции для заданного аргумента достаточно легко вычислить, но практически невозможно вычислить значение аргумента для заданного значения функции. Непосредственно для шифрования однонаправленные функции не используются. Для шифрования применяются однонаправленные функции с лазейкой (ловушкой). Для такой функции вычислить обратную функцию достаточно просто, если известна некоторая секретная информация, и практически невозможно, если эта информация неизвестна.

Основными задачами, приводящими к однонаправленным функциям, являются задачи разложения на множители и дискретного логарифмирования в конечном поле.

В асимметричных криптосистемах, построенных на однонаправленных функциях с лазейкой, открытый ключ определяет конкретную реализацию функции, а закрытый ключ дает информацию о лазейке.

Наиболее безопасными и практичными являются следующие криптосистемы с открытым ключом: RSA, Рабина и Эль-Гамаля.

Здесь следует сделать небольшое отступление. В некоторых источниках можно встретить упоминание о том, что асимметричные криптосистемы – это и есть современная криптография, а симметричные шифры устарели и небезопасны. Такую информацию

всерьез воспринимать нельзя, она говорит, прежде всего, о неквалифицированности авторов. Криптография с открытым ключом и традиционная криптография не противопоставляются друг другу, а наоборот совместно используются в современных системах защиты информации.

Теперь рассмотрим криптосистему RSA.

В 1978 году трое ученых, Рональд Райвест, Ади Шамир и Леонард Адлеман, опубликовали первый криптографический алгоритм с открытым ключом, названный ими (по первым буквам фамилий) RSA. Этот алгоритм основывается на сложности проблемы факторизации, то есть разложения числа на простые множители.

Рассмотрим информационный канал между двумя пользователями A и B, которых по традиции будем называть Алиса и Боб.

Алгоритм генерации ключей.

1. Алиса генерирует два больших простых числа p и q, отличных друг от друга. При этом $|p - q|$ – большое число, хотя p и q имеют приблизительно одинаковый битовый размер.

2. Держа p и q в секрете, Алиса вычисляет их произведение $n = p \cdot q$, которое называют модулем алгоритма.

3. Алиса вычисляет значение функции Эйлера для n по формуле $\varphi(n) = (p - 1)(q - 1)$.

4. Алиса выбирает целое число e, взаимно простое со значением функции $\varphi(n)$. Это число называется экспонентой шифрования. Как правило, e берут равным 3, 17 или 65537.

5. Алиса применяет расширенный алгоритм Евклида к паре чисел e и $\varphi(n)$ и вычисляет значение d, удовлетворяющее соотношению $e \cdot d \equiv 1 \pmod{\varphi(n)}$. Это значение называется экспонентой расшифрования.

6. Пара (e, n) публикуется в качестве открытого ключа Алисы, d является закрытым ключом и держится в секрете.

Алгоритм зашифрования.

1. Боб получает аутентичную копию открытого ключа Алисы – пару (e, n).

2. Боб представляет сообщение в виде числа m, меньшего модуля алгоритма. В общем случае сообщение может быть разбито на блоки, каждый из которых представляется своим числом.

3. Боб вычисляет $C = m^e \pmod{n}$.

4. Зашифрованное сообщение отправляется Алисе.

Алгоритм расшифрования.

1. Алиса получает криптограмму C от Боба.

2. Алиса вычисляет $m = C^d \pmod{n}$.

Теперь покажем, что расшифрование в RSA осуществляется корректно. Для этого сформулируем и докажем теорему.

Теорема.

Расшифрование в криптосистеме RSA корректно, то есть $C^d \pmod{n} = m$.

Доказательство.

Случай 1. Оба числа p и q не делят m, то есть НОД(m, n) = 1.

Из расширенного алгоритма Евклида $ed - 1 = s\varphi(n),\ s \in Z$.

$C^d \pmod{n} = \left(m^e \pmod{n}\right)^d \bmod n = m^{ed} \pmod{n}$.

Так как $m \in Z_n$ и НОД(m, n) = 1, то мы можем применить теорему Эйлера.

$m^{\varphi(n)} \equiv 1 \pmod{n}$.

$m^{s\varphi(n)} \equiv 1^s \pmod{n} \equiv 1 \pmod{n}$.

$m^{ed-1} \equiv 1 \pmod{n}$.

$m^{ed} \equiv m \pmod{n}$.

$C^d \equiv m(\mathrm{mod}\, n)$.

Случай 2. В точности одно простое число (скажем p) делит m.

В этом случае $p|m$ и НОД(q, m) = 1. Применим теорему Эйлера.

$m^{\varphi(q)} \equiv 1(\mathrm{mod}\, q)$.

$\left(m^{\varphi(q)}\right)^{\varphi(p)} \equiv 1^{\varphi(p)}(\mathrm{mod}\, q) \equiv 1(\mathrm{mod}\, q)$.

$m^{\varphi(n)} \equiv 1(\mathrm{mod}\, q)$.

$m^{s\varphi(n)} \equiv 1^s(\mathrm{mod}\, q) \equiv 1(\mathrm{mod}\, q)$.

$m^{ed-1} \equiv 1(\mathrm{mod}\, q)$.

$m^{ed} \equiv m(\mathrm{mod}\, q)$.

$C^d \equiv m(\mathrm{mod}\, q)$.

Так как $p \mid m \Rightarrow p \mid m^{ed} \Rightarrow \left(m^{ed} - m\right)$.

$m^{ed} \equiv m(\mathrm{mod}\, p)$.

$C^d \equiv m(\mathrm{mod}\, p)$

Запишем систему

$$\begin{cases} C^d \equiv m(\mathrm{mod}\, q) \\ C^d \equiv m(\mathrm{mod}\, p) \end{cases}.$$

Согласно китайской теореме об остатках $C^d \equiv m(\mathrm{mod}\, pq) \Rightarrow C^d \equiv m(\mathrm{mod}\, n)$.

Случай 3. Оба числа p и q делят m.

$p|m$ и $q|m$, тогда и $n = pq \mid m, \Rightarrow$

$n \mid m^{ed} \Rightarrow n \mid \left(m^{ed} - m\right) \Rightarrow m^{ed} \equiv m(\mathrm{mod}\, n) \Rightarrow C^d \equiv m(\mathrm{mod}\, n)$.

Теорема доказана.

Рассмотрим простой пример шифрования с помощью криптосистемы RSA.

Пример.

В качестве шифртекста возьмем слово RSA. Каждый символ заменим соответствующим элементом кольца классов вычетов по модулю 26. Получим последовательность 17 18 0. Представим полученные числа в двоичном виде, при этом, так как латинский алфавит содержит 26 знаков, каждый элемент алфавита можно закодировать 5-ю битами. Таким образом, мы получим битовую строку 100011001000000.

Приступим к генерации ключей.

1. Возьмем следующие простые числа p = 41, q = 29.
2. Вычислим $n = p \cdot q = 41 \cdot 29 = 1189$.
3. Вычислим значение функции Эйлера $\varphi(n) = (41 - 1)(29 - 1) = 1120$.
4. Возьмем в качестве экспоненты шифрования e = 3.
5. Вычислим с помощью расширенного алгоритма Евклида экспоненту расшифрования.

Пример вычислений

q	r	x	y	a	b	x2	x1	y2	y1
-	-	-	-	1120	3	1	0	0	1
373	1	1	-373	3	1	0	1	1	-373
3	0	-3	1120	**1**	0	**1**	-3	**-373**	1120

d = -373 = 747 (mod 1120).

Таким образом, открытым ключом будет пара (3, 1189), а закрытым – число 747.

Переведем 1189 в двоичный вид и получим битовую строку 10010100101. Видно, что заданный открытый текст представляет собой двоичное число, большее модуля алгоритма. Значит, открытый текст нужно разбить на блоки. Так как длина двоичного

представления модуля равна 11 бит, то длину блока выберем 10 бит. Получим два блока $m_1 = 10001_2 = 17$ и $m_2 = 1001000000_2 = 576$.

Зашифруем по отдельности каждый блок с помощью полученного открытого ключа.

$$C_1 = m_1^e \pmod n = 17^3 \pmod{1189} = 157;$$

$$C_2 = m_2^e \pmod n = 576^3 \pmod{1189} = 951.$$

Теперь расшифруем блоки криптограммы с помощью закрытого ключа.

$$m_1 = C_1^d \pmod n = 157^{747} \pmod{1189} = 157^{2\cdot 373+1} \pmod{1189} = 157 \cdot \left(157^2\right)^{373} \pmod{1189} =$$

$$= 157 \cdot 869^{373} \pmod{1189} = 157 \cdot 869^{2\cdot 186+1} \pmod{1189} = 157 \cdot 869 \cdot \left(869^2\right)^{186} \pmod{1189} =$$

$$= 887 \cdot 146^{186} \pmod{1189} = 887 \cdot \left(146^2\right)^{93} \pmod{1189} = 887 \cdot 1103^{93} \pmod{1189} =$$

$$= 887 \cdot 1103^{2\cdot 46+1} \pmod{1189} = 887 \cdot 1103 \cdot \left(1103^2\right)^{46} \pmod{1189} = 1003 \cdot 262^{46} \pmod{1189} =$$

$$= 1003 \cdot \left(262^2\right)^{23} \pmod{1189} = 1003 \cdot 871^{23} \pmod{1189} = 1003 \cdot 871^{2\cdot 11+1} \pmod{1189} =$$

$$= 1003 \cdot 871 \cdot \left(871^2\right)^{11} \pmod{1189} = 887 \cdot 59^{11} \pmod{1189} = 887 \cdot 59^{2\cdot 5+1} \pmod{1189} =$$

$$= 887 \cdot 59 \cdot \left(59^2\right)^5 \pmod{1189} = 17 \cdot 1103^5 \pmod{1189} = 17 \cdot 1103^{2\cdot 2+1} \pmod{1189} =$$

$$= 17 \cdot 1103 \cdot \left(1103^2\right)^2 \pmod{1189} = 916 \cdot 262^2 \pmod{1189} = 916 \cdot 871 \pmod{1189} = 17 \pmod{1189};$$

$$m_2 = C_2^d \pmod n = 951^{747} \pmod{1189} = 576 \pmod{1189}.$$

Алгоритмы работы с большими числами

Отличительной особенностью криптосистем с открытым ключом является использование больших чисел, длиной в сотни и даже тысячи бит. Существуют специальные алгоритмы, реализующие арифметические операции над такими числами.

Рассматривая криптосистему RSA, мы показали, что криптографическое преобразование, как прямое, так и обратное, заключается в возведении в степень по модулю. Но, вычисляя $a^b \pmod n$, нельзя сначала подсчитать a^b, а затем привести полученное значение по модулю n, если речь идет о больших числах, поскольку это вычислительно невозможно. Необходимо приводить по модулю n все промежуточные результаты.

Этот прием мы использовали, когда рассматривали пример шифрования в криптосистеме RSA. Формализуем описанный подход и запишем алгоритм.

Алгоритм экспоненциальной модуляции.

Вход: $a, k \in Z_n$, $k = \sum_{i=0} k_i \cdot 2^i$.

Выход: $a^k \bmod n$.

1. $b \leftarrow 1$. Если $k = 0$, то возврат b.
2. $A \leftarrow a$.
3. Если $k_0 = 1$, то $b \leftarrow a$.
4. Для $i = \overline{1,t}$ выполняем следующее:
   4.1. $A \leftarrow A^2 \bmod n$.
   4.2. Если $k_i = 1$, то $b \leftarrow (A \cdot b) \bmod n$.
5. Возврат b.

Теперь рассмотрим пример: с помощью алгоритма экспоненциальной модуляции вычислим $5^{596} \bmod 1234$.

Сначала представим 596 в двоичном виде: $596_2 = 1001010100_2$. Двоичные коэффициенты будем просматривать, начиная с младшего разряда.

Всю последовательность вычислений представим в виде таблицы.

i	0	1	2	3	4	5	6	7	8	9
k_i	0	0	1	0	1	0	1	0	0	1
A	5	25	625	681	1011	369	421	779	947	925
b	1	1	625	625	67	67	1059	1059	1059	1013

Таким образом, $5^{596} \bmod 1234 = 1013$.

Рассмотренный пример показывает, что время работы алгоритма экспоненциальной модуляции зависит от количества единичных бит в двоичном представлении показателя степени. На каждом шаге работы алгоритма мы осуществляем две операции: возведение в квадрат и умножение. При этом возведение в квадрат осуществляется независимо от значения соответствующего двоичного коэффициента, а умножение – только когда двоичный коэффициент равен единице.

Этой особенностью алгоритма объясняется выбор в качестве экспоненты зашифрования в криптосистеме RSA чисел 3, 17, 65537, так как они содержат всего два единичных бита.

Умножение и деление больших целых чисел также осуществляется с помощью специальных методов, более быстрых по сравнению со стандартными «наивными» методами.

Еще одним важным аспектом криптографии с открытым ключом является использование простых чисел, в частности, в криптосистеме RSA элементом открытого ключа является произведение двух больших простых чисел $N = p \cdot q$.

Каким образом определить, является ли заданное число простым? Наивный способ проверки числа p на простоту заключается в пробном делении, то есть попытке разделить p на все целые числа от 2 до $\sqrt{p}$. Очевидно, что этот метод работает безошибочно – мы либо найдем делитель p, либо докажем, что p является простым – но приводит к огромным временным затратам.

Наиболее развитые вероятностные алгоритмы проверки чисел на простоту основаны на малой теореме Ферма.

Малая теорема Ферма.

Пусть p – простое число, $a \neq 0$ и $a \in Z_p$. Тогда $a^{p-1} \equiv 1 (\bmod p)$.

Соотношение, приведенное в теореме, используется в тесте, проверяющем, является ли заданное число составным. Этот тест называют тестом Ферма.

Тест Ферма.

Вход: нечетное число n.

Выход: ответ на вопрос “является ли n простым”.

1. Для $i = \overline{1,t}$ выполняем следующее:

   1.1. Выбираем случайные числа из интервала [2, ..., n - 1].

   1.2. Вычисляем $r = a^{n-1} \bmod n$ по алгоритму экспоненциальной модуляции.

   1.3. Если $r \neq 1$, то возврат “n – составное”.

Тест Ферма по основанию a определяет простоту n с вероятностью $\frac{1}{2}$, после t итераций вероятность ошибки составляет $\frac{1}{2^t}$.

Криптосистема Эль-Гамаля

Теперь рассмотрим следующую криптосистему с открытым ключом – криптосистему Эль-Гамаля, названную так по имени ее создателя. Эта система основана на дискретном логарифмировании в конечном поле или группе точек эллиптической кривой.

Пусть $(G;\ \cdot)$ – конечная абелева группа. Задача нахождения дискретного логарифма заключается в определении целого числа *x*, которое при заданных $A, B \in G$ удовлетворяет соотношению $A^x = B$. Можно записать $x = \log_A B$.

В отличие от RSA, в криптосистеме Эль-Гамаля существуют некоторые открытые параметры, которые могут использоваться определенными группами пользователей. Они называются параметрами домена и выглядят следующим образом:

– p – большое простое число, насчитывающее около 1024 бит, такое, что $p-1$ делится на другое простое число q меньшей битовой длины.

– g – элемент мультипликативной группы поля F_p^*, степени которого по модулю p порождают большое число элементов F_p^*.

Все параметры домена выбираются таким образом, чтобы элемент $g^{\frac{p-1}{q}} \pmod p$ был образующим абелевой группы порядка q.

Алгоритм генерации ключей.

1. А генерирует большое простое число p, такое, что $p-1$ делится на другое большое простое число q, и выбирает элемент $g \in F_p^*$, такой, что $g^{\frac{p-1}{q}} \pmod p$ является образующим абелевой группы порядка q.

2. А выбирает случайное число x в интервале $1 < x < p-1$.

3. А вычисляет $h = g^x \bmod p$

4. Открытый ключ А есть h.

5. Закрытый ключ А есть x.

Алгоритм зашифрования.

1. В получает аутентичную копию открытого ключа А – число h.

2. В представляет сообщение в виде числа m в интервале $1 < m < p-1$. В общем случае сообщение может быть разбито на блоки, каждый из которых представляется своим числом.

3. В вычисляет сеансовый ключ k в интервале $1 < k < p-1$.

4. В вычисляет $C_1 = g^k \bmod p$ и $C_2 = m \cdot h^k \bmod p$.

5. В отправляет пару (C_1, C_2) А.

Алгоритм расшифрования.

1. А получает шифртекст – пару (C_1, C_2).

2. А, используя свой секретный ключ, вычисляет $\frac{C_2}{C_1^x} = \frac{m \cdot h^k}{\left(g^k\right)^x} = \frac{m \cdot \left(g^x\right)^k}{\left(g^x\right)^k} = m$.

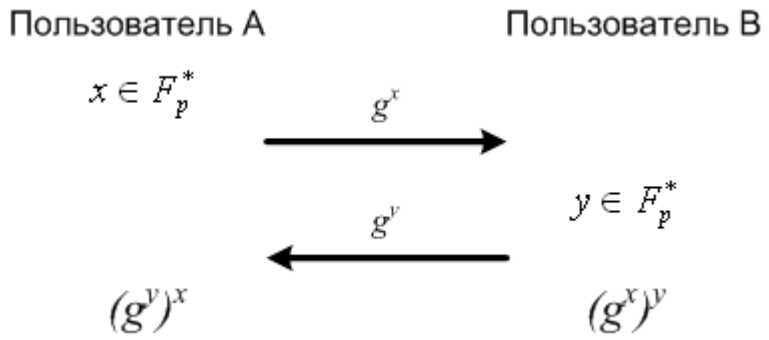
Протокол Диффи-Хеллмана

Мы уже говорили о том, что концепция криптографии с открытым ключом была разработана Диффи и Хеллманом и представлена ими в статье «Новые направления в криптографии», опубликованной в 1976 году. Эта статья содержала саму постановку задачи, но ее решение было только частичным. Диффи и Хеллман не смогли разработать асимметричную криптосистему, но смогли решить проблему распределения ключей. Их протокол распределения ключей, названный протоколом обмена ключами Диффи-Хеллмана, позволяет двум сторонам информационного обмена сформировать общий секретный ключ, используя открытый канал связи, и не прибегая к личной встрече. Стойкость этого протокола основывается на проблеме дискретного логарифмирования в конечной абелевой группе.

Рассмотрим протокол обмена ключами Диффи-Хеллмана для двух пользователей, которых традиционно обозначим А и В.

Открытыми параметрами протокола являются большое простое число p и образующий группы F_p^* g. Эти параметры постоянны и известны всем – как законным пользователям, так и возможному злоумышленнику.

Схема формирования пользователями общего секретного ключа приведена на рисунке.



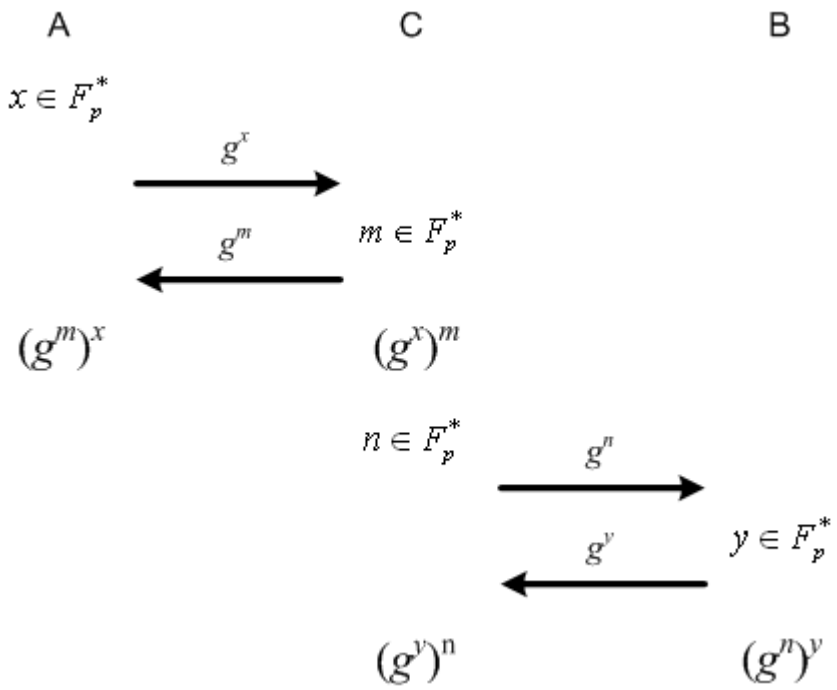
Протокол Диффи-Хеллмана

Пользователь А генерирует большое число *x*, вычисляет $g^x \pmod p$ и отправляет полученное значение пользователю В. Пользователь В генерирует большое число *y*, вычисляет $g^y \pmod p$ и отправляет полученное значение пользователю А. Затем пользователь А вычисляет $\left(g^y\right)^x \pmod p$, а пользователь В вычисляет $\left(g^x\right)^y \pmod p$. Так как $\left(g^y\right)^x \pmod p = \left(g^x\right)^y \pmod p = g^{xy} \pmod p$, то пользователи А и В получают общий секретный ключ $k = g^{xy} \pmod p$.

Если пользоваться традиционной терминологией криптографии с открытым ключом, то значения x и y, которые необходимо держать в секрете, можно назвать закрытыми ключами пользователей А и В, а значения $g^x \pmod p$ и $g^y \pmod p$, соответственно, открытыми ключами.

Какие действия в данной ситуации может осуществить злоумышленник? Он может перехватить значения $g^x \pmod p$ и $g^y \pmod p$, которые пользователи А и В посылают друг другу. Но вычислить на основе перехваченных значений секретный ключ $g^{xy} \pmod p$ злоумышленник сможет, только определив *x* по известному $g^x \pmod p$ и возведя $g^y \pmod p$ в полученную степень. Таким образом, мы приходим к задаче дискретного логарифмирования в конечном поле. Если значения *p* и *g* выбраны должным образом, такие вычисления практически невозможны.

Однако протокол обмена ключами Диффи-Хеллмана не защищен от атаки называемой «человек посередине», называемой также атакой посредника. В чем она заключается? Дело в том, что у законных участников информационного обмена не может быть твердой уверенности, что они общаются именно друг с другом. Возможна ситуация, изображенная на рисунке ниже.



Атака «человек посередине»

Как видно из рисунка, пользователь A договаривается об общем секретном ключе со злоумышленником C, думая, что договаривается с пользователем B. Пользователь B также договаривается об общем секретном ключе со злоумышленником, думая, что договаривается с пользователем A. В итоге вся переписка проходит через злоумышленника C, действия которого, если он не вносит изменений в сообщения, не могут быть обнаружены.

Один из способов пресечь атаку посредника на протокол обмена ключами Диффи-Хеллмана – это использование цифровой подписи. В этом случае стороны информационного обмена будут точно знать, с кем ведут переписку.

Криптосистема Рабина

Теперь мы рассмотрим еще одну криптосистему с открытым ключом – криптосистему Рабина, названную, как и все предыдущие, по имени ее создателя. Эта криптосистема основывается на трудности задачи извлечения квадратного корня по модулю составного числа, которая, в свою очередь, сводится к уже известной нам задаче факторизации целых чисел.

Алгоритм генерации ключей.

1. A генерирует два больших простых числа p и q таких, что $p = q = 3 (\mathrm{mod}\, 4)$. Такой специальный вид простых чисел ускоряет процедуру извлечения квадратных корней по модулю p и q.

2. A вычисляет $n = p \cdot q$.

3. Открытый ключ A есть n.

4. Закрытый ключ A есть пара (p, q).

Алгоритм зашифрования.

1. B получает аутентичную копию ключа A, то есть число n.

2. B представляет сообщение в виде числа m, меньшего n. В общем случае сообщение может быть разбито на блоки, каждый из которых представляется своим числом.

3. B вычисляет $C = m^2 (\mathrm{mod}\, n)$.

4. Зашифрованное сообщение B отправляет A.

Алгоритм расшифрования.

1. A получает шифртекст C.

2. A извлекает из C 4 квадратных корня по модулю *n*.

3. A определяет нужное значение *m* из 4-х корней.

Почему A может извлечь квадратные корни из C по модулю n, в то время, как злоумышленник этого сделать не может? Потому что задача извлечения квадратного корня по модулю простого числа решается достаточно легко, а задача извлечения квадратного корня по модулю составного числа может быть легко решена только при известной факторизации составного модуля. В этом случае необходимо извлечь квадратные корни по модулям простых множителей составного числа и воспользоваться китайской теоремой об остатках.

Еще один вопрос, который может возникнуть относительно криптосистемы Рабина: каким образом определить, какое из 4-х значений квадратного корня является значением открытого текста? Очевидно, что перед зашифрованием в открытый текст (блоки открытого текста) необходимо добавить некоторую избыточную информацию, заголовок или хэш-значение.

Теперь рассмотрим задачу извлечения квадратного корня по модулю более подробно.

Алгоритм извлечения квадратного корня по составному модулю при известной факторизации.

Вход: *n* – составное число; *p* и *q* – простые множители *n*; *a* – квадратичный вычет по модулю *n*.

Выход: 4 квадратных корня из *a* по модулю *n*.

1. Находим два квадратных корня *r* и –*r* из a по модулю *p*.

1.1. Вычисляем символ Лежандра $\left(\frac{a}{p}\right)$. Если $\left(\frac{a}{p}\right) = -1$, то квадратных корней нет.

1.2. Выбираем целое *b* такое, что $\left(\frac{b}{p}\right) = -1$.

1.3. Представляем $p - 1 = 2^s \cdot t$, где *t* – нечетное.

1.4. Вычисляем $a^{-1} \pmod p$ по расширенному алгоритму Евклида.

1.5. Вычисляем $C_0 = b^t \pmod p$, $r = a^{\frac{t+1}{2}} \pmod p$.

1.6. Для $i = \overline{1, s-1}$^

1.6a) Вычисляем $d_i = \left(r^2 \cdot a^{-1}\right)^{2^{s-i-1}} \bmod p$.

1.6b) Если $d_i \equiv -1 \pmod p$, то $r \leftarrow r \cdot C_0 \pmod p$.

1.6c) $C_0 \leftarrow C_0^2 \pmod p$.

1.7. Возврат (*r*; -*r*).

2. Аналогично находим два квадратных корня *s* и –*s* из *a* по модулю *p*.

3. Используя расширенный алгоритм Евклида, находим $c_1 p + d_1 q = 1$.

4. Вычисляем $x = (rd_1 q + sc_1 p) \bmod n$ и $y = (rd_1 q - sc_1 p) \bmod n$.

5. Возврат (±*x*; ±*y*).

Глава 6. Хеширование

Важным понятием в криптографии является понятие хэш-функции или функции хеширования. Запишем следующие определения.

Хеширование – это преобразование входной битовой строки произвольной длины в выходную битовую строку фиксированной длины. Соответственно, хэш-функция – это функция, принимающая на вход строку битов произвольной длины и возвращающая на выходе строку битов заданной длины. Значение хэш-функции называют хэш-значением, хэш-кодом, сверткой, а также дайджестом сообщения.

Вообще говоря, хеширование не является исключительно криптографическим понятием. Его смысл заключается в получении характеристического признака входных данных, по которому эти данные можно впоследствии идентифицировать. Таким образом можно осуществлять проверку на наличие ошибок в передаваемых или хранимых данных – вместе с самими данными передавать или хранить на носителе информации их хэш-значение, которое в этом случае обычно называется контрольной суммой. Кроме того, хеширование позволяет ускорить поиск в информационных массивах. Допустим, в базе данных нужно найти некоторую символьную строку. Поиск будет осуществляться путем посимвольного сравнения искомой строки с записями в базе данных. Вместе с записями в базе можно хранить их хэш-значения, и сравнивать с ними хэш-значение искомой строки, что значительно облегчит процедуру поиска.

Однако хэш-функции, использующиеся в криптографии, должны удовлетворять особым требованиям. Если в некриптографических приложениях функций хеширования основными требованиями к ним являются легкость реализации, как программной, так и аппаратной, и скорость работы, то в криптографии хэш-функции должны быть однонаправленными и защищенными от коллизий (хотя требований по простоте реализации и скорости преобразований также никто не отменял).

С понятием однонаправленной функции мы уже встречались, когда говорили о концепции криптографии с открытым ключом. Однонаправленная хэш-функция защищена в вычислительном отношении от восстановления прообразов. Это означает, что по элементу Y из множества значений хэш-функции невозможно подобрать такое значение x из множества определения, при котором $h(x) = Y$.

Коллизией (а также повторением или столкновением) по отношению к функциям хеширования называют ситуацию, когда для двух различных значений m_1 и m_2 $h(m_1) = h(m_2)$. Очевидно, что любая хэш-функция обладает бесконечным числом коллизий, поскольку представляет собой отображение бесконечного множества битовых строк произвольной длины в конечное множество битовых строк заданной длины. Защищенность от коллизий заключается в том, что хотя коллизии и существуют, их вычислительно невозможно обнаружить.

Основной принцип проектирования хэш-функции заключается в том, что ее действие должно сопровождаться лавинным эффектом – это означает, что небольшое изменение входных данных приводит к значительному изменению выходных данных.

В целом криптографические хэш-функции должны обладать следующими тремя свойствами:

1. *Защищенностью от восстановления прообразов*: должно быть невозможно в вычислительном отношении найти сообщение с данным значением хэш-функции.

2. *Защищенностью от коллизий*: вычислительно невозможно найти два разных сообщения с одним и тем же значением хэш-функции.

3. *Защищенностью от вторых прообразов*: по данному сообщению невозможно найти другое сообщение с тем же значением хэш-функции.

Еще один важный вопрос, относящийся к проектированию хэш-функций, это выбор длины выходного хэш-значения. Чем меньше длина выходного значения, тем меньше множество значений хэш-функции, и тем проще злоумышленнику обнаружить коллизию. Для хэш-функций, однако, не существует универсальной «лобовой» атаки, такой, как полный перебор всего ключевого пространства для криптосистем. Одной из стандартных атак на функцию хеширования является атака, основанная на парадоксе о днях рождения.

Этот парадокс заключается в том, что если рассматривать группу из 23 человек и больше, то вероятность того, что в этой группе найдутся два человека, родившихся в один день года, превышает 50 %. Это кажется парадоксальным, так как год может содержать 366 дней. Однако это утверждение является верным в соответствии с теорией вероятности и не содержит логических противоречий, поэтому парадоксом в научном смысле, по сути, не является. Парадокс заключается в различии между интуитивным человеческим восприятием ситуации и математическим расчетом.

Атака, основанная на рассмотренном парадоксе, направлена на обнаружение коллизии. В чем она заключается? Пусть есть некоторая хэш-функция, которая может принимать N значений. Если вычислять хэш-значения для сообщений из некоторого потока с помощью данной функции, то появления первой коллизии следует ожидать через $\sqrt{N}$ вычислений. Таким образом, если длина хэш-значения составляет 64 бита, нам потребуется всего 2^{32} вычислений (или чуть больше), чтобы обнаружить коллизию.

Длина выходного значения существующих хэш-функций составляет от 128 до 512 бит. Однако, с учетом того, что нижний предел безопасности в современном мире составляет 80 бит, минимальная длина хэш-значения должна составлять 160 бит, что обеспечит защиту от атак, направленных на обнаружение коллизий.

ГОСТ Р 34.11-94

Изучение существующих криптографических хэш-функций мы начнем с российского стандарта ГОСТ Р 34.11-94, который описан в нормативном документе «ГОСТ Р 34.11-94. Информационная технология. Криптографическая защита информации. Функция хеширования». Данный стандарт был принят и введен в действие 23 мая 1994 года и действует по настоящий день.

При рассмотрении алгоритма хеширования будем использовать обозначения, которые указаны в стандарте.

$B = \{0,1\}$ – двоичный алфавит.

B^* – множество всех конечных слов в двоичном алфавите; чтение слов и нумерация символов в них осуществляется справа налево, начиная с единицы.

$|A|$ – длина слова A.

$V_k(2)$ – множество всех двоичных слов длины k.

$A \| B$ – операция конкатенации (слияния) двух слов A и B.

$A \oplus B$ – операция поразрядного сложения по модулю 2 двух слов A и B.

$A[+]B$ – операция сложения по модулю 2^{256} двух 256-битовых слов A и B.

M – последовательность двоичных символов, подлежащая хешированию.

h – хэш-функция, отображающая последовательность $M \in B^*$ в слово $h(M) \in V_{256}(2)$.

$E_k(A)$ – результат зашифрования слова $A \in V_{64}(2)$ на ключе $K \in V_{256}(2)$ по алгоритму шифрования ГОСТ 28147-89 в режиме простой замены.

H – стартовый вектор хеширования.

Под функцией *h* понимается отображение $h: B^* \to V_{256}(2)$. Таким образом, длина выходного значения рассматриваемой хэш-функции составляет 256 бит. Это делает несостоятельной атаку, направленную на поиск коллизий.

В основу ГОСТ Р 34.11-94, а также и в основу других функций хеширования, положен тот же принцип, на котором построены современные симметричные блочные шифры – многократное повторение основного криптографического преобразования.

Поэтому для определения хэш-функции нам необходимы:

1) алгоритм вычисления шаговой функции хеширования κ, где $\kappa: V_{256}(2) \times V_{256}(2) \to V_{256}(2)$;

2) описание итеративной процедуры вычисления значения хэш-функции *h*.

В свою очередь, алгоритм вычисления шаговой функции хеширования состоит из трех этапов:

1) генерации 4-х 256-битовых ключей;

2) шифрующего преобразования – зашифрования 4-х 64-битовых подслов слова *H* на 4-х сгенерированных ключах по алгоритму шифрования ГОСТ 28147-89 в режиме простой замены;

3) перемешивающего преобразования результата шифрования.

Генерация ключей.

Рассмотрим 256-битовое слово $X = (b_{256}, b_{255}, ..., b_1) \in V_{256}(2)$.

Пусть $X = x_4 \| x_3 \| x_2 \| x_1 = \eta_{16} \| \eta_{15} \| ... \| \eta_1 = \xi_{32} \| \xi_{31} \| ... \| \xi_1$, где $x_i \in V_{64}(2)$, $i = \overline{1,4}$, $\eta_j \in V_{16}(2)$, $j = \overline{1,16}$, $\xi_k \in V_8(2)$, $i = \overline{1,32}$.

То есть мы представляем 256-битовое слово *X* как 4 64-битовых слова, 16 16-битовых слов или 32 8-битовых слова.

Обозначим $A(X) = (x_1 \oplus x_2) \| x_4 \| x_3 \| x_2$.

Зададим преобразование $P: V_{256}(2) \to V_{256}(2)$ слова $\xi_{32} \| \xi_{31} \| ... \| \xi_1$ в слово $\xi_{\varphi(32)} \| \xi_{\varphi(31)} \| ... \| \xi_{\varphi(1)}$, где $\varphi(i+1+4(k-1)) = 8i+k$, $i = \overline{0,3}$, $k = \overline{1,8}$.

Для генерации ключей необходимо использовать следующие исходные данные:

– слова $H, M \in V_{256}(2)$

– слова C_2, C_3, C_4, являющиеся константами и имеющие значения $C_2 = C_4 = 0^{256}$, $C_3 = 1^8 0^8 1^{16} 0^{24} 1^{16} 0^8 (0^8 1^8)^2 1^8 0^8 (0^8 1^8)^4 (1^8 0^8)^4$, где 0^i означает слово длины *i*, заполненное нулями, а 1^i – слово длины *i*, заполненное единицами.

При вычислении ключей реализуется следующий алгоритм:

1. Присвоить значения
   $i \leftarrow 1$, $U \leftarrow H$, $V \leftarrow M$.
2. Выполнить вычисления
   $W \leftarrow U \oplus V$, $K_1 \leftarrow P(W)$.
3. Присвоить значение
   $i \leftarrow i+1$.
4. Проверить условие $i = 5$
   При положительном исходе перейти к шагу 7.
   При отрицательном исходе перейти к шагу 5.
5. Выполнить вычисление
   $U \leftarrow A(U) \oplus C_i$, $V \leftarrow A(A(V))$, $W \leftarrow U \oplus V$, $K_i \leftarrow P(W)$.
6. Перейти к шагу 3.
7. Конец работы алгоритма.

Шифрующее преобразование.

Данное преобразование требует следующих исходных данных: 256-битовое слово $H = h_4 \| h_3 \| h_2 \| h_1$, где $h_i \in V_{64}(2)$, $i = \overline{1,4}$ и набор ключей K_1, K_2, K_3, K_4.

Каждое 64-битовое слово h_i зашифровывается на ключе K_i, $i = \overline{1,4}$, по алгоритму ГОСТ 28147-89 в режиме простой замены. Результатом шифрования будут слова $s_i = E_{K_i}(h_i)$, $i = \overline{1,4}$.

В итоге образуется 256-битовое слово $S = s_4 \| s_3 \| s_2 \| s_1$

Перемешивающее преобразование.

На данном этапе осуществляется перемешивание полученного 256-битового слова с применением регистра сдвига.

Исходными данными для этого этапа являются слова $H, M, S \in V_{256}(2)$.

Зададим отображение $\varphi : V_{256}(2) \to V_{256}(2)$ слова $\eta_{16} \| \eta_{15} \| ... \| \eta_1$ в слово $\eta_1 \oplus \eta_2 \oplus \eta_3 \oplus \eta_4 \oplus \eta_{13} \oplus \eta_{16} \| \eta_{16} \| ... \| \eta_2$, где $\eta_i \in V_{16}(2)$, $i = \overline{1,16}$.

Тогда в качестве значения шаговой функции хеширования примем слово $\kappa(M, H) = \varphi^{61}\left(H \oplus \varphi\left(M \oplus \varphi^{12}(S)\right)\right)$, где φ^i – i-я степень преобразования φ.

Основное криптографическое преобразование ГОСТ Р 34.11-94 определено, теперь необходимо рассмотреть саму процедуру вычисления хэш-функции.

Исходными данными для вычисления значения функции h является подлежащая хешированию последовательность $M \in B^*$. Параметром является стартовый вектор хеширования H – произвольное слово из $V_{256}(H)$.

Процедура вычисления функции h на каждой итерации использует следующие величины:

$M \in B^*$ – часть исходной последовательности M, еще не прошедшая обработку.

$H \in V_{256}(2)$ – текущее значение хэш-функции.

$\Sigma \in V_{256}(2)$ – текущее значение контрольной суммы.

$L \in V_{256}(2)$ – текущее значение длины уже обработанной части исходной последовательности M.

Алгоритм вычисления функции h состоит из следующих этапов:

1. Присвоить начальные значения

$M \leftarrow M$,

$H \leftarrow H$,

$\Sigma \to 0^{256}$,

$L \leftarrow 0^{256}$.

2. Проверить условие $|M| > 256$.

При положительном исходе перейти к этапу 3.

При отрицательном исходе выполнить вычисления

$L \leftarrow (L + |M|) \bmod 2^{256}$,

$M' \leftarrow 0^{256-|M|} \| M$,

$\Sigma \leftarrow \Sigma[+]M'$,

$H \leftarrow \kappa(M', H)$,

$H \leftarrow \kappa(L, H)$,

$H \leftarrow \kappa(\Sigma, H)$.

Конец работы алгоритма.

3. Вычислить подслово $M_S \in V_{256}(2)$ слова M $(M = M_P \| M_S)$.

$H \leftarrow \kappa(M_S, H)$,

$L \leftarrow (L + 256) \bmod 2^{256}$,

$\Sigma \leftarrow \Sigma[+]M_S$,

$M = M_P$.

Перейти к этапу 2.

Значение *H*, полученное на 2-м этапе содержит значение хэш-функции *h*(*M*).

MD5

В отличие от единственного отечественного стандарта хеширования ГОСТ Р 34.11-94, рассмотренного на предыдущей лекции, за рубежом и, прежде всего, в США, широкое применение находят несколько хэш-функций. Наиболее популярными являются MD5, SHA-1, а также SHA-2.

MD5 является представителем целого семейства хэш-функций, разработанных Роном Райвестом. Первой их них была функция хеширования MD4 (Message Digest 4), разработанная и опубликованная в 1990 году, улучшенной версией которой является уже упомянутая хэш-функция MD5. Кроме того, данное семейство включает в себя созданный в 1992 году алгоритм хеширования MD2 и недавнюю разработку – алгоритм MD6, созданный, в свою очередь, в 2008 году.

SHA-1 (Secure Hash Algorithm 1) – это алгоритм хеширования, созданный Агентством национальной безопасности США совместно с Национальным институтом стандартов и технологий, и опубликованный в 1995 году в качестве стандарта. SHA-2 – вторая версия “безопасного алгоритма хеширования”, опубликованная в 2002 году. Данные функции хеширования построены на основе тех же принципов, что и MD4, поэтому их также можно рассматривать в качестве представителей этого семейства.

Рассмотрим хэш-функцию MD5, которая достаточно широко используется на сегодняшний день (например, для безопасного хранения паролей в Windows и Linux).

MD5 принимает на входе битовую последовательность произвольной длины, обрабатывает ее блоками по 512 бит и возвращает на выходе 128-битовое хэш-значение.

Алгоритм преобразований выглядит следующим образом.

1. Добавление битов заполнения.

Исходная последовательность дополняется некоторым количеством битов, чтобы ее битовая длина была сравнима с 448 по модулю 512. Для этого в конец последовательности дописывается единичный бит, а затем нужное количество нулевых битов.

Добавление битов происходит, даже если длина сообщения уже сравнима с 448 по модулю 512.

2. Добавление длины последовательности.

Длина исходной последовательности представляется 64-битовым значением и дописывается в конец последовательности, полученной на первом шаге. Если длина последовательности превосходит $2^{64} – 1$, то учитываются только младшие 64 бита.

Теперь последовательность содержит целое число 512-битовых блоков.

3. Инициализация буфера.

Для хранения результатов промежуточных вычислений используются 4 32-битовых слова A, B, C, D, в которые в начале записываются следующие значения:

A = 0x01234567,

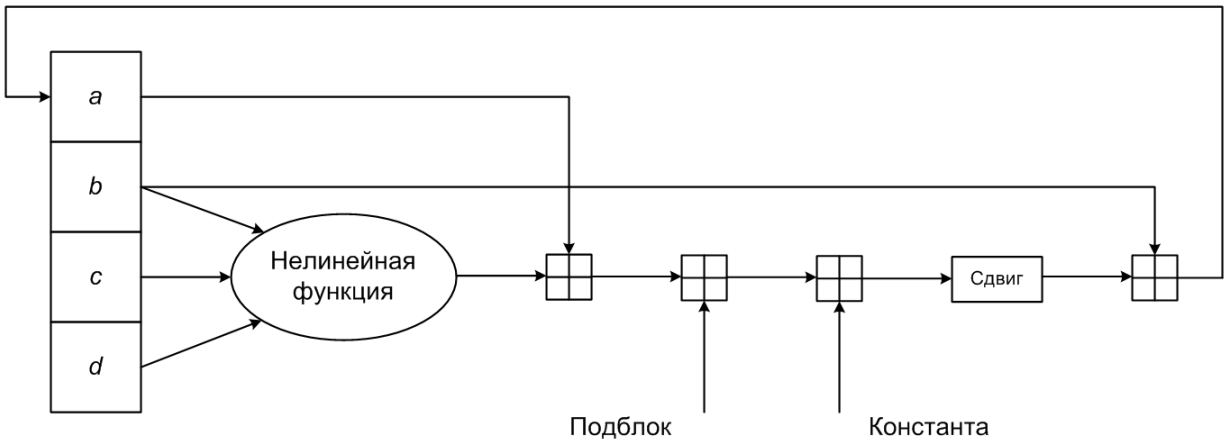
B = 0x89abcdef,

C = 0xfedcba98,

D = 0x76543210.

Начальное состояние (A, B, C, D) называется инициализирующим вектором.

Шаговая функция хеширования MD5 представлена на рисунке 5.



Шаговая функция хеширования MD5

Каждый 512-битовый блок разбивается на 16 32-битовых подблоков и обрабатывается в 4 этапа. На каждом этапе три из четырех переменных *a*, *b*, *c*, *d* преобразуются посредством одной из заданных нелинейных функций, далее следует ряд операций, одна из которых заключается в сложении текущего результата преобразований с очередным подблоком.

Таким образом, шаговая функция хеширования повторяется 64 для каждого блока входной последовательности данных.

Глава 7. Коды аутентичности сообщений. Электронная подпись

На самом первом занятии, рассматривая основные положения криптографии, мы сказали, что одной из основных задач, для решения которых предназначены криптографические методы защиты информации, является обеспечение целостности данных. Обеспечение целостности достигается посредством добавления некоторой избыточной информации, являющейся своего рода идентификатором, к передаваемым или хранимым данным. Для вычисления такой информации могут быть использованы функции хеширования.

Другой, не менее важной, задачей является аутентификация в широком смысле, то есть проверка подлинности сторон, данных, подтверждение авторства и т. д.

Для одновременного решения задач обеспечения целостности данных (аутентификации данных) и аутентификации источника данных в криптографии используются так называемые коды аутентичности сообщений, которые в отечественной терминологии также называют имитовставками.

Код аутентичности сообщения (Message Authentication Code – MAC) – некоторая дополнительная информация, добавляемая к исходным данным, вырабатываемая по определенному алгоритму и зависящая от исходных данных и секретного элемента – ключа аутентификации.

Основные подходы к вычислению MAC – это CBC-MAC и HMAC. Первый основан на использовании симметричного блочного шифра, второй – на использовании однонаправленной хэш-функции.

Алгоритм CBC-MAC построен следующим образом: исходные данные зашифровываются с помощью блочного шифра в режиме CBC, затем откидываются все блоки шифртекста кроме последнего. Этот последний блок и будет являться значением MAC. Естественно, что ключ аутентификации используется в качестве ключа шифрования блочного шифра.

Если исходные данные представляют собой последовательность блоков m_1, m_2, …, m_n, то значение MAC будет результатом следующих преобразований:

$H_0 \leftarrow IV$;

$H_i \leftarrow E_k(m_i \oplus H_{i-1})$;

$MAC \leftarrow H_n$.

IV – это вектор инициализации (аналог синхропосылки в ГОСТе), который в алгоритме CBC-MAC обычно берется нулевым.

В общем случае в качестве значения MAC можно взять не весь последний блок, а только его некоторые биты.

Если помимо обеспечения целостности данных и подтверждения подлинности источника сообщений необходимо обеспечить конфиденциальность передаваемых данных, то есть осуществить их зашифрование, то обязательных требованием является использование разных ключей для шифрования и вычисления кода аутентичности.

Алгоритм HMAC вычисляет значение MAC по следующей формуле:

$MAC = h(K \oplus a \parallel h(K \oplus b \parallel m))$,

где h – функция хеширования, а a и b – битовые строки, используемые для дополнения входных данных до полных блоков.

Конкретные реализации приведенных алгоритмов будут настолько надежны, насколько надежны блочный шифр или функция хеширования, положенные, соответственно, в их основу.

Помимо рассмотренных существуют и другие алгоритмы вычисления MAC. В частности, один из режимов работы ГОСТ 28147-89 – режим выработки имитовставки – непосредственно предназначен для вычисления MAC, называемого в этом случае имитовставкой.

При передаче информации по каналу связи значение MAC присоединяется к исходным данным, в случае необходимости результат зашифровывается и пересылается адресату. Получатель, знающий как ключ шифрования, так и ключ аутентификации, расшифровывает полученные данные, если они были до этого зашифрованы, вычисляет значение MAC и сравнивает с полученным. Если результат такой проверки положителен, то адресат получает уверенность в следующем:

– в целостности сообщения, то есть в том, что оно не было изменено при передаче;

– в его оригинальности, то есть в том, что сообщение было послано именно тем пользователем, с которым ведет переписку адресат.

Однако MAC не сможет обеспечить уверенности в отсутствии ренегатства, когда отправитель впоследствии будет утверждать, что не посылал сообщения. Помимо этого возможна фальсификация сообщения или его изменение получателем. В первом случае пользователь формирует некоторое сообщение, используя все известные ему законные реквизиты, и заявляет, что получил это сообщение от другого пользователя. Во втором случае получатель может внести какие-либо изменения в сообщение, отправленное другим пользователем, и утверждать, что это сообщение было им получено именно в таком виде. С другой стороны, отправитель может внести изменения в уже переданное сообщение и сказать впоследствии, что он отправлял его именно в таком виде.

Таким образом, MAC может обеспечить должную безопасность, когда участники информационного обмена доверяют друг другу и хотят защититься от стороннего злоумышленника. В том случае, когда между сторонами отсутствует доверие, и потенциальным злоумышленником может быть одна из них, необходимы другие средства защиты.

Таким средством является цифровая подпись или, по-другому, электронная цифровая подпись (ЭЦП). Мы уже вводили определение ЭЦП и краткое описание ее свойств, теперь поговорим о конкретных алгоритмах, используя уже полученные знания из других областей криптографии.

Само понятие электронной цифровой подписи было предложено Диффи и Хеллманом в их статье «Новые направления в криптографии», о которой мы не раз говорили. Но первая практическая разработка принадлежит Райвесту, Шамиру и Адлеману, создателям криптосистемы RSA.

Существуют две основные схемы построения электронной цифровой подписи: на основе симметричных криптосистем и на основе криптографии с открытым ключом.

Первая схема предусматривает наличие посредника или арбитра, обладающего доверием всех участников информационного обмена. Каждый пользователь имеет с посредником общий ключ шифрования, отличный от аналогичных ключей других пользователей. Эти ключи выдаются пользователям посредником и могут использоваться многократно.

Если пользователь A хочет подписать сообщение и отправить его пользователю B, то он зашифровывает его на ключе k_A, являющемся общим для него и посредника, и отправляет посреднику T. T расшифровывает полученное сообщение, добавляет к нему утверждение, что он получил это сообщении от A, зашифровывает результат на ключе k_B, который является общим для него и пользователя B, и отправляет пользователю B. B расшифровывает сообщение и видит сами исходные данные и подтверждение посредника, что это сообщение было отправлено пользователем A.

Такая схема достаточно хорошо работает, но проблемой является нахождение непогрешимого посредника, не важно, человек это, или техническое устройство.

Схемы электронной цифровой подписи на основе асимметричных криптосистем являются наиболее распространенными на сегодняшний день, и именно такие схемы мы будем рассматривать в дальнейшем.

Особый интерес представляет криптосистема RSA, поскольку ее можно непосредственно использовать для подписывания данных.

Рассмотрим систему ЭЦП на основе криптосистемы RSA.

Допустим, пользователь А хочет отправить подписанное сообщение пользователю В. Он должен выполнить следующее преобразование:

$$SIG(m) = E_{e_B,n_B}\left(E_{d_A,n_A}(m)\right).$$

То есть пользователь А сначала зашифровывает данные с помощью своего закрытого ключа, а затем зашифровывает результат с помощью открытого ключа пользователя В.

Затем пользователь А передает пользователю В пару $(m, SIG(m))$.

Пользователь В, получив сообщение пользователя А, с помощью своего закрытого ключа расшифровывает сообщение, выполнив преобразование

$$E_{d_A,n_A} = E_{d_B,n_B}\left(SIG(m)\right) = E_{d_B,n_B}\left(E_{e_B,n_B}\left(E_{d_A,n_A}(m)\right)\right) = E_{d_A,n_A}(m).$$

Затем он с помощью открытого ключа пользователя А выполняет преобразование

$$\mathrm{m} = \mathrm{E}_{\mathrm{e_A},n_A}\left(E_{d_A,n_A}(m)\right).$$

После этого пользователь В может сравнить результат всех преобразований с полученным сообщением *m* и сделать вывод о его подлинности или подложности.

Эллиптические кривые

Перейдем к изучению существующих на сегодняшний день стандартов формирования электронной цифровой подписи. По традиции, начнем с отечественных стандартов.

Первым российским стандартом цифровой подписи был стандарт ГОСТ Р 34.10-94, в 2001 году его сменил стандарт ГОСТ Р 34.10-2001. Оба стандарта очень похожи, отличие заключается в том, что новый стандарт основан на сложности дискретного логарифмирования в группе точек эллиптической кривой над конечным полем, в то время как в старом стандарте часть операций проводилась над конечным полем по модулю простого числа.

В современной криптографии выделяют целый раздел, называемый эллиптической криптографией, который изучает асимметричные криптосистемы, основанные на эллиптических кривых над конечными полями. Важным преимуществом эллиптических кривых является то, что стойкость криптографических алгоритмов на их основе достигается при использовании групп меньших порядков, чем в случае конечных полей, что уменьшает затраты на передачу и хранение информации.

Введем основные теоретические положения эллиптической криптографии.

Пусть F – произвольное поле. Эллиптической кривой E над полем F называется гладкая кривая, задаваемая уравнением вида

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,\ a_i \in F,\ i = \overline{1,6}.$$

Обозначим через $E(F)$ множество, состоящее из точек $(x, y) \in F^2$, удовлетворяющих данному уравнению, и «бесконечно удаленной» точки 0.

Если характеристика поля F отлична от 2 и 3 (char F ≠ 2, 3), то эллиптическая кривая имеет вид

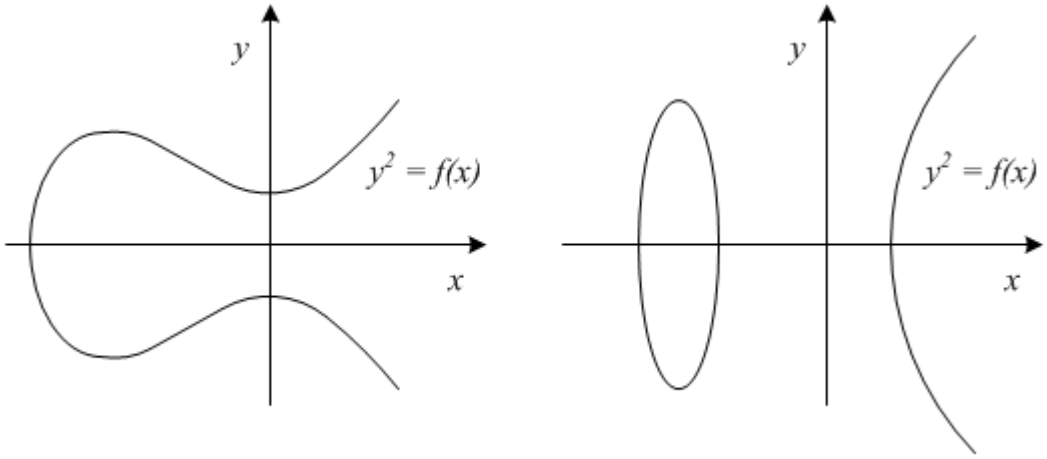
$$y^2 = x^3 + ax + b,\ a, b \in F.$$

Условие гладкости кривой в этом случае означает, что кубический многочлен $f(x) = x^3 + ax + b$ не имеет кратных корней. Это выполняется тогда и только тогда, когда дискриминант Δ многочлена $f(x)$, равный

$$\Delta = -4a^3 - 27b^2,$$

отличен от нуля.

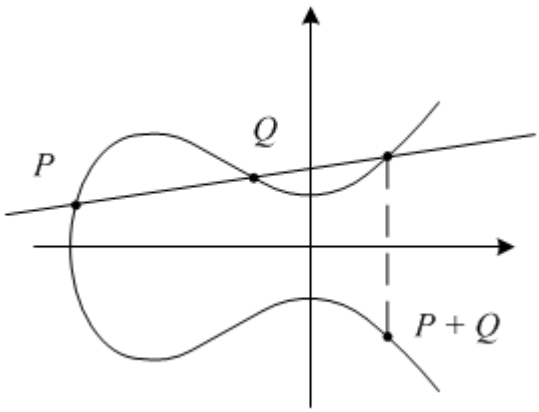
На рисунке ниже показано графическое представление эллиптической кривой над полем вещественных чисел *R*.



Эллиптические кривые

Зададим алгебраическую операцию сложения в группе точек эллиптической кривой. Проиллюстрируем эту операцию на рисунке ниже.

Возьмем на кривой *E* две точки *P* и *Q* и проведем через них прямую. В общем случае эта прямая имеет третью точку с кривой *E*. Отразим эту точку от оси 0*x* и назовем полученную точку суммой точек *P* и *Q* (*P* + *Q*).



Сложение точек

Пусть $P = (x_1, y_1)$, $Q = (x_2, y_2)$. Тогда координаты точки $P + Q = (x_3, y_3)$ будем вычислять следующим образом:

1. $P \neq Q$

$$\begin{cases} x_3 = \left(\dfrac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2, \\ y_3 = \left(\dfrac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1. \end{cases}.$$

2. $P = Q,\ \ P + Q = P + P = 2P$.

$$\begin{cases} x_3 = \left(\dfrac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1, \\ y_3 = \left(\dfrac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1. \end{cases}$$

Таким образом, множество точек эллиптической кривой $E(F)$ над полем F образует абелеву группу относительно сложения. Если F – конечное поле, то эта группа – конечная абелева группа.

DSS

В 2000 г. в США была принята новая редакция стандарта на выработку и верификацию электронной (цифровой) подписи DSS (Digital Signature Standard), согласно которому электронная подпись может вырабатываться по одному из трех алгоритмов: DSA (Digital Signature Algorithm), ANSI X9.31 (RSA DSA) или ANSI X9.63 (Elliptic Curve Digital Signature Algorithm).

Алгоритм выработки электронной подписи с помощью асимметричной криптосистемы RSA был рассмотрен в предыдущем разделе. Алгоритмы DSA и ECDSA построены по одной схеме – отличие заключается в том, что первый основан на проблеме дискретного логарифмирования в конечном поле, а второй – на проблеме дискретного логарифмирования в группе точек эллиптической кривой. В современной криптографии выделяют целый раздел, называемый эллиптической криптографией, который изучает асимметричные криптосистемы, основанные на эллиптических кривых над конечными полями. Важным преимуществом эллиптических кривых является то, что стойкость криптографических алгоритмов на их основе достигается при использовании групп меньших порядков, чем в случае конечных полей, что уменьшает затраты на передачу и хранение информации.

Согласно рассмотренному ранее материалу Общепризнанная схема (модель) цифровой подписи охватывает три процесса:

– генерация ключей (подписи и проверки);

– формирование подписи;

– проверка подписи.

Рассмотрим данные этапы в составе алгоритма ECDSA.

1. Выбор параметров.

Прежде всего необходимо задать параметры эллиптической кривой, однако этот момент опустим, поскольку рассмотрение соответствующего математического аппарата выходит за пределы данной лекции.

Также на данном этапе генерируется ключевая пара пользователя. В качестве секретного ключа выбирается целое число d, $0 < d < n$, где n – порядок базовой точки эллиптической кривой G. Затем вычисляется $Q = dG$ и публикуется в качестве открытого ключа пользователя.

2. Выработка подписи

Для подписания сообщения m пользователь:

1. выбирает случайное число k, $0 < k < n - 1$;

2. вычисляет $kG = (x_1, y_1)$, $r = x_1 \bmod n$. Если r = 0, то возврат к шагу 1;

3. вычисляет $k^{-1} \bmod n$;

4. вычисляет хэш-значение сообщения m по алгоритму SHA $e = \mathrm{SHA}(m)$;

5. вычисляет $s = k^{-1} (e + dr) \mod n$. Если $s = 0$, то возврат к шагу 1;

6. подписью сообщения m является пара (r, s).

3. Верификация (проверка) электронной подписи

Для проверки подписи получатель осуществляет следующее:

1. проверяет, что r и s лежат в интервале $(0, n)$;

2. вычисляет $e = \mathrm{SHA}(m)$;

3. вычисляет $w = s^{-1} \bmod n$;

4. $u_1 = ew \bmod n$ и $u_2 = rw \bmod n$;

5. вычисляет $X = u_1G + u_2Q$. Если X является бесконечно удаленной точкой, то подпись отвергается, в противном случае вычислить $v = x_1 \bmod n$, где $X = (x_1, y_1)$;

6. принять подпись только в том случае, если $v = r$.

Для обеспечения криптостойкости алгоритма длина элементов ключевой пары должна составлять не меньше 160 бит.

Глава 8. Управление ключами. Распределение симметричных ключей

В ходе предыдущих лекций мы рассмотрели все основные направления современной криптографии: симметричные криптосистемы, начиная с классических, функции хеширования, криптосистемы с открытым ключом, системы электронной цифровой подписи. Теперь пришло время более детально рассмотреть проблему управления ключами, о которой мы уже говорили раньше, когда рассматривали концепцию криптографии с открытым ключом.

Основной задачей, относящейся к проблеме управления ключами, да, и вообще являющейся одной из фундаментальных задач криптографии, является распределение ключей. Возможные способы решения этой задачи уже упоминались. Прежде всего, это физическое распределение, которое до семидесятых годов прошлого века было единственным безопасным путем распределения ключей. Но такой способ практически не применим для систем с достаточно сложной структурой и большим числом пользователей. Кроме того, криптостойкость всей системы в этом случае будет зависеть не столько от криптографических аспектов, сколько от надежности курьера, которого можно подкупить, захватить и, наконец, убить. Но это уже область не криптографии, а скорее психологии, социологии и, наконец, криминологии.

Другое решение задачи распределения ключей шифрования основано на использовании протоколов с секретным ключом. В этом случае долговременные ключи распределены между пользователями системы и неким центром, обычно называемым центром доверия (а также доверенным центром или доверенным лицом), который можно использовать для генерирования ключей и обмена между любыми двумя пользователями всякий раз, когда в этом возникает необходимость. Такой способ распределения предусматривает работу центра и пользователей в режиме онлайн. Кроме того, долговременные ключи при этом должны распределяться физическим путем.

Рассмотрим некоторые протоколы распределения ключей между пользователями, основанные на симметричной криптографии. Протоколы, основанные на криптографии с открытым ключом, а также различия между протоколами двух данных типов рассмотрим позднее.

Для начала введем два определения, разделив все многообразие симметричных ключей на два отдельных класса.

Статичный (долговременный) ключ – ключ, который используется в течение достаточно большого периода времени. Точное значение слова «большого» зависит от приложений, где ключ используется, и период времени, о котором идет речь, может варьироваться от нескольких часов до нескольких лет. Компрометация (раскрытие) статичного ключа обычно считается главной проблемой с потенциально катастрофическими последствиями.

Сеансовый или эфемерный (кратковременный) ключ – ключ, который применяется лишь малое время, от нескольких секунд до одного дня. Он используется для обеспечения конфиденциальности в одном сеансе связи. Раскрытие сеансового ключа повлечет за собой нарушение секретности данного сеанса, но не должно повлиять на безопасность всей системы.

Обычно за каждым пользователем системы закрепляется единственный долговременный ключ, используя который он связывается с доверенным центром. Когда двое пользователей хотят обменяться некоторой информацией, они формируют общий сеансовый ключ, который будет использован только для передачи одного или нескольких сообщений. Этот ключ формируется с посредническим участием доверенного центра при помощи определенного протокола.

Такое решение на порядки сокращает объем ключевой информации в системе. Поясним это следующим образом.

Обозначим количество пользователей в системе n. Если не использовать сеансовые ключи, то каждому пользователю понадобится $n-1$ долговременных ключей для безопасного обмена информацией со всеми другими пользователями системы. Тогда общее количество ключей в такой системе будет равно $\frac{n(n-1)}{2}$.

Если же использовать рассмотренный выше подход, то понадобится всего лишь n долговременных ключей, по одному на каждого пользователя.

Говоря о проблеме управления ключами, также следует отметить такие относящиеся к ее компетенции вопросы, как выбор (генерация) секретных ключей, их правильное хранение и уничтожение. Секретный ключ должен быть действительно случайным и генерироваться посредством некоторого физического источника случайных данных. Кроме того, любое средство криптографической защиты информации должно обладать собственным механизмом уничтожения устаревших ключей. Эту проблему нельзя перекладывать на операционную систему, поскольку удаление файла с носителя информации или освобождение участка оперативной памяти, не стирает само содержимое этой области памяти, а только сообщает системе, что ячейки этой области теперь свободны и на них могут быть записаны новые данные.

Глава 9. Инфраструктура открытого ключа

Рассмотрев на предыдущем занятии протоколы распределения ключей, основанные на симметричных криптосистемах, перейдем к рассмотрению аналогичных протоколов, основанных на криптографии с открытым ключом.

Как мы выяснили, основной проблемой симметричной криптографии является безопасная передача секретного ключа шифрования. Поэтому два абонента, использующие общий секретный ключ, могут быть уверены в том, что имеют идентичные копии этого ключа, и задача состоит в том, чтобы еще одна копия не оказалась у противника.

В криптографии с открытым ключом проблема совсем другая. Для того чтобы получить копию открытого ключа пользователя нет необходимости использовать тайное физическое распределение ключей. Асимметричная криптография, собственно, и

создавалась для того, чтобы решить эту проблему. Каждый пользователь может опубликовать свой открытый ключ в любом удобном для него месте – в общедоступной базе данных или на своей собственной веб-странице. Но в этом случае нельзя будет дать стопроцентной гарантии, что ключ, открытый для всеобщего доступа, действительно является открытым ключом соответствующего пользователя и не был подменен.

Процесс сопоставления открытого ключа физическому лицу или уполномоченному агенту, будь это человек, машина или процесс, называется привязкой. Один из способов привязки, общий для многих ситуаций, в которых владелец ключа должен присутствовать лично, заключается в передаче физического объекта, например, интеллектуальной карточки. Владение таким символом и знание некоторой дополнительной информации (пароля) считается достаточным для удостоверения личности. Но большинство владельцев ключей не являются людьми, это вычислительные устройства. Кроме того, таким образом мы опять приходим к физическому распределению ключевой информации, от которого так стремились избавиться с помощью криптографии с открытым ключом.

Поэтому необходимо иметь несколько форм привязки, которые можно было бы использовать в различных ситуациях. Основной инструмент привязки, который эксплуатируется в настоящее время, называется цифровым сертификатом. Это понятие заложено в концепции инфраструктуры открытого ключа.

Запишем определение.

Инфраструктура открытого ключа (Public Key Infrastructure – PKI) представляет собой систему, с помощью которой можно определить, кому принадлежит тот или иной открытый ключ.

Для начала рассмотрим стандартную концепцию инфраструктуры открытого ключа, а затем обсудим ряд проблем, которые возникают при ее использовании в реальном мире.

Базовым понятием данной концепции является понятие центра сертификации (центра сертификатов). Это центральный орган, служащий посредником между пользователями и отвечающий за аутентичность (подлинность) их открытых ключей.

Принцип работы центра сертификации заключается в следующем:

— Центр сертификации обладает парой «открытый ключ-закрытый ключ».
— Каждый из пользователей обладает надежной копией открытого ключа центра сертификации. Так как данный открытый ключ является долговременным, то его аутентичное распространение не является сложной задачей.
— Чтобы присоединиться к инфраструктуре открытого ключа, пользователь A генерирует свою ключевую пару и, сохраняя закрытый ключ в секрете, передает открытый ключ центру сертификации.
— Центр сертификации проверяет, что пользователь A действительно является тем, за кого себя выдает и ставит цифровую подпись на строке вида (пользователь A, открытый ключ пользователя A). Эта строка данных вместе с подписью называется цифровым сертификатом. Центр выдаст этот сертификат только в том случае, если сможет достоверно убедиться, что открытый ключ действительно принадлежит пользователю A.
— Если теперь пользователь A вышлет пользователю B свой открытый ключ, содержащийся в цифровом сертификате, то у того не будет причин сомневаться в подлинности ключа, поскольку он доверяет центру сертификации.
— Пользователь B может также получить цифровой сертификат, содержащий его открытый ключ, и отправить пользователю A.
— Имея открытые ключи друг друга, причем подлинность этих ключей гарантирует центр сертификации, пользователи A и B могут согласовать симметричный ключ для последующего безопасного общения.

Важным дополнением данной схемы является использование не одного, а нескольких центров сертификации, которые образуют иерархию, что приводит к появлению многоуровневой структуры сертификатов.

Кроме того, обычной практикой является подписывание одним центром сертификатов открытого ключа другого центра сертификатов и наоборот — процесс, известный как перекрестная (взаимная) сертификация.

Необходимость в перекрестной сертификации возникает, когда функционирует более одного центра сертификатов, так как пользователь может не иметь достоверной копии открытого ключа определенного центра сертификатов, нужной, чтобы проверить подлинность открытого ключа, содержащегося в выданном сертификате. Эта проблема решается перекрестной сертификацией, когда открытый ключ одного центра сертификатов подписывается другим центром сертификатов. Пользователь сначала проверяет открытый ключ соответствующего центра сертификатов, а затем ключ клиента, подписанный этим центром сертификатов.

Список литературы

1. Основы криптографии: учебное пособие для вузов / А.П. Алферов, А.Ю. Зубов, А.С. Кузьмин, А.В. Черемушкин. — 3-е изд., испр. и доп. — М.: Гелиос АРВ, 2005. — 479 с.
2. Основы современной криптографии: учебный курс / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов. — 2-е изд., перераб. и доп. — М.: Горячая линия-Телеком, 2002. — 176 с.
3. Смарт Н. Криптография: учебник для вузов. — М.: Техносфера, 2005. — 525 с.
4. Росошек С.К. Специальные главы математики (математические основы криптографии): учебное пособие. Часть 1. — Томск: ТУСУР, 2004. — 93 с.
5. Росошек С.К. Специальные главы математики (математические основы криптографии): учебное пособие. Часть 2. — Томск: ТУСУР, 2005. — 190 с.

Вопросы для самоконтроля

1. Перечислите и кратко охарактеризуйте основные задачи обеспечения информационной безопасности, решаемые с помощью криптографических методов.
2. Раскройте определения: шифрование, зашифрование, расшифрование, дешифрование.
3. Чем шифрование отличается от кодирования?
4. Приведите известные вам классификации криптосистем.
5. Укажите основные отличия между современной и классической криптографией.
6. Сравните аффинный шифр и шифр Хилла с точки зрения криптостойкости.
7. Опишите способы криптоанализа:
8. аффинного шифра;
9. шифра Хилла;
10. шифра гаммирования.
11. Укажите основные отличия между современными и классическими блочными шифрами.
12. Перечислите режимы работы ГОСТ 28147-89. Для чего служит каждый из данных режимов?
13. Сравните DES и ГОСТ 28147-89.
14. Сравните AES и ГОСТ 28147-89.
15. Перечислите основные свойства хеш-функций.
16. Чем хеширование отличается от выработки контрольных сумм?
17. Чем хеширование отличается от выработки имитовставки?
18. Укажите два подхода к построению функций хеширования.
19. Укажите основной недостаток кодов аутентичности сообщений.
20. В чем заключается проблема управления симметричными ключами?
21. Сравните криптосистему RSA и криптосистему Рабина.
22. Сравните криптосистему RSA и криптосистему Эль-Гамаля.
23. Решение каких задач обеспечивает электронная подпись?
24. Как построить схему выработки и проверки электронной подписи на основе криптосистемы RSA?
25. Что такое эллиптическая криптография?
26. Дайте понятие криптографического протокола.

Задачи

1. Зашифровать слово CRYPTOGRAPHY подстановочным шифром, самостоятельно выбрав ключ шифрования из симметрической группы S26.

2. Зашифровать слово CRYPTOGRAPHY перестановочным шифром, самостоятельно выбрав ключ шифрования из симметрической группы

    2.1. S3.

    2.2. S4.

    2.3. S6.

3. Зашифровать слово CRYPTOGRAPHY аффинным шифром, самостоятельно выбрав ключ шифрования. Пояснить, какие особенности данного шифра проявились в шифртексте.

4. Зашифровать слово CRYPTOGRAPHY аффинным рекуррентным шифром, самостоятельно выбрав ключ шифрования. Указать, какие преимуществами обладает аффинный рекуррентный шифр по сравнению с аффинным.

5. Зашифровать слово CRYPTOGRAPHY шифром Хилла, самостоятельно выбрав ключевую матрицу размерности

    5.1. 2x2.

    5.2. 3x3.

    5.4. 4x4.

6. Выбрать открытый текст и зашифровать его таким образом, чтобы показать преимущества рекуррентного шифра Хилла над шифром Хилла.

7. Оценить, какой из двух шифров труднее взломать методом грубой силы:

    7.1. простой замены или перестановочный?

    7.2. простой замены или аффинный?

    7.3. Аффинный или шифр Хилла?

8. Зашифровать слово CRYPTOGRAPHY табличным шифром гаммирования, самостоятельно выбрав ключ шифрования.

9. Зашифровать слово CRYPTOGRAPHY шифром Вижинера, самостоятельно выбрав

    9.1. циклический ключ шифрования.

    9.2. самоключ по открытому тексту.

    9.3. самоключ по шифртексту.

10. Зашифровать слово CRYPTOGRAPHY шифром простой замены над конечным полем, самостоятельно выбрав ключ шифрования и неприводимый многочлен.

11. Выработать общий секретный элемент данных по протоколу Диффи-Хеллмана, взяв значения p = 127, g = 12, a x и y выбрав самостоятельно.

12. Зашифровать слово CRYPTOGRAPHY по алгоритму RSA, самостоятельно выбрав ключевую пару.

13. Зашифровать слово CRYPTOGRAPHY по алгоритму Эль-Гамаля, самостоятельно выбрав ключевую пару.

14. Зашифровать слово CRYPTOGRAPHY по алгоритму Рабина, самостоятельно выбрав ключевую пару.

15. Вычислить

    5.1. 682126.

    5.2. 346234.

    5.3. 807452.

16. Определить (с вероятностью не менее 0,99609375), является ли простым число

    6.1. 353.

    6.2. 489.

    6.3. 1213.