МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
ЛУГАНСКОЙ НАРОДНОЙ РЕСПУБЛИКИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
ЛУГАНСКОЙ НАРОДНОЙ РЕСПУБЛИКИ
«ЛУГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ ВЛАДИМИРА ДАЛЯ»

Стахановский инженерно-педагогический институт менеджмента
Кафедра информационных систем

КОНСПЕКТ ЛЕКЦИЙ

по дисциплине

«МЕТОДЫ И СРЕДСТВА ЗАЩИТЫ КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ»

для студентов направления подготовки
Профессиональное обучение (по отраслям)
профиль
«Информационные технологии и системы»
(в 3-х частях). Часть 3

Луганск 2023

УДК 612.313

Рекомендовано к изданию Учебно-методическим советом
ГОУ ВО ЛНР «ЛГУ им. В. ДАЛЯ»
(протокол № от 2023 г.)

Конспект лекций по дисциплине «**Методы и средства защиты компьютерной информации**» для студентов направления подготовки **Профессиональное обучение (по отраслям)**, профиль **«Информационные технологии и системы» (**в 3-х частях). Часть 3. / Сост.: И.В.Ганзенко. – **Стаханов**: ГОУ ВО ЛНР «ЛГУ им. В. Даля», 2023. – 50 с.

Конспект лекций содержит материал по 5 темам предмета, описание которых сопровождается теоретическими сведениями, к каждой теме приведены вопросы для самопроверки.

Составитель: ст. преп. Ганзенко И.В.

Ответственный за выпуск: доц. Карчевский В.П.

Рецензент: доц. Карчевская Н.В.

СОДЕРЖАНИЕ

Введение……..... 3

Лекция 8 Криптографические методы защиты информации. Основные понятия и определения. Подстановочные и перестановочные шифры. Шифры Цезаря, Виженера, Вернама……..... 5

Лекция 9 Основные понятия, относящиеся к обеспечению целостности сообщений с помощью MAC и хэш-функций; простые хэш-функции и хэш-функция MD5…… 15

Лекция 10 Блочные и поточные алгоритмы симметричного шифрования. Стандарты и алгоритмы: американский DES, отечественный ГОСТ, режимы их выполнения. Основные понятия, относящиеся к алгоритмам симметричного шифрования. Определение устойчивости алгоритма. Сеть Фейштеля…… 20

Лекция 11 Асимметричные системы шифрования (системы с открытым ключом). RSA. Функции дискретного логарифмирования и основанные на ней алгоритмы: схема Диффи-Хеллмана, схема Эль-Гамаля. Схема RSA: алгоритм шифрования, его оборачиваемость, вопросы устойчивости…….. 34

Лекция 12 Основные требования к цифровой подписи, прямая и арбитражная цифровые подписи, стандарты цифровой подписи ГОСТ 3410 и DSS……..... 40

Список использованных источников……... 49

Введение

Наступивший новый этап в развитии обмена информацией, который характеризуется интенсивным внедрением современных информационных технологий, широким распростр локальных, корпоративных и глобальных сетей во всех сферах жизни цивилизованного государства, создает новые возможности и качество информационного обмена. В связи с этим проблемы информационной безопасности (ИБ) приобретают первостепенное значение, актуальность и важность которых обусловлена следующими факторами:

– высокие темпы роста парка персональных компьютеров (ПК), применяемых в разных сферах деятельности, и как следствие, резкое расширение круга пользователей, имеющих непосредственный доступ к вычислительным сетям и информационным ресурсам;

– увеличение объемов информации, накапливаемой, хранимой и обрабатываемой с помощью ПК и других средств автоматизации;

– бурное развитие аппаратно-программных средств и технологий, не соответствующих современным требованиям безопасности;

– несоответствие бурного развития средств обработки информации и проработки теории ИБ разработки международных стандартов и правовых норм, обеспечивающих необходимый уровень защиты информации (ЗИ);

– повсеместное распространение сетевых технологий, создание единого информационно-коммуникационного мирового пространства на базе сети Интернет, которая по своей идеологии не обеспечивает достаточного уровня ИБ.

Указанные выше факторы создают определенный спектр угроз ИБ на уровне личности, общества и государства. Средством нейтрализации значительной их части является формирование теории ИБ и методологии защиты информации.

Целью курса является формирование профессиональной компетентности на основе системы теоретических и методологических знаний и специальных умений в области информационной безопасности и их использования в профессиональной деятельности будущего специалиста.

Лекция 8 Криптографические методы защиты информации. Основные понятия и определения. Подстановочные и перестановочные шифры. Шифры Цезаря, Виженера, Вернама

План изложения

1. Классификация криптографических методов
2. Системы подстановок
3. Подстановка Цезаря
4. Многоалфавитные системы. Системы одноразового использования
5. Системы шифрования Вижинера

1 Классификация криптографических методов

Криптография является методологической основой современных систем обеспечения безопасности информации в компьютерных системах и сетях.

Криптография (от др.-греч. κρυπτός - скрытый и γράφω - пишу) - наука о методах обеспечения конфиденциальности (невозможности прочтения информации посторонним) и аутентичности (целостности и подлинности авторства, а также невозможности отказа от авторства) информации.

Изначально криптография изучала методы шифрования информации - обратимого преобразования открытого (исходного) текста на основе секретного алгоритма или ключа в шифрованный текст (шифротекст). Традиционная криптография образует раздел симметричных криптосистем, в которых зашифрование и расшифрование проводится с использованием одного и того же секретного ключа.

Помимо этого раздела современная криптография включает в себя асимметричные криптосистемы, системы электронной цифровой подписи (ЭЦП), хеш-функции, управление ключами, получение скрытой информации, квантовую криптографию.

Криптография не занимается: защитой от обмана, подкупа или шантажа законных абонентов, кражи ключей и других угроз информации, возникающих в защищенных системах передачи данных.

Основным достоинством криптографических методов является то, что они обеспечивают высокую гарантированную стойкость защиты, которую можно рассчитать и выразить в числовой форме (средним числом операций или временем, необходимым для раскрытия зашифрованной информации или вычисления ключей).

К числу основных недостатков криптографических методов следует отнести:

– значительные затраты ресурсов (времени, производительности процессоров) на выполнение криптографических преобразований информации;

– трудности совместного использования зашифрованной (подписанной) информации, связанные с управлением ключами (генерация, распределение и т.д.);

– высокие требования к сохранности секретных ключей и защиты открытых ключей от подмены.

Известны различные подходы к классификации методов криптографического преобразования информации. По виду воздействия на исходную информацию методы криптографического преобразования информации могут быть разделены на четыре группы (рис. 1).

Рисунок 1 - Классификация методов криптографического преобразования информации

Процесс **шифрования** заключается в проведении обратимых математических, логических, комбинаторных и других преобразований исходной информации, в результате которых зашифрованная информация представляет собой хаотический набор букв, цифр, других символов и двоичных кодов. Для шифрования информации используются алгоритм преобразования и ключ. Как правило, алгоритм для определенного метода шифрования является неизменным. Исходными данными для алгоритма шифрования служат информация, подлежащая зашифрованию, и ключ шифрования. Ключ содержит управляющую информацию, которая определяет выбор преобразования на определенных шагах алгоритма и величины операндов, используемые при реализации алгоритма шифрования. В отличие от других методов криптографического преобразования информации, методы **стеганографии** позволяют скрыть не только смысл хранящейся или передаваемой информации, но и сам факт хранения или передачи закрытой информации. В компьютерных системах практическое использование стеганографии только начинается, но проведенные исследования показывают ее перспективность. В основе всех методов стеганографии лежит маскирование закрытой информации среди открытых файлов. Обработка мультимедийных файлов в КС открыла практически неограниченные возможности перед стеганографией.

Существует несколько методов скрытой передачи информации. Одним из них является простой метод скрытия файлов при работе в операционной системе MS-DOS. За текстовым открытым файлом записывается скрытый двоичный файл, объем которого много меньше текстового файла. В конце текстового файла помещается метка EOF (комбинация клавиш Control и Z). При обращении к этому текстовому файлу стандартными средствами ОС считывание прекращается по достижению метки EOF, и скрытый файл остается недоступен. Для двоичных файлов никаких меток в конце файла не предусмотрено. Конец такого файла определяется при обработке атрибутов, в

которых хранится длина файла в байтах. Доступ к скрытому файлу может быть получен, если файл открыть как двоичный. Скрытый файл может быть зашифрован. Если кто-то случайно обнаружит скрытый файл, то зашифрованная информация будет воспринята как сбой в работе системы.

Графическая и звуковая информация представляются в числовом виде. Так в графических объектах наименьший элемент изображения может кодироваться одним байтом. В младшие разряды определенных байтов изображения в соответствии с алгоритмом криптографического преобразования помещаются биты скрытого файла. Если правильно подобрать алгоритм преобразования и изображение, на фоне которого помещается скрытый файл, то человеческому глазу практически невозможно отличить полученное изображение от исходного. Очень сложно выявить скрытую информацию и с помощью специальных программ. Наилучшим образом для внедрения скрытой информации подходят изображения местности: фотоснимки со спутников, самолетов и т. п. С помощью средств стеганографии могут маскироваться текст, изображение, речь, цифровая подпись, зашифрованное сообщение. Комплексное использование стеганографии и шифрования многократно повышает сложность решения задачи обнаружения и раскрытия конфиденциальной информации.

Содержанием процесса **кодирования** информации является замена смысловых конструкций исходной информации (слов, предложений) кодами. В качестве кодов могут использоваться сочетания букв, цифр, букв и цифр. При кодировании и обратном преобразовании используются специальные таблицы или словари. Кодирование информации целесообразно применять в системах с ограниченным набором смысловых конструкций. Такой вид криптографического преобразования применим, например, в командных линиях АСУ. Недостатками кодирования конфиденциальной информации является необходимость хранения и распространения кодировочных таблиц, которые необходимо часто менять, чтобы избежать раскрытия кодов статистическими методами обработки перехваченных сообщений.

Сжатие информации может быть отнесено к методам криптографического преобразования информации с определенными оговорками.

Целью сжатия является сокращение объема информации. В то же время сжатая информация не может быть прочитана или использована без обратного преобразования. Учитывая доступность средств сжатия и обратного преобразования, эти методы нельзя рассматривать как надежные средства криптографического преобразования информации. Даже если держать в секрете алгоритмы, то они могут быть сравнительно легко раскрыты статистическими методами обработки. Поэтому сжатые файлы конфиденциальной информации подвергаются последующему шифрованию. Для сокращения времени целесообразно совмещать процесс сжатия и шифрования информации.

Основным видом криптографического преобразования информации в КС является шифрование. Под шифрованием понимается процесс преобразования открытой информации в зашифрованную информацию (шифртекст) или процесс обратного преобразования зашифрованной

информации в открытую. Процесс преобразования открытой информации в закрытую получил название зашифрование, а процесс преобразования закрытой информации в открытую - расшифрование.

За многовековую историю использования шифрования информации человечеством изобретено множество методов шифрования или шифров. Методом шифрования (шифром) называется совокупность обратимых преобразований открытой информации в закрытую информацию в соответствии с алгоритмом шифрования. Большинство методов шифрования не выдержали проверку временем, а некоторые используются и до сих пор. Появление ЭВМ и КС инициировало процесс разработки новых шифров, учитывающих возможности использования ЭВМ как для зашифрования/расшифрования информации, так и для атак на шифр. Атака на шифр (криптоанализ) - это процесс расшифрования закрытой информации без знания ключа и, возможно, при отсутствии сведений об алгоритме шифрования.

Современные методы шифрования должны отвечать следующим требованиям:

– стойкость шифра противостоять криптоанализ (криптостойкость) должна быть такой, чтобы вскрытие его могло быть осуществлено только путем решения задачи полного перебора ключей;

– криптостойкость обеспечивается не секретностью алгоритма шифрования, а секретностью ключа;

– шифртекст не должен существенно превосходить по объему исходную информацию;

– ошибки, возникающие при шифровании, не должны приводить к искажениям и потерям информации;

– время шифрования не должно быть большим;

– стоимость шифрования должна быть согласована со стоимостью закрываемой информации.

Криптостойкость шифра является его основным показателем эффективности. Она измеряется временем или стоимостью средств, необходимых криптоаналитику для получения исходной информации по шифртексту, при условии, что ему неизвестен ключ.

Сохранить в секрете широко используемый алгоритм шифрования практически невозможно. Поэтому алгоритм не должен иметь скрытых слабых мест, которыми могли бы воспользоваться криптоаналитики. Если это условие выполняется, то криптостойкость шифра определяется длиной ключа, так как единственный путь вскрытия зашифрованной информации - перебор комбинаций ключа и выполнение алгоритма расшифрования. Таким образом, время и средства, затрачиваемые на криптоанализ, зависят от длины ключа и сложности алгоритма шифрования.

В качестве примера удачного метода шифрования можно привести шифр DES (Data Encryption Standard), применяемый в США с 1978 года в качестве государственного стандарта. Алгоритм шифрования не является секретным и был опубликован в открытой печати. За все время использования этого шифра не было обнародовано ни одного случая обнаружения слабых мест в алгоритме шифрования.

В конце 70-х годов использование ключа длиной в 56 бит гарантировало, что для раскрытия шифра потребуется несколько лет непрерывной работы самых мощных по тем временам компьютеров. Прогресс в области вычислительной техники позволил значительно сократить время определения ключа путем полного перебора. Согласно заявлению специалистов Агентства национальной безопасности США 56-битный ключ для DES может быть найден менее чем за 453 дня с использованием суперЭВМ Cray T3D, которая имеет 1024 узла и стоит 30 млн. долл. Используя чип FPGA (Field Programmably Gate Array - программируемая вентильная матрица) стоимостью 400 долл., можно восстановить 40-битный ключ DES за 5 часов. Потратив 10000 долл. за 25 чипов FPGA, 40-битный ключ можно найти в среднем за 12 мин. Для вскрытия 56-битного ключа DES при опоре на серийную технологию и затратах в 300000 долл. требуется в среднем 19 дней, а если разработать специальный чип, то - 3 часа. При затратах в 300 млн. долл. 56-битные ключи могут быть найдены за 12 сек. Расчеты показывают, что в настоящее время для надежного закрытия информации длина ключа должна быть не менее 90 бит.

Все методы шифрования могут быть классифицированы по различным признакам. Один из вариантов классификации приведен на рис. 2.

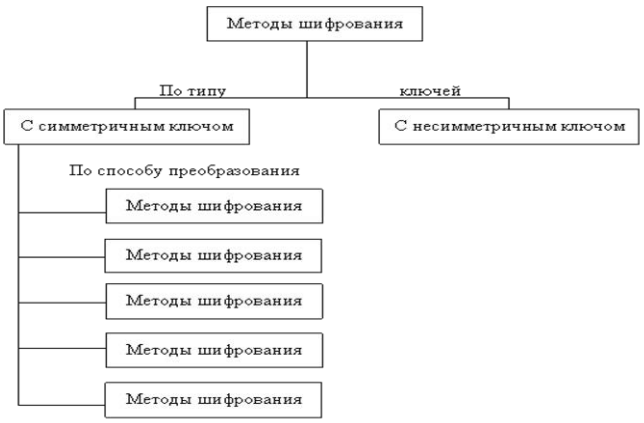


Рисунок 2 - Варианты классификации методов шифровании

2 Системы подстановок

Сущность методов замены (подстановки) заключается в замене символов исходной информации, записанных в одном алфавите, символами из другого алфавита по определенному правилу. Самым простым является *метод прямой замены*. Символам s_{0i} исходного алфавита A_0, с помощью которых записывается исходная информация, однозначно ставятся в соответствие символы s_{1i} шифрующего алфавита A_1. В простейшем случае оба алфавита могут состоять из одного и того же набора символов. Например, оба алфавита могут содержать буквы русского алфавита.

Задание соответствия между символами обоих алфавитов осуществляется с помощью преобразования числовых эквивалентов символов исходного текста T_0, длиной - К символов, по определенному алгоритму. Алгоритм моноалфавитной замены может быть представлен в виде последовательности шагов.

Шаг 1. Формирование числового кортежа L_{0h} путем замены каждого символа $s_{0i} \in T_0 (i = \overline{1,K})$, представленного в исходном алфавите A_0 размера [1xK], на число $h_{0i}(s_{0i})$, соответствующее порядковому номеру символа s_{0i} в алфавите A_0.

Шаг 2. Формирование числового кортежа L_{1h} путем замены каждого числа кортежа L_{0h} на соответствующее число h_{1i} кортежа L_{1h}, вычисляемое по формуле:

$$h_{li}=k_1*h_{0i}(s_{0i})+k_2)(\text{mod } R),$$

где k_1 - десятичный коэффициент; k_2 - коэффициент сдвига. Выбранные коэффициенты k_1, k_2 должны обеспечивать однозначное соответствие чисел h_{0i} и h_{1i}, а при получении $h_{1i}=0$ выполнить замену $h_{1i}=R$.

Шаг 3. Получение шифртекста T_1 путем замены каждого числа $h_{li}(s_{li})$ кортежа L_{1h} соответствующим символом $s_{1i} \in T_1 (i = \overline{1,K})$ алфавита шифрования A_1размера [1xR].

Шаг 4. Полученный шифртекст разбивается на блоки фиксированной длины b. Если последний блок оказывается неполным, то в конец блока помещаются специальные символы-заполнители (например, символ *).

Пример. Исходными данными для шифрования являются:$_0$=<М Е Т О Д _ Ш И Ф Р О В А Н И Я>;$_0$=<А Б В Г Д Е Ж З И К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я _>;$_1$=<О Р Щ Ь Я Т Э _ Ж М Ч Х А В Д Ы Ф К С Е З П И Ц Г Н Л Ъ Ш Б У Ю>;=32; k_1=3; k_2=15; b=4.

Пошаговое выполнение алгоритма приводит к получению следующих результатов.

Шаг 1. L_{0h}=<12, 6, 18, 14, 5, 32, 24, 9, 20, 16, 14, 3, 1, 13, 9, 31>.

Шаг 2. L_{1h}=<19, 1, 5, 25, 30, 15, 23, 10, 11, 31, 25, 24, 18, 22, 10, 12>.

Шаг 3. T_1=<С О Я Г Б Д И М Ч У Г Ц К П М Х>.

Шаг 4. T_2=<С О Я Г Б Д И М Ч У Г Ц К П М Х>.

При расшифровании сначала устраняется разбиение на блоки. Получается непрерывный шифртекст T_1 длиной K символов. Расшифрование осуществляется путем решения целочисленного уравнения:

$$k_1h_{01}+k_2=nR+h_{1i},$$

При известных целых величинах k_1, k_2, h_{1i} и R величина h_{0i} вычисляется методом перебора n.

Последовательное применение этой процедуры ко всем символам шифртекста приводит к его расшифрованию.

По условиям приведенного примера может быть построена таблица замены, в которой взаимозаменяемые символы располагаются в одном столбце (табл. 2.3.1.1).

Таблица замены																														
Таблица 1 s_{0i}	А	Б	В	Г	Д	Е	Ж	З	И	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Ъ	Ы	Ь	Э	Ю	Я	_
h_{0i}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	25	26	27	28	29	30	31	32
s_{1i}	К	З	Ц	Л	Б	О	Ь	Э	М	А	Ы	С	П	Г	Ъ	У	Р	Я	_	Ч	В	Ф	Е	Н	Ш	Ю	Щ	Т	Ж	Х	Д
h_{1i}	18	21	24	27	30	1	4	7	10	13	16	19	22	25	28	31	2	5	8	11	14	17	20	26	29	32	3	6	9	12	15

Использование таблицы замены значительно упрощает процесс шифрования. При шифровании символ исходного текста сравнивается с символами строки s_{0i} таблицы. Если произошло совпадение в i-м столбце, то символ исходного текста заменяется символом из строки s_{1i}, находящегося в том же столбце i таблицы. Расшифрование осуществляется аналогичным образом, но вход в таблицу производится по строке s_{1i}.

Основным недостатком метода прямой замены является наличие одних и тех же статистических характеристик исходного и закрытого текста. Зная, на каком языке написан исходный текст и частотную характеристику употребления символов алфавита этого языка, криптоаналитик путем статистической обработки перехваченных сообщений может установить соответствие между символами обоих алфавитов.

Существенно более стойкими являются методы полиалфавитной замены. Такие методы основаны на использовании нескольких алфавитов для замены символов исходного текста. Формально полиалфавитную замену можно представить следующим образом. При N - алфавитной замене символ s_{01} из исходного алфавита A_0 заменяется символом s_{11} из алфавита A_1, s_{02} заменяется символом s_{22} из алфавита A_2 и так далее. После замены s_{0N}символом s_{NN} из A_N символ $s_{0(N+1)}$ замещается символом $s_{1(N+1)}$ из алфавита A_1 и так далее.

Наибольшее распространение получил алгоритм полиалфавитной замены с использованием таблицы (матрицы) Вижинера T_B, которая представляет собой квадратную матрицу [RxR], где R - количество символов в используемом алфавите. В первой строке располагаются символы в алфавитном порядке. Начиная со второй строки, символы записываются со сдвигом влево на одну позицию. Выталкиваемые символы заполняют освобождающиеся позиции справа (циклический сдвиг). Если используется русский алфавит, то матрица Вижинера имеет размерность [32 x 32] (рис. 3).

$$T_B = \begin{vmatrix} А & Б & В & Г & Д & \cdots & Ъ & Э & Ю & Я & \_ \\ Б & В & Г & Д & Е & \cdots & Э & Ю & Я & \_ & А \\ В & Г & Д & Е & Ж & \cdots & Ю & Я & \_ & А & Б \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \_ & А & Б & В & Г & \cdots & Ы & Ъ & Э & Ю & Я \end{vmatrix}$$

Рисунок 3 - Матрица Вижинера

Шифрование осуществляется с помощью ключа, состоящего из М неповторяющихся символов. Из полной матрицы Вижинера выделяется матрица шифрования $T_ш$, размерностью [(М+1), R]. Она включает первую строку и строки, первые элементы которых совпадают с символами ключа. Если в качестве ключа выбрано слово <ЗОНД>, то матрица шифрования содержит пять строк (рис. 4).

$$T_ш = \begin{vmatrix} А\ Б\ В\ Г\ Д\ Е\ Ж\ З\ И\ К\ Л\ М\ Н\ О\ П\ Р\ С\ Т\ У\ Ф\ Х\ Ц\ Ч\ Ш\ Щ\ Ъ\ Ы\ Ь\ Э\ Ю\ Я\ \_ \\ З\ И\ К\ Л\ М\ Н\ О\ П\ Р\ С\ Т\ У\ Ф\ Х\ Ц\ Ч\ Ш\ Щ\ Ъ\ Ы\ Ь\ Э\ Ю\ Я\ \_\ А\ Б\ В\ Г\ Д\ Е\ Ж \\ О\ П\ Р\ С\ Т\ У\ Ф\ Х\ Ц\ Ч\ Ш\ Щ\ Ъ\ Ы\ Ь\ Э\ Ю\ Я\ \_\ А\ Б\ В\ Г\ Д\ Е\ Ж\ З\ И\ К\ Л\ М\ Н \\ Н\ О\ П\ Р\ С\ Т\ У\ Ф\ Х\ Ц\ Ч\ Ш\ Щ\ Ъ\ Ы\ Ь\ Э\ Ю\ Я\ \_\ А\ Б\ В\ Г\ Д\ Е\ Ж\ З\ И\ К\ Л\ М \\ Д\ Е\ Ж\ З\ И\ К\ Л\ М\ Н\ О\ П\ Р\ С\ Т\ У\ Ф\ Х\ Ц\ Ч\ Ш\ Щ\ Ъ\ Ы\ Ь\ Э\ Ю\ Я\ \_\ А\ Б\ В\ Г \end{vmatrix}$$

Рисунок 4 - Матрица шифрования для ключа <ЗОНД>

Алгоритм зашифрования с помощью таблицы Вижинера представляет собой следующую последовательность шагов.

Шаг 1. Выбор ключа К длиной М символов.

Шаг 2. Построение матрицы шифрования $T_ш=(b_{ij})$ размерностью [(М+1), R] для выбранного ключа К.

Шаг 3. Под каждым символом s_{0r} исходного текста длиной I символов размещается символ ключа k_m. Ключ повторяется необходимое число раз.

Шаг 4. Символы исходного текста последовательно замещаются символами, выбираемыми из $T_ш$ по следующему правилу:

1. определяется символ km ключа К, соответствующий замещаемому символу sor;

2. находится строка i в Тш, для которой выполняется условие km=bi1;

. определяется столбец j, для которого выполняется условие: sor=b1j;

. символ sor замещается символом bij.

Шаг 5. Полученная зашифрованная последовательность разбивается на блоки определенной длины, например, по четыре символа. Последний блок дополняется, при необходимости, служебными символами до полного объема.

Расшифрование осуществляется в следующей последовательности:

Шаг 1. Под шифртекстом записывается последовательность символов ключа по аналогии с шагом 3 алгоритма зашифрования.

Шаг 2. Последовательно выбираются символы s_{1r} из шифртекста и соответствующие символы ключа K_m. В матрице $T_ш$ определяется строка i, для которой выполняется условие $K_m = b_{i1}$. В строке 1 определяется элемент $b_{ij} = s_{1r}$. В расшифрованный текст на позицию r помещается символ b_{1j}.

Шаг 3. Расшифрованный текст записывается без разделения на блоки. Убираются служебные символы.

3 Подстановка Цезаря

Подстановка Цезаря является самым простым вариантом подстановки. Она относится к группе *моноалфавитных подстановок*.

Например, ВЫШЛИТЕ_НОВЫЕ_УКАЗАНИЯ посредством подстановки C_3 преобразуется в еюыолхиврсеюивцнгкгрл

Таблица 1

А→г	Й→м	Т→х	Ы→ю
Б→д	К→н	У→ц	Ь→я
В→е	Л→о	Ф→ч	Э→_
Г→ж	М→п	Х→ш	Ю→а
Д→з	Н→р	Ц→щ	Я→б
Е→и	О→с	Ч→ъ	_→в
Ж→й	П→т	Ш→ы	
З→к	Р→у	Щ→ь	
И→л	С→ф	Ъ→э	

При своей несложности система легко уязвима. Если злоумышленник имеет:

1) шифрованный и соответствующий исходный текст;

2) шифрованный текст выбранного злоумышленником исходного текста, то определение ключа и дешифрование исходного текста тривиальны.

Более эффективны обобщения подстановки Цезаря - *шифр Хилла* и *шифр Плэйфера*. Они основаны на подстановке не отдельных символов, а 2-грамм (шифр Плэйфера) или *n*-грамм (шифр Хилла). При более высокой криптостойкости они значительно сложнее для реализации и требуют достаточно большого количества ключевой информации.

4 Многоалфавитные системы. Системы одноразового использования

Слабая криптостойкость моноалфавитных подстановок преодолевается с применением подстановок многоалфавитных.

Многоалфавитная подстановка определяется ключом К=($К_1$, $К_2$, ...), содержащим не менее двух различных подстановок. В начале рассмотрим многоалфавитные системы подстановок с нулевым начальным смещением.

Для такой системы подстановки используют также термин “одноразовая лента” и “одноразовый блокнот”. Пространство ключей К системы одноразовой подстановки является вектором рангов (K_0, K_1, ..., K_{n-1}) и содержит m^n точек.

Рассмотрим небольшой пример шифрования с бесконечным ключом. В качестве ключа примем текст

“БЕСКОНЕЧНЫЙ_КЛЮЧ....”.

Зашифруем с его помощью текст “ШИФР_НЕРАСКРЫВАЕМ”. Шифрование оформим в таблицу:

ШИФРУЕМЫЙ_ТЕКСТ	24	8	20	16	19	5	12	27	9	32	18	5	10	17	18
БЕСКОНЕЧНЫЙ_КЛЮЧ	1	5	17	10	14	13	5	23	13	27	9	32	10	11	30
ЩРДЪАТТССЦЪЫДФЫП	25	13	4	26	0	18	17	17	22	26	27	4	20	28	15

Исходный текст невозможно восстановить без ключа.

Наложение белого шума в виде бесконечного ключа на исходный текст меняет статистические характеристики языка источника. Системы одноразового использования *теоретически не расшифруемы*, так как не содержат достаточной информации для восстановления текста.

Почему же эти системы неприменимы для обеспечения секретности при обработке информации? Ответ простой - они непрактичны, так как требуют независимого выбора значения ключа для каждой буквы исходного текста. Хотя такое требование может быть и не слишком трудным при передаче по прямому кабелю Москва - Нью-Йорк, но для информационных оно непосильно, поскольку там придется шифровать многие миллионы знаков.

Посмотрим, что получится, если ослабить требование шифровать каждую букву исходного текста отдельным значением ключа.

5 Системы шифрования Вижинера

Начнем с конечной последовательности ключа

$k = (k_0, k_1, ..., k_n)$,

которая называется *ключом пользователя*, и продлим ее до бесконечной последовательности, повторяя цепочку. Таким образом, получим *рабочий ключ*

$k = (k_0, k_1, ..., k_n)$, $k_j = k_{(j \bmod r)}$, $0 \square j < \square$.

Например, при $r = \square$ и ключе пользователя 15 8 2 10 11 4 18 рабочий ключ будет периодической последовательностью:

15 8 2 10 11 4 18 15 8 2 10 11 4 18 15 8 2 10 11 4 18 ...

Определение. Подстановка Вижинера VIG_k определяется как

$\mathrm{VIG}_k : (x_0, x_1, ..., x_{n-1}) \square (y_0, y_1, ..., y_{n-1}) = (x_0+k, x_1+k, .., x_{n-1}+k)$.

Таким образом:

1) исходный текст x делится на r фрагментов

$x_i = (x_i, x_{i+r}, ..., x_{i+r(n-1)})$, $0 \square i < r$;

2) i-й фрагмент исходного текста x_i шифруется при помощи подстановки Цезаря C_k :

$(x_i, x_{i+r}, ..., x_{i+r(n-1)}) \square (y_i, y_{i+r}, ..., y_{i+r(n-1)})$.

Вариант системы подстановок Вижинера при m=2 называется *системой Вернама (1917 г)*.

В то время ключ $k=(k_0, k_1, ..., k_{к-1})$ записывался на бумажной ленте. Каждая буква исходного текста в алфавите, расширенном некоторыми дополнительными знаками, сначала переводилась с использованием *кода Бодо* в пятибитовый символ. К исходному тексту Бодо добавлялся ключ (по модулю 2). Старинный телетайп фирмы AT&T со считывающим устройством Вернама и оборудованием для шифрования, использовался корпусом связи армии США.

Очень распространена плохая, с точки зрения секретности, *практика использовать слово или фразу в качестве ключа* для того, чтобы $k=(k_0, k_1, ..., k_{к-1})$ было легко запомнить. В ИС для обеспечения безопасности информации это

недопустимо. Для получения ключей должны использоваться программные или аппаратные средства случайной генерации ключей.

Пример. Преобразование текста с помощью подстановки Вижинера (r=4).

Исходный текст (ИТ1):

НЕ_СЛЕДУЕТ_ВЫБИРАТЬ_НЕСЛУЧАЙНЫЙ_КЛЮЧ

Ключ: КЛЮЧ

Разобьем исходный текст на блоки по 4 символа:

НЕ_С ЛЕДУ ЕТ_В ЫБИР АТЬ_ НЕСЛ УЧАЙ НЫЙ_ КЛЮЧ

и наложим на них ключ (используя таблицу Вижинера):

Н+К=Ч, Е+Л=Р и т.д.

Получаем зашифрованный (ЗТ1) текст:

ЧРЭЗ ХРБЙ ПЭЭЩ ДМЕЖ КЭЩЦ ЧРОБ ЭБЮ_ ЧЕЖЦ ФЦЫН

Следует признать, что и многоалфавитные подстановки в принципе доступны криптоаналитическому исследованию. Криптостойкость многоалфавитных систем резко убывает с уменьшением длины ключа.

Тем не менее такая система как шифр Вижинера допускает несложную аппаратную или программную реализацию и при достаточно большой длине ключа может быть использован в современных ИС.

Контрольные вопросы

1. Что собой представляет криптология? Какие основные направления она включает?
2. Дайте определения шифру, ключу, криптосистеме.
3. Сформулируйте основную задачу криптографии.
4. Что собой представляют шифры подстановки (замены)? Приведите примеры таких шифров.
5. Что собой представляют шифры перестановки? Приведите примеры таких шифров.
6. Как осуществляется шифрование методом гаммирования?
7. Определите основные характеристики криптографических средств защиты.

Лекция 9 Основные понятия, относящиеся к обеспечению целостности сообщений с помощью MAC и хэш-функций; простые хэш-функции и хэш-функция MD5

План изложения

1. Основные понятия

2. MAC и хэш-функции
3. Простые хэш-функции
4. Хэш-функция MD5

1 **Основные понятия**

Хэш-функцией называется односторонняя функция, предназначенная для получения *дайджеста* или "отпечатков пальцев" файла, сообщения или некоторого блока данных.

Хэш-код создается функцией H: h = H (M)

Где M является сообщением произвольной длины и h является *хэш-кодом* фиксированной длины.

Хэш-функция H, которая используется для аутентификации сообщений, должна обладать следующими свойствами:

1. *Хэш-функция* H должна применяться к блоку данных любой длины.
2. *Хэш-функция* H создает выход фиксированной длины.
3. H (M) относительно легко (за полиномиальное время) вычисляется для любого значения M.
4. Для любого данного значения *хэш-кода* h вычислительно невозможно найти M такое, что H (M) = h.
5. Для любого данного x вычислительно невозможно найти y ≠x, что H (y) = H (x).
6. Вычислительно невозможно найти произвольную пару (x, y) такую, что H (y) = H (x).

Первые три свойства требуют, чтобы *хэш-функция* создавала *хэш-код* для любого сообщения.

Четвертое свойство определяет требование односторонности *хэш-функции*: легко создать *хэш-код* по данному сообщению, но невозможно восстановить сообщение по данному *хэш-коду*. Это свойство важно, если аутентификация с использованием *хэш-функции* включает секретное значение. Само секретное значение может не посылаться, тем не менее, если *хэш-функция* не является односторонней, противник может легко раскрыть секретное значение.

Пятое свойство гарантирует, что невозможно найти другое сообщение, чье значение *хэш-функции* совпадало бы со значением *хэш-функции* данного сообщения. Это предотвращает подделку аутентификатора при использовании зашифрованного *хэш-кода*. В данном случае противник может читать сообщение и, следовательно, создать его *хэш-код*. Но так как противник не владеет секретным ключом, он не имеет возможности изменить сообщение так, чтобы получатель этого не обнаружил . Если данное свойство не выполняется, атакующий имеет возможность выполнить следующую последовательность действий: перехватить сообщение и его зашифрованный *хэш-код*, вычислить *хэш-код* сообщения, создать альтернативное сообщение с тем же самым *хэш-кодом*, заменить исходное сообщение на поддельное. Поскольку *хэш-коды* этих сообщений совпадают, получатель не обнаружит подмены.

Хэш-функция, которая удовлетворяет первым пяти свойствам, называется ***простой или слабой хэш-функцией***. Если кроме того выполняется шестое свойство, то такая функция называется ***сильной хэш-функцией***. Шестое свойство защищает против класса атак, известных как атака "*день рождения*".

1 MAC и хэш-функции

В данном случае под *MAC* (**Message Authentication Code**) понимается некоторый аутентификатор, являющийся определенным способом вычисленным блоком данных, с помощью которого можно проверить целостность сообщения. В некоторой степени симметричное шифрование всего сообщения может выполнять функцию аутентификации этого сообщения. Но в таком случае сообщение должно содержать достаточную избыточность, которая позволяла бы проверить, что сообщение не было изменено. Избыточность может быть в виде определенным образом отформатированного сообщения, текста на конкретном языке и т.п. Если сообщение допускает произвольную последовательность битов (например, зашифрован ключ сессии), то симметричное шифрование всего сообщения не может обеспечивать его целостность, так как при дешифровании в любом случае получится последовательность битов, правильность которой проверить нельзя.

Поэтому гораздо чаще используется критографически созданный небольшой блок данных фиксированного размера, так называемый аутентификатор или имитовставка, с помощью которого проверяется целостность сообщения. Этот блок данных может создаваться с помощью секретного ключа, который разделяют отправитель и получатель. *MAC* вычисляется в тот момент, когда известно, что сообщение корректно. После этого *MAC* присоединяется к сообщению и передается вместе с ним получателю. Получатель вычисляет *MAC*, используя тот же самый секретный ключ, и сравнивает вычисленное значение с полученным. Если эти значения совпадают, то с большой долей вероятности можно считать, что при пересылке изменения сообщения не произошло.

MAC на основе алгоритма симметричного шифрования

Для вычисления *MAC* может использоваться алгоритм симметричного шифрования (например, DES) в режиме CBC и нулевой инициализирующий вектор. В этом случае сообщение представляется в виде последовательности блоков, длина которых равна длине блока алгоритма шифрования. При необходимости последний блок дополняется справа нулями, чтобы получился блок нужной длины.

MAC на основе хэш-функции

Другим способом обеспечения целостности является использование хэш-функции. Хэш-код присоединяется к сообщению в тот момент, когда известно, что сообщение корректно. Получатель проверяет целостность сообщения вычислением хэш-кода полученного сообщения и сравнением его с полученным хэш-кодом, который должен быть передан безопасным способом. Одним из таких безопасных способов может быть шифрование хэш-кода закрытым ключом отправителя, т.е. создание подписи. Возможно также шифрование полученного хэш-кода алгоритмом симметричного шифрования, если отправитель и получатель имеют общий ключ симметричного шифрования.

HMAC

Еще один вариант использования хэш-функции для получения *MAC* состоит в том, чтобы определенным образом добавить секретное значение к сообщению, которое подается на вход хэш-функции. Такой алгоритм носит название ***HMAC***, и он описан в RFC 2104.

В алгоритме *HMAC* хэш-функция представляет собой "черный ящик". Это, во-первых, позволяет использовать существующие реализации хэш-функций, а во-вторых, обеспечивает легкую замену существующей хэш-функции на новую.

2 Простые хэш-функции

Все *хэш-функции* выполняются следующим образом. Входное значение (сообщение, файл и т.п.) рассматривается как последовательность n-битных блоков. Входное значение обрабатывается последовательно блок за блоком, и создается n-битное значение *хэш-кода*.

Одним из простейших примеров *хэш-функции* является побитный XOR каждого блока:

$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{ik}$

Где C_i - i-ый бит *хэш-кода,* $1 \le i \le n$,

k - число n-битных блоков входа.

b_{ij} - i-ый бит в j-ом блоке.

$\oplus$- операция XOR.

В результате получается *хэш-код* длины n, известный как продольный избыточный контроль. Это эффективно при случайных сбоях для проверки целостности данных.

Часто при использовании подобного продольного избыточного контроля для каждого блока выполняется однобитный циклический сдвиг после

вычисления *хэш-кода*. Это даст эффект "случайности" входа и уничтожит любую регулярность, которая присутствует во входных значениях.

Хотя второй вариант считается более предпочтительным для обеспечения целостности данных и предохранения от случайных сбоев, он не может использоваться для обнаружения преднамеренных модификаций передаваемых сообщений. Зная сообщение, атакующий легко может создать новое сообщение, которое имеет тот же самый *хэш-код*. Для этого следует подготовить альтернативное сообщение и затем присоединить n-битный блок, который является *хэш-кодом* нового сообщения, и блок, который является *хэш-кодом* старого сообщения.

3 Хэш-функция MD5

Рассмотрим алгоритм получения *дайджеста сообщения MD5* (RFC 1321), разработанный Роном Ривестом из MIT. Алгоритм получает на входе сообщение произвольной длины и создает в качестве выхода *дайджест сообщения* длиной 128 бит.

Рисунок 1 - Логика выполнения MD5

ABCD – инициализирующий вектор, состоящий из 4 подвекторов длиной 8 шестнадцатиричных цифр (4 байта). Y_i – i-ый блок исходного текста, HMD5 – модуль, состоящий из 4 циклических обработок.

Алгоритм MD4 является более ранней разработкой того же автора Рона Ривеста. Первоначально данный алгоритм был опубликован в октябре 1990 г., незначительно измененная версия была опубликована в RFC 1320 в апреле 1992 г. *MD5* является более сложным и, следовательно, более медленным при выполнении, чем MD4. Считается, что добавление сложности оправдывается возрастанием уровня безопасности

Контрольные вопросы

1. Дайте определение хеш-функции?

2. Назовите различие между функцией аутентификации пользователя и функцией аутентификации сообщений?

3. Что такое криптоанализ?

4. На какие классы можно разделить хеш-функции?

5. Что такое криптография?

Лекция 10 Блочные и поточные алгоритмы симметричного шифрования. Стандарты и алгоритмы: американский DES, отечественный ГОСТ, режимы их выполнения. Основные понятия, относящиеся к алгоритмам симметричного шифрования. Определение устойчивости алгоритма. Сеть Фейштеля

План изложения

1. Эволюция создания блочных шифров.
2. Общие сведения о блочных шифрах.
3. Сеть Фейштеля (Фейстеля–в разных источниках по-разному)
4. Блочный шифр TEA
5. Контрольные вопросы

1. Эволюция создания блочных шифров

Ныне действующий в России стандарт шифрования, ГОСТ 28147-89, был принят в 1989 году. Однако ряд фактов свидетельствуют о том, что разработан он был как минимум на десять лет раньше. По своей архитектуре ГОСТ 28147-89 достаточно близок к прежнему стандарту шифрования США - DES (Data Encryption Standard), который был разработан примерно в то же самое время и действовал в период с 1977 по 2001 год.

В 1970-е годы основной доминантой при проектировании шифров, в том числе ГОСТа и DES, было обеспечение приемлемой стойкости при жестких требованиях к сложности реализации, т.е. минимизации необходимых для реализации ресурсов. Это было обусловлено ограничениями электронных компонентов вычислительных систем, разработанных к тому времени. Важнейшими отличиями ГОСТа от DES были более простая функция шифрования и гораздо больший размер ключа.

Блочные шифры шифруют целые блоки информации (от 4 до 32 байт) как единое целое – это значительно увеличивает стойкость преобразований к атаке полным перебором и позволяет использовать различные математические и алгоритмические преобразования.

Практически все современные блочные шифры являются *композиционными* - т.е состоят из композиции простых преобразований или $\mathbf{F=F_1 \square F_2 o F_3 o F_4 o .. o F_n}$, где **F**-преобразование шифра, $\mathbf{F_i}$-простое преобразование, называемое также i-ым *циклом шифрования (раундом)*. Само по себе преобразование может и не обеспечивать нужных свойств, но их цепочка

позволяет получить необходимый результат. Например, стандарт DES состоит из 16 циклов. В иностранной литературе такие шифры часто называют послойными (layered). Если же используется одно и то же преобразование, т.е. $\mathbf{F_i}$ постоянно для □i, то такой композиционный шифр называют *итерационным* шифром.

2. Общие сведения о блочных шифрах

В зависимости от характера воздействий, производимых над данными, блочные алгоритмы подразделяются на:

1. Перестановочные (transposition, permutation, P-блоки);

Блоки информации (байты, биты, более крупные единицы) не изменяются сами по себе, но изменяется их порядок следования, что делает информацию недоступной стороннему наблюдателю.

2. Подстановочные (замены) (substitution, S-блоки).

Сами блоки информации изменяются по законам криптоалгоритма. Подавляющее большинство современных алгоритмов принадлежит к этой группе.

Шифры перестановок переставляют элементы открытых данных (биты, буквы, символы) в некотором новом порядке. Различают шифры горизонтальной, вертикальной, двойной перестановки, решетки, лабиринты, лозунговые и др.

Шифры замены заменяют элементы открытых данных на другие элементы по определенному правилу. Различают шифры простой, сложной, парной замены, буквенно-слоговое шифрование и шифры колонной замены. Шифры замены делятся на две группы:

- моноалфавитные (код Цезаря);
- полиалфавитные (шифр Видженера, цилиндр Джефферсона, диск Уэтстоуна, Enigma).

В моноалфавитных шифрах замены буква исходного текста заменяется на другую, заранее определенную букву. Например, в коде Цезаря буква заменяется на букву, отстоящую от нее в латинском алфавите на некоторое число позиций. Очевидно, что такой шифр взламывается совсем просто. Нужно подсчитать, как часто встречаются буквы в зашифрованном тексте, и сопоставить результат с известной для каждого языка частотой встречаемости букв.

В полиалфавитных подстановках для замены некоторого символа исходного сообщения в каждом случае его появления последовательно используются различные символы из некоторого набора. Понятно, что этот набор не бесконечен, через какое-то количество символов его нужно использовать снова. В этом слабость чисто полиалфавитных шифров.

В современных криптографических системах, как правило, используют оба способа шифрования (замены и перестановки). Такой шифратор называют составным (product cipher). Он более стойкий, чем шифратор, использующий только замены или перестановки.

На сегодняшний день разработано достаточно много стойких блочных шифров. Практически все алгоритмы используют для преобразований определенный набор биективных (обратимых) математических преобразований.

Характерной особенностью блочных криптоалгоритмов является тот факт, что в ходе своей работы они производят преобразование блока входной информации фиксированной длины и получают результирующий блок того же объема, но недоступный для прочтения сторонним лицам, не владеющим ключом. Таким образом, схему работы блочного шифра можно описать функциями Z=EnCrypt(X, Key) (шифрование) и X=DeCrypt(Z, Key) (дешифрование).

Ключ Key является параметром блочного криптоалгоритма и представляет собой некоторый блок двоичной информации фиксированного размера. Исходный (X) и зашифрованный (Z) блоки данных также имеют фиксированную разрядность, равную между собой, но необязательно равную длине ключа.

Блочные шифры являются основой, на которой реализованы практически все криптосистемы. Методика создания цепочек из зашифрованных блочными алгоритмами байт позволяет шифровать ими пакеты информации неограниченной длины. Такое свойство блочных шифров, как быстрота работы, используется асимметричными криптоалгоритмами, медлительными по своей природе. Отсутствие статистической корреляции между битами выходного потока блочного шифра используется для вычисления контрольных сумм пакетов данных и в хешировании паролей.

Следующие разработки всемирно признаны стойкими алгоритмами и публикаций об универсальных методах их взлома в средствах массовой информации на момент создания материала не встречалось.

Название алгоритма	**Автор**	**Размер блока**	**Длина ключа**
IDEA	Xuejia Lia and James Massey	64 бита	128 бит
CAST128		64 бита	128 бит
BlowFish	Bruce Schneier	64 бита	128 – 448 бит
ГОСТ	НИИ ***	64 бита	256 бит
TwoFish	Bruce Schneier	128 бит	128 – 256 бит
MARS	Корпорация IBM	128 бит	128 – 1048 бит

Криптоалгоритм именуется идеально стойким, если прочесть зашифрованный блок данных можно только, перебрав все возможные ключи, до тех пор, пока сообщение не окажется осмысленным. Так как по теории вероятности искомый ключ будет найден с вероятностью 1/2 после перебора половины всех ключей, то на взлом идеально стойкого криптоалгоритма с ключом длины N потребуется в среднем 2^{N-1} проверок. Таким образом, в общем случае стойкость блочного шифра зависит только от длины ключа и возрастает экспоненциально с ее ростом. Даже предположив, что перебор ключей производится на специально созданной многопроцессорной системе, в которой

благодаря диагональному параллелизму на проверку 1 ключа уходит только 1 такт, то на взлом 128-битного ключа современной технике потребуется не менее 10^{21}лет. Естественно, все сказанное относится только к идеально стойким шифрам, которыми, например, с большой долей уверенности являются приведенные в таблице выше алгоритмы.

Кроме этого условия к идеально стойким криптоалгоритмам применяется еще одно очень важное требование, которому они должны обязательно соответствовать. При известных исходном и зашифрованном значениях блока ключ, которым произведено это преобразование, можно узнать также только полным перебором. Ситуации, в которых постороннему наблюдателю известна часть исходного текста, встречаются повсеместно. Это могут быть стандартные надписи в электронных бланках, фиксированные заголовки форматов файлов, довольно часто встречающиеся в тексте длинные слова или последовательности байт. В свете этой проблемы описанное выше требование не является ничем чрезмерным и также строго выполняется стойкими криптоалгоритмами, как и первое.

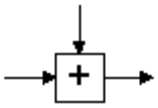
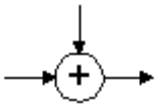
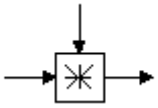
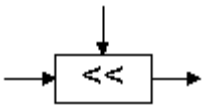
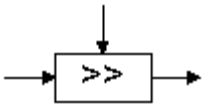
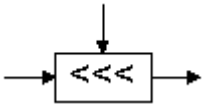
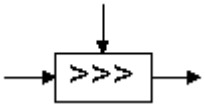
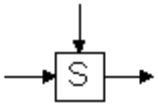
Таким образом, на функцию стойкого блочного шифра Z=EnCrypt(X, Key) накладываются следующие условия:

1. Функция EnCrypt должна быть обратимой.
2. Не должно существовать иных методов прочтения сообщения X по известному блоку Z, кроме как полным перебором ключей Key.
3. Не должно существовать иных методов определения, каким ключом Key было произведено преобразование известного сообщения X в сообщение Z, кроме как полным перебором ключей.

Давайте рассмотрим методы, с помощью которых разработчики блочных криптоалгоритмов добиваются одновременного выполнения этих трех условий с очень большой долей достоверности.

Все действия, производимые над данными блочным криптоалгоритмом, основаны на том факте, что преобразуемый блок может быть представлен в виде целого неотрицательного числа из диапазона, соответствующего его разрядности. Так, например, 32-битный блок данных можно интерпретировать как число из диапазона 0..4'294'967'295 (2^{32}). Кроме того, блок, разрядность которого обычно является "степенью двойки", можно трактовать как несколько независимых неотрицательных чисел из меньшего диапазона (рассмотренный выше 32-битный блок можно также представить в виде 2 независимых чисел из диапазона 0..65535 (2^{16}) или в виде 4 независимых чисел из диапазона 0..255).

Над этими числами блочным криптоалгоритмом и производятся по определенной схеме следующие действия (слева даны условные обозначения этих операций на графических схемах алгоритмов):

Биективные математические функции		
	Сложение	X'=X+V
	Исключающее ИЛИ	X'=X XOR V
	Умножение по модулю 2^N+1	X'=(X*V) mod (2^N+1)
	Умножение по модулю 2^N	X'=(X*V) mod (2^N)
Битовые сдвиги		
	Арифметический сдвиг влево	X'=X SHL V
	Арифметический сдвиг вправо	X'=X SHR V
	Циклический сдвиг влево	X'=X ROL V
	Циклический сдвиг вправо	X'=X ROR V
Табличные подстановки		
	S-box (англ. Substitute)	X'=Table[X,V]

Табличная подстановка S-box – действие, при котором группа битов отображается в другую группу битов – когда промежуточный результат модифицируется обычно с помощью таблицы (например, по каждым двум байтам промежуточного результата из таблицы выбирается соответствующее им новое значение. S-box'ы не должны быть похожи друг на друга. Например, количество входных значений, дающих одно и тоже значение на выходе из разных S-box'ов, должно быть минимально.

В качестве параметра V для любого из этих преобразований может использоваться:

1. фиксированное число (например, X'=X+125);
2. число, получаемое из ключа (например, X'=X+F(Key));
3. число, получаемое из независимой части блока (например, $X_2'=X_2+F(X_1)$).

Последний вариант используется в схеме, названной по имени ее создателя сетью Фейштеля (нем. Feistel).

Последовательность выполняемых над блоком операций, комбинации перечисленных выше вариантов V и сами функции F и составляют "ноу-хау" каждого конкретного блочного криптоалгоритма. Один-два раза в год исследовательские центры мира публикуют очередной блочный шифр, который под яростной атакой криптоаналитиков либо приобретает за несколько лет статус стойкого криптоалгоритма, либо (что происходит неизмеримо чаще) бесславно уходит в историю криптографии.

Характерным признаком блочных алгоритмов является многократное и косвенное использование материала ключа. Это диктуется в первую очередь требованием невозможности обратного декодирования в отношении ключа при известных исходном и зашифрованном текстах. Для решения этой задачи в приведенных выше преобразованиях чаще всего используется не само значение ключа или его части, а некоторая, иногда необратимая (небиективная) функция от материала ключа. Более того, в подобных преобразованиях один и тот же блок или элемент ключа используется многократно. Это позволяет при выполнении условия обратимости функции относительно величины X сделать функцию необратимой относительно ключа Key.

Поскольку операция зашифровки или расшифровки отдельного блока в процессе кодирования пакета информации выполняется многократно (иногда до сотен тысяч раз), а значение ключа и, следовательно, функций $V_i(Key)$ остается неизменным, то иногда становится целесообразно заранее однократно вычислить данные значения и хранить их в оперативной памяти совместно с ключом. Поскольку эти значения зависят только от ключа, то они в криптографии называются материалом ключа. Необходимо отметить, что данная операция никоим образом не изменяет ни длину ключа, ни криптостойкость алгоритма в целом. Здесь происходит лишь оптимизация скорости вычислений путем кеширования (англ. caching) промежуточных результатов. Описанные действия встречаются практически во многих блочных криптоалгоритмах и носят название расширение ключа (англ. key scheduling)

3. Сеть Фейштеля (Фейстеля–в разных источниках по-разному)

Сетью Фейштеля (петлей Фейстеля) называется метод обратимых преобразований текста, при котором значение, вычисленное от одной из частей текста, накладывается на другие части. Часто структура сети выполняется таким образом, что для шифрования и дешифрования используется один и тот же алгоритм – различие состоит только в порядке использования материала ключа.

Сеть Фейштеля является дальнейшей модификацией описанного выше метода смешивания текущей части шифруемого блока с результатом некоторой функции, вычисленной от другой независимой части того же блока. Эта методика получила широкое распространение, поскольку обеспечивает выполнение требования о многократном использовании ключа и материала исходного блока информации.

Классическая сеть Фейштеля имеет следующую структуру:

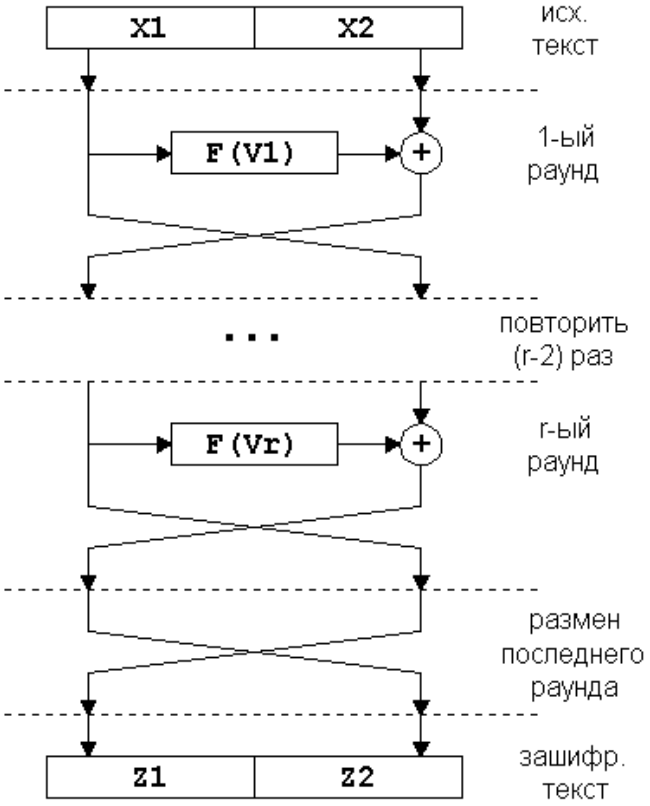


Рисунок 1 - Классическая сеть Фейштеля

Независимые потоки информации, порожденные из исходного блока, называются ветвями сети. В классической схеме их две. Величины V_i именуются параметрами сети, обычно это функции от материала ключа. Функция F называется образующей. Действие, состоящее из однократного вычисления образующей функции и последующего наложения ее результата на другую ветвь с обменом их местами, называется циклом или раундом (англ. round) сети Фейштеля. Оптимальное число раундов – от 8 до 32. Важно то, что увеличение количества раундов значительно увеличивает криптоскойстость любого блочного шифра к криптоанализу. Возможно, эта особенность и повлияла на столь активное распространение сети Фейштеля – ведь при обнаружении, скажем, какого-либо слабого места в алгоритме, почти всегда достаточно увеличить количество раундов на 4-8, не переписывая сам алгоритм. Часто количество раундов не фиксируется разработчиками алгоритма, а лишь указываются разумные пределы (обязательно нижний, и не всегда – верхний) этого параметра.

Ключ имеет фиксированную длину. Однако при прокрутке хотя бы 8 циклов шифрования с размером блока, скажем, 128 бит даже при простом прибавлении посредством XOR потребуется 8*128=1024 бита ключа, поскольку нельзя добавлять в каждом цикле одно и то же значение - это ослабляет шифр. Посему для получения последовательности ключевых бит придумывают специальный *алгоритм выработки цикловых ключей (ключевое расписание - key schedule)*. В результате работы этого алгоритма из исходных бит ключа шифрования получается массив бит определенной длины, из которого по

определенным правилам составляются цикловые ключи. Каждый шифр имеет свой алгоритм выработки цикловых ключей.

Сразу же возникает вопрос, – является ли данная схема обратимой? Очевидно, да. Сеть Фейштеля обладает тем свойством, что даже если в качестве образующей функции F будет использовано необратимое преобразование, то и в этом случае вся цепочка будет восстановима. Это происходит вследствие того, что для обратного преобразования сети Фейштеля не нужно вычислять функцию F^{-1}.

Более того, как нетрудно заметить, сеть Фейштеля симметрична. Использование операции XOR, обратимой своим же повтором, и инверсия последнего обмена ветвей делают возможным раскодирование блока той же сетью Фейштеля, но с инверсным порядком параметров V_i. Заметим, что для обратимости сети Фейштеля не имеет значение является ли число раундов четным или нечетным числом. В большинстве реализаций схемы, в которых оба вышеперечисленные условия (операция XOR и уничтожение последнего обмена) сохранены, прямое и обратное преобразования производятся одной и той же процедурой, которой в качестве параметра передается вектор величин V_i либо в исходном, либо в инверсном порядке.

С незначительными доработками сеть Фейштеля можно сделать и абсолютно симметричной, то есть выполняющей функции шифрования и дешифрования одним и тем же набором операций. Математическим языком это записывается как "Функция EnCrypt тождественно равна функции DeCrypt". Если мы рассмотрим граф состояний криптоалгоритма, на котором точками отмечены блоки входной и выходной информации, то при каком-то фиксированном ключе для классической сети Фейштеля мы будем иметь картину, изображенную на рис.2а, а во втором случае каждая пара точек получит уникальную связь, как изображено на рис. 2б. Модификация сети Фейштеля, обладающая подобными свойствами приведена на рисунке 3. Как видим, основная ее хитрость в повторном использовании данных ключа в обратном порядке во второй половине цикла. Необходимо заметить, однако, что именно из-за этой недостаточно исследованной специфики такой схемы (то есть потенциальной возможности ослабления зашифрованного текста обратными преобразованиями) ее используют в криптоалгоритмах с большой осторожностью.

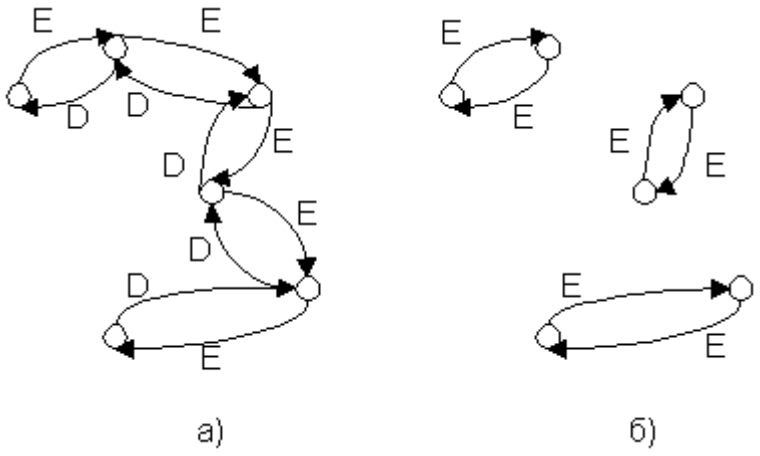


Рисунок 2 - Сеть Фейштеля с фиксированным ключом

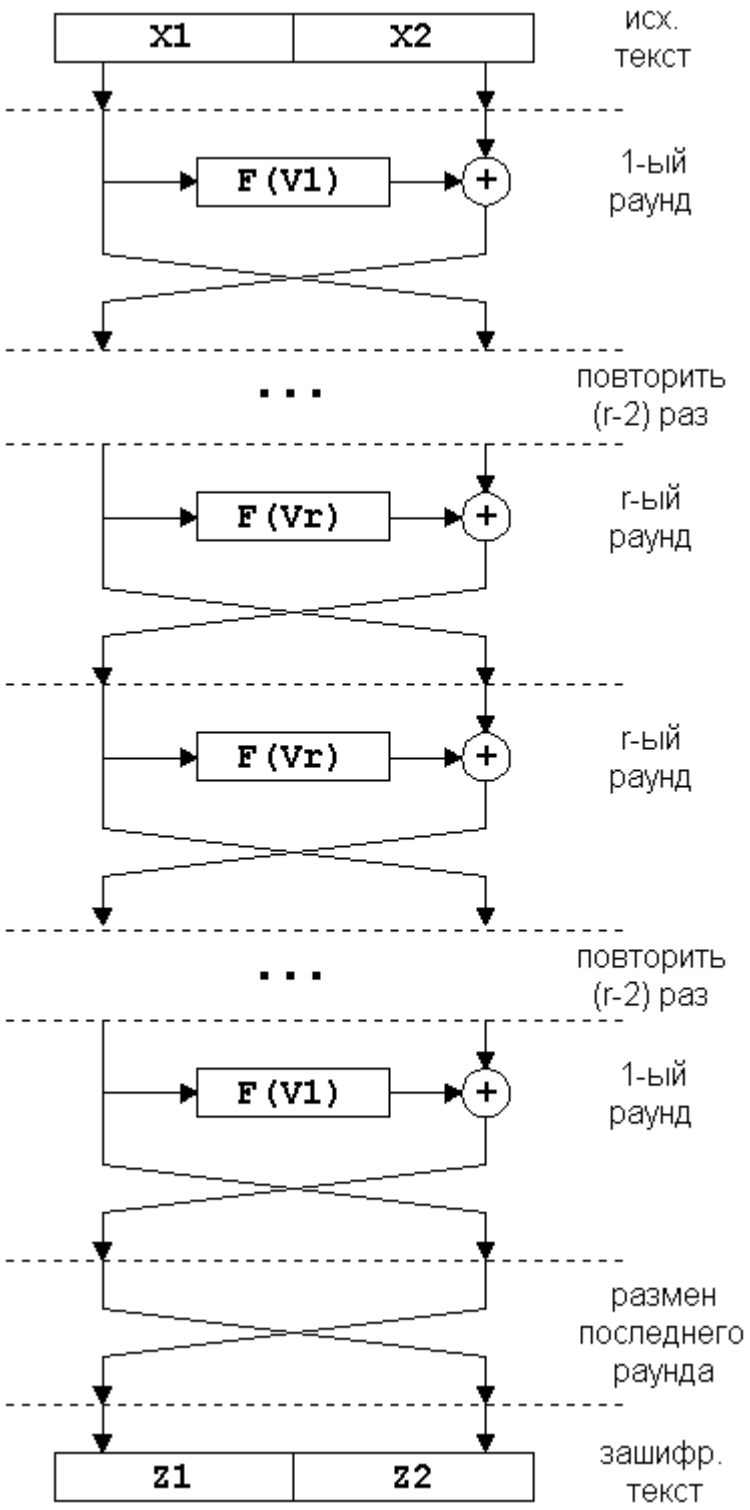


Рисунок 3 - Модификация сети Фейштеля

А вот модификацию сети Фейштеля для большего числа ветвей применяют гораздо чаще. Это в первую очередь связано с тем, что при больших размерах кодируемых блоков (128 и более бит) становится неудобно работать с математическими функциями по модулю 64 и выше. Как известно, основные единицы информации, обрабатываемые процессорами на сегодняшний день – это байт и двойное машинное слово 32 бита. Поэтому все чаще и чаще в блочных криптоалгоритмах встречается сеть Фейштеля с 4-мя ветвями. Самый простой принцип ее модификации изображен на рисунке 4а. Для более

быстрого перемешивания информации между ветвями (а это основная проблема сети Фейштеля с большим количеством ветвей) применяются две модифицированные схемы, называемые "type-2" и "type-3". Они изображены на рисунках 4б и 4в.

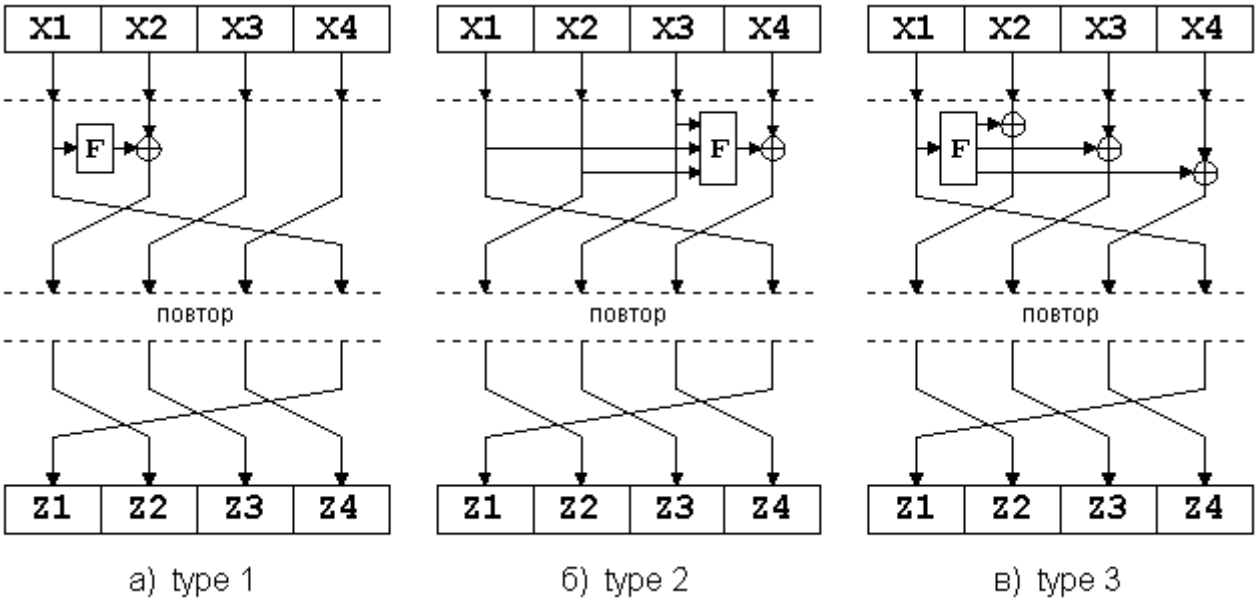


Рисунок 4 - Сеть Фейштеля с 4-мя ветвями

Сеть Фейштеля надежно зарекомендовала себя как криптостойкая схема произведения преобразований, и ее можно найти практически в любом современном блочном шифре. Незначительные модификации касаются обычно дополнительных начальных и оконечных преобразований (англоязычный термин – whitening) над шифруемым блоком. Подобные преобразования, выполняемые обычно также либо "исключающим ИЛИ" или сложением имеют целью повысить начальную рандомизацию входного текста. Таким образом, криптостойкость блочного шифра, использующего сеть Фейштеля, определяется на 95% функцией F и правилом вычисления V_i из ключа. Эти функции и являются объектом все новых и новых исследований специалистов в области криптографии.

4. Блочный шифр TEA

Блочный алгоритм TEA приведен как пример одного из самых простых в реализации стойких криптоалгоритмов.

Рассмотрим один из самых простых в реализации, но признанно стойких криптоалгоритмов – TEA (Tiny Encryption Algorithm).

Параметры алгоритма:

Размер блока – 64 бита;

Длина ключа – 128 бит.

В алгоритме использована сеть Фейштеля с двумя ветвями в 32 бита каждая. Образующая функция F обратима. Сеть Фейштеля несимметрична из-за

использования в качестве операции наложения не исключающего "ИЛИ", а арифметического сложения.

Ниже приведен код криптоалгоритма на языке программирования PASCAL.

```
type TLong2=array[0.. 1] of longint;
  TLong2x2=array[0.. 1] of TLong2;
const Delta=$9E3779B9;
var  key:TLong2x2;
procedure EnCryptRouting(var data);
var y,z,sum:longint; a:byte;
begin
y:=TLong2(data)[0];z:=TLong2(data)[1];sum:=0;
for a:=0 to 31 do
 begin
 inc(sum,Delta);
 inc(y,((z shl 4)+key[0,0]) xor (z+sum) xor ((z shr 5)+key[0,1]));
 inc(z,((y shl 4)+key[1,0]) xor (y+sum) xor ((y shr 5)+key[1,1]));
 end;
TLong2(data)[0]:=y;TLong2(data)[1]:=z
end;
```

Схема работы алгоритма приведена на рисунке 5.

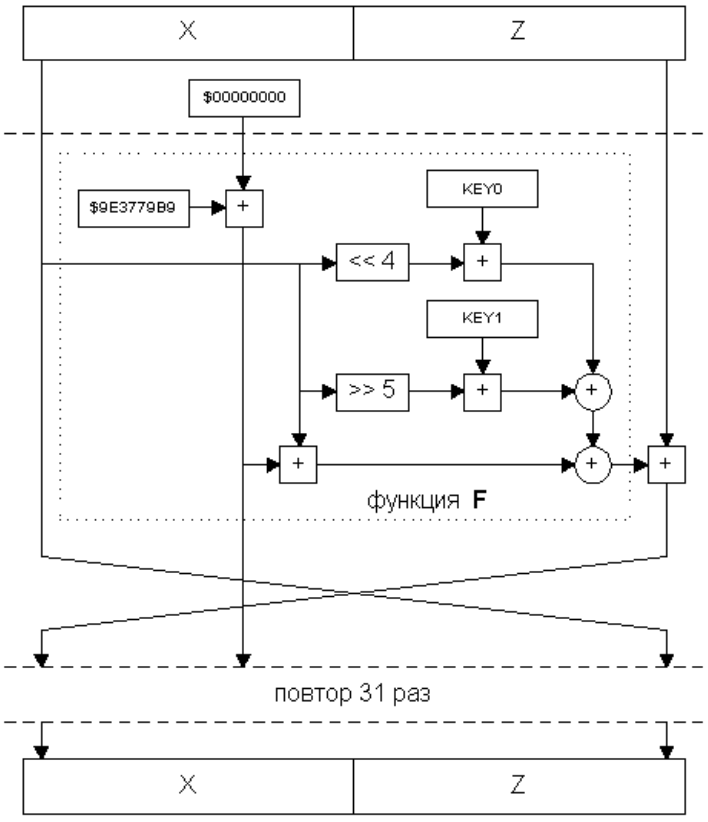


Рисунок 5 - Схема работы алгоритма

Отличительной чертой криптоалгоритма TEA является его размер. Простота операций, отсутствие табличных подстановок и оптимизация под 32-разрядную архитектуру процессоров позволяет реализовать его на языке ASSEMBLER в предельно малом объеме кода. Недостатком алгоритма является некоторая медлительность, вызванная необходимостью повторять цикл Фейштеля 32 раза (это необходимо для тщательного "перемешивания данных" из-за отсутствия табличных подстановок).

Аутентификация сообщений с помощью блочных шифров.

Итак, как же можно проверить подлинность сообщения с помощью блочного шифра? Довольно просто.

1.Отправитель А хочет отправить некое сообщение $(a_1,..,a_t)$. Он зашифровывает его на секретном ключе, который знает только он и получатель, а затем из получившегося шифротекста берет последний блок b_t из $к$ бит (при этом $к$ должно быть достаточно большим).

2.Отправитель А посылает сообщение $(a_1, .. , a_t , b_t)$ получателю в открытом виде или зашифровав его на другом ключе.

3.Получатель В, получив сообщение, $(a_1, .. , a_t, b_t)$, зашифровывает $(a_1,..,a_t)$ в том же режиме, что и А (должна быть договоренность) на том же секретном ключе (который знает только он и А).

4.Сравнивая полученный результат с b_t он удостоверяется, что сообщение отправил А, что оно не было подделано на узле связи (в случае передачи в открытом виде).

В данной схеме b_t является *кодом аутентификации сообщения (MAC)*. Для российского стандарта шифрования процесс получения кода аутентификации называется работой в режиме *имитовставки*.

Некоторые комментарии:

Режим шифрования должен быть обязательно с распространением ошибок ("распространение ошибки" - изменение блока открытого текста меняет все последующие блоки шифротекста – применяется в некоторых режимах шифрования). Необходимо использовать шифр с достаточной длиной блока, а то может появиться ситуация, когда из-за небольшого числа используемых бит для аутентификации возможно подменить исходное сообщение и при знании ключа получить тот же самый результат. Также очевидно, что схема опирается на то, что оба абонента имеют один и тот же секретный ключ, который получили заранее.

Итоги:

Для определения блочного шифра необходимо задать следующее:

1. числовые параметры алгоритма:
- размер шифруемого блока данных $|\boldsymbol{T}| = \boldsymbol{N}$;
- размер ключа $|\boldsymbol{K}| = \boldsymbol{NK}$;
- размер “шагового” ключа $|\boldsymbol{Ki}| = \boldsymbol{NK'}$;
- число шагов шифрования (раундов) $\boldsymbol{n}$;

2. функцию шифрования $\boldsymbol{f}$, принимающую в качестве аргумента и возвращающую в качестве значения $\boldsymbol{N}/2$-битовые блоки данных. Функция шифрования зависит также от $\boldsymbol{NK'}$ -битового секретного блока данных – “шагового” ключа;

3. набора функций φ(I), $1 \le i \le n$ для выработки “шаговых” ключей Ki из исходного ключа шифрования K: $Ki = \varphi i\ (K)$;

4. перестановки битов Y в N-битовом блоке данных.

Сейчас мы рассмотрим ГОСТ по намеченной выше схеме:

1. определим числовые характеристики алгоритма:

- размер шифруемого блока равен 64 битам: $|T| = 64$;
- размер ключа равен 256 битам $|K| = 256$;
- на каждом шаге алгоритма используется 32-битовый ключевой элемент: $|Ki| = 32$;
- число повторений основного шага (число раундов) $n = 32$;

2. функция шифрования определяется следующим образом:

- исходными данными для функции шифрования являются 32-битовые элементы данных T и “шаговый” ключ Ki;
- блок данных T и “шаговый” ключ Ki складываются по модулю 2^{32} :$S = (T + Ki) mod 2^{32}$;
- полученный 32-битовый блок данных интерпретируется как массив из восьми 4-битовых групп $S = (S_1,S_2,...,S_8)$, $|Si| = 4$, и в каждой группе выполняется подстановка с помощью соответствующего узла замен: $Si' = hi, Si$. В результате мы получаем следующий блок данных:

$S' = (S_1', S_2', ..., S_8') = (h_1,S_1,h_2,S_2,...,h_8,S_8)$;

- результат предыдущего шага вращается на 11 битов влево, то есть в сторону старших разрядов: если $S' = b31b30...b21b20...b1b0$, то $T' = R\_11(S') = b20...b1b0b31b30...b21$;
- полученный блок данных T' является значением функции шифрования: $T' = fi\ (T,K)$;

3. при шифровании используется следующая секретная (ключевая) информация:

- **ключ** – 256-битовый массив информации, структурированный в массив из восьми 32-битовых элементов, которые мы пронумеруем в порядке их использования в цикле шифрования:

$K = (k1,k2,...,k8)$, $|K| = 256$, $|ki| = 32$;

- **таблица замен** – набор из восьми – по числу битовых групп, на которые разбивается 32-битовый блок данных при вычислении функции шифрования – **узлов замен**: $H = (H1,H2,...,H8)$. Каждый **узел замен** представляет из себя подстановку в 16-элементном множестве всех возможных значений 4-битовых кодов, и таким образом может быть представлен в виде массива из 16 различных 4-битовых чисел: $Hi = (hi,0,hi,1,...,hi,15)$, $|hi,j| = 4$, $0 \le hi,j \le 15$, для любых i, j, k ($j \ne k$), должно выполняться условие $hi,j \ne hi,k$. Размер каждого узла замен равен $|Hi| = 8 \cdot |hi,j| = 64$ бита, а размер всей таблицы замен $|H| = 8 \cdot |Hi| = 8 \cdot 64 = 512$ бит или 64 байта;

4. определим порядок использования ключа при шифровании, то есть зададим функцию φ(i) выработки “шагового” ключа из исходного ключа шифрования K: $Ki = \varphi i\ (K)$. В качестве “шаговых” ключей Ki в ГОСТе берутся определенные элементы kj ключа K: $Ki = kj$. Однако таких элементов всего 8, а шагов в цикле шифрования 32, значит, необходимо задать функцию $\pi(i)$,

отображающую 32-элементное множество шагов в цикле шифрования в 8-элементное множество частей ключа: $\boldsymbol{Ki} = \boldsymbol{k}\pi(\boldsymbol{i})$, $1 \le \boldsymbol{i} \le 32$, $1 \le \pi(\boldsymbol{i}) \le 8$.

Зададим эту функцию в табличном и формульном виде:

$\boldsymbol{i}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi(\boldsymbol{i})$	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
$\boldsymbol{i}$	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
$\pi(\boldsymbol{i})$	1	2	3	4	5	6	7	8	8	7	6	5	4	3	2	1

$$\pi(i) = \begin{cases} i & 1 \le i \le 8 \\ i-8 & 9 \le i \le 16 \\ i-16 & 17 \le i \le 24 \\ 33-i & 25 \le i \le 32 \end{cases}$$

Иными словами, в ходе цикла зашифрования элементы ключа используются последовательно друг за другом, при этом ключ “просматривается” четыре раза – первые три в прямом направлении, четвертый – в обратном. Видно, что в цикле расшифрования элементы ключа используются в обратном по отношению к циклу зашифрования порядке, то есть сначала ключ просматривается один раз в прямом направлении, затем три раза в обратном.

Отметим, что основным секретным элементом российского шифра является ключ. Алгоритм должен оставаться криптостойким даже в случае раскрытия таблицы замен, что, однако, имеет место не при всех ее возможных значениях. Существование “слабых” таблиц замен, то есть таблиц, при использовании которых криптостойкость шифра существенно снижается, не вызывает сомнения – примером здесь может служить тривиальная таблица, при использовании которой всякое значение заменяется на него же самого.

Контрольные вопросы

1. Какие принципы блочного шифрования вы знаете?
2. В чем различие между блочными и потоковыми шифрами?
3. Какие требования выдвигаются к функции шифрования для блочных шифров?
4. На чем основана криптостойкость блочных шифров?
5. Какие алгоритмы блочного шифрования вам известны?
6. Опишите структуру шифра Файстеля.
7. Дайте определение понятиям диффузия и конфузия.
8. Какие параметры шифра Файстеля вам известны?
9. Какова размерность ключа и блока данных для алгоритма шифрования DES?
10. Опишите структуру раунда шифрования DES.
11. Опишите алгоритм формирования подключей DES.
12. Охарактеризуйте надежность DES.
13. Перечислите принципы, положенные в основу IDEA.

14. Каковы основные параметры алгоритма шифрования IDEA?
15. Опишите алгоритм развертывания ключей для RC5.
16. какие режимы блочного шифрования вам известны? Кратко охарактеризуйте каждый.

Лекция 11 Асимметричные системы шифрования (системы с открытым ключом). RSA. Функции дискретного логарифмирования и основанные на ней алгоритмы: схема Диффи-Хеллмана, схема Эль-Гамаля. Схема RSA: алгоритм шифрования, его оборачиваемость, вопросы устойчивости.

План изложения

1. Асимметричные системы шифрования (системы с открытым ключом).
2. Схема RSA
3. Функции дискретного логарифмирования
4. Схема Эль-Гамаля.

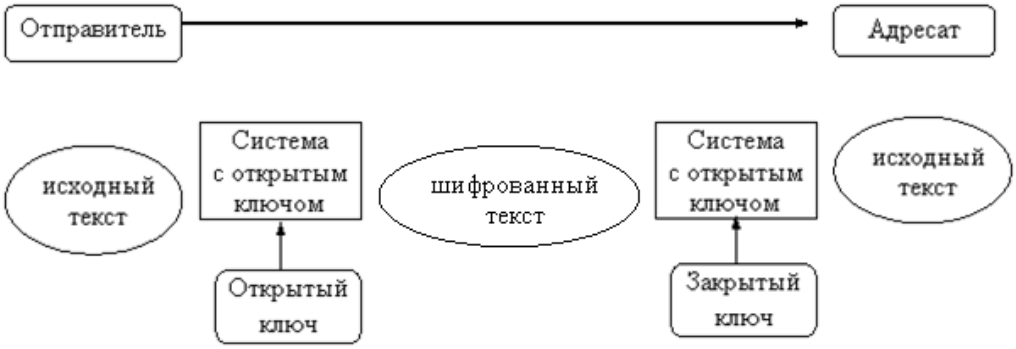
1 Асимметричные системы шифрования (системы с открытым ключом)

Как бы ни были сложны и надежны криптографические системы - их слабое мест при практической реализации - проблема *распределения ключей*. Для того, чтобы был возможен обмен конфиденциальной информацией между двумя субъектами ИС, ключ должен быть сгенерирован одним из них, а затем каким-то образом опять же в конфиденциальном порядке передан другому. Т.е. в общем случае для передачи ключа опять же требуется использование какой-то криптосистемы.

Для решения этой проблемы на основе результатов, полученных классической и современной алгеброй, были предложены *системы с открытым ключом.*

Суть их состоит в том, что каждым адресатом ИС генерируются два ключа, связанные между собой по определенному правилу. Один ключ объявляется *открытым*, а другой *закрытым*. Открытый ключ публикуется и доступен любому, кто желает послать сообщение адресату. Секретный ключ сохраняется в тайне.

Исходный текст шифруется открытым ключом адресата и передается ему. Зашифрованный текст в принципе не может быть расшифрован тем же открытым ключом. Дешифрование сообщения возможно только с использованием закрытого ключа, который известен только самому адресату.



Криптографические системы с открытым ключом используют так называемые *необратимые или односторонние функции*, которые обладают следующим свойством: при заданном значении *x* относительно просто вычислить значение *f(x),* однако если *y=f(x*), то нет простого пути для вычисления значения *x.*

Множество классов необратимых функций и порождает все разнообразие систем с открытым ключом. Однако не всякая необратимая функция годится для использования в реальных ИС.

В самом определении необратимости присутствует неопределенность. Под *необратимостью* понимается не теоретическая необратимость, а практическая невозможность вычислить обратное значение, используя современные вычислительные средства за обозримый интервал времени.

Поэтому чтобы гарантировать надежную защиту информации, к системам с открытым ключом (СОК) предъявляются два важных и очевидных требования:

1) Преобразование исходного текста должно быть необратимым и исключать его восстановление на основе открытого ключа.

2) Определение закрытого ключа на основе открытого также должно быть невозможным на современном технологическом уровне. При этом желательна точная нижняя оценка сложности (количества операций) раскрытия шифра.

Алгоритмы шифрования с открытым ключом получили широкое распространение в современных информационных системах. Так, алгоритм RSA стал мировым стандартом де-факто для открытых систем и рекомендован МККТТ.

Вообще же все предлагаемые сегодня криптосистемы с открытым ключом опираются на один из следующих типов необратимых преобразований:

1) Разложение больших чисел а простые множители.

2) Вычисление логарифма в конечном поле.

3) Вычисление корней алгебраических уравнений.

Здесь же следует отметить, что алгоритмы криптосистемы с открытым ключом (СОК) можно использовать в трех назначениях.

1) Как *самостоятельные средства защиты* передаваемых и хранимых данных.

2) Как *средства для распределения ключей*. Алгоритмы СОК более трудоемки, чем традиционные криптосистемы. Поэтому часто на практике рацио-

нально с помощью СОК распределять ключи, объем которых как информации незначителен. А потом с помощью обычных алгоритмов осуществлять обмен большими информационными потоками.

3) *Средства аутентификации пользователей*. Об этом будет рассказано в главе «Электронная подпись».

1 Функции дискретного логарифмирования

Понятия *однонаправленной функции* и *однонаправленной функции с "потайным ходом"*являются центральными понятиями всей криптографии с открытым ключом.

Функция $F(x) = y$ называется *однонаправленной* при выполнении 2-х условий:

- при любом x вычислить $F(x) = y$ было легко;
- зная y, вычислить такой x, чтобы $F(x) = y$ было недостижимо трудно.

Очевидно, что однонаправленные функции не могут непосредственно использоваться в качестве криптосистем, поскольку тогда даже законный получатель не сможет раскрыть зашифрованный текст!

Поэтому используется понятие *однонаправленной функции с потайным ходом*. Функция $F(x) = y$ называется однонаправленной функцией с потайным ходом, если она может эффективно вычисляться в обе стороны при условии знания определенного секрета.

Секрет, с помощью которого могут быть получены оба эффективных алгоритма, как раз и называется *потайным ходом*.

Однонаправленными (на современном уровне наших знаний) являются функции:

– Умножение целых чисел $f(x, y) = xy$

– Возведение заданного числа в степень по данному модулю

$F_{n,a}(m) = a^m \bmod n$

– Возведение числа в заданную степень по данному модулю

$F_{n,a}(a) = a^m \bmod n$

Нахождение функций, обратных к данным, приводит к задачам:

– Факторизация целого числа: $f = xy$, найти x и y, зная f.

– Дискретное логарифмирование: пусть $f_{n,a}(m) = a^m \bmod n$, найти m, зная n, f, a.

– Извлечение корня по модулю: пусть $F_{n,a}(a) = a^m \bmod n$, найти a, зная n, f, m.

Примечание: mod – остаток от деления

З а м е ч а н и е (об извлечении корня по модулю). Для последней задачи точно известно, что существует эффективный алгоритм, находящий корень по

модулю или устанавливающий, что такого корня нет. Выяснено, что для построения эффективного алгоритма достаточно знать разложение ***n*** на простые множители. Таким образом, используемых односторонних функций, собственно, две, и современные криптосистемы почти все основаны на проблеме факторизации целых чисел или на проблеме дискретного логарифма в конечной абелевой группе. Задачи эти в настоящее время одинаково трудны.

Факторизация(разложение на множители) — представление целого числа в виде произведения простых делителей

$N = p * q$, например

- 10 = 2 * 5,
- 1993423291715412685783 = 1868569 * 1066818132868207

Определение множителей является очень трудоемкой вычислительной операцией.

Дискретное логарифмирование – задача обращения функции $\mathbf{A^x = B}$.

Обратная данной функции будет $\mathbf{x = \log_A B}$.

Эффективные алгоритмы для решения этой задачи пока не известны.

Факторизация

Существует несколько методов факторизации числа $N = p * q$с простыми p и q:

Средневековые методы

- *Пробное деление,*
- *(p – 1)—метод,*
- *(p + 1)-метод,*
- *p-метод Полларда.*

Современные методы:

- *Метод непрерывных дробей,*
- *Квадратичное решето,*
- *Квадратичное решето в числовом поле.*

Время работы современных алгоритмов лежит где-то между полиномиальным и экспоненциальным, т.е. в суб-экспоненциальной области. Средневековые методы имеют экспоненциальную сложность.

Рассмотрим более подробно метод пробного деления, для рассмотрения других методов обратимся к дополнительной литературе.

Пробное деление. Это элементарный алгоритм факторизации.

N= p* q (p и q— простые числа) В этом алгоритме для всех простых чисел **p**, не превосходящих **N** проверяется условие — является ли **N/p** целым числом. Также существуют дополнительные условия, по которым мы может отсеять заведомо непростые числа, чтобы не тратить время на их проверку,

например никакое четное число больше 2 не может быть простым, а любое число с суммой цифр, кратной 3 делится на 3 и т.д.

Алгоритм имеет экспоненциальную сложность.

В известном конкурсном списке RSA приводятся большие числа, за факторизацию которых объявлены премии. Выпишем здесь для наглядности два из этих чисел:

число RSA-640 (640 двоичных, 193 десятичных цифры, уже факторизовано):

3107418240490043721350750035888567930037346022842727545720161948823206440518081504563468296a2328678243791627283803341547107310850191954852900733772482278352574238645401469173660247765234660 9;

число RSA-704 (704 двоичных, 212 десятичных цифр, не факторизовано, премия $30000):

7403756347956171282804679609742957314259318888923128908493623263897276503402826627689199641962511784399589433050212758537011896809828673317327310893090055250511687706329907239638078671008609696253793 4650563796359.

2 Схема RSA

На односторонней функции ***f(x, y) = xy*** основана предложенная в 1978 г. и выдержавшая испытание временем криптосистема RSA (Rivest, Shamir, Adleman).

Представим себе сеть абонентов, где каждые два должны иметь возможность обмениваться секретной информацией.

Каждый из абонентов сети:

– выбирает 2 различных простых числа ***p*** и ***q***;

– находит ***n = pq*** и функцию Эйлера ***m = (p–1)(q–1)***;

– Выбирает целое число ***e***, взаимно простое с ***m***.

– Вычисляет число ***d***, удовлетворяющее условию: ***ed = 1 (mod m)***.

– Секретным ключом абонента является тройка чисел ***(p, q, d)***, открытым ключом — пара чисел ***(n, e)***.

– Открытые ключи всех абонентов помещаются в общедоступный справочник.

Закодированный с помощью RSA текст защищён от несанкционированного прочтения настолько, насколько затруднено разложение на множители числа n.

3 Функции дискретного логарифмирования

Идея, лежащая в основе подобных систем, опирается на высокую вычислительную сложность обращения функции $F_{n,a}(x) = a^x \ mod n$

В данном случае, операция дискретного логарифмирования является обратной к данной функции. Последняя вычисляется достаточно просто, в то время как даже самые современные алгоритмы вычисления дискретного логарифма имеют очень высокую сложность, которая сравнима со сложностью наиболее быстрых алгоритмов разложения чисел на множители.

Другая возможность эффективного решения задачи вычисления дискретного логарифма связана с квантовыми вычислениями. Теоретически доказано, что, используя их, дискретный логарифм может быть вычислен за полиномиальное время. В любом случае, если полиномиальный алгоритм вычисления дискретного логарифма будет реализован, это будет означать практическую непригодность криптосистем на его основе.

Классическими криптографическими схемами, базирующимися на сложности задачи дискретного логарифмирования, являются схема выработки общего ключа Диффи-Хеллмана, система электронной подписи Эль-Гамаля, криптосистема Мэсси-Омуры для передачи сообщений.

Алгоритмы решения:

Существует алгоритм решения для любых групп чисел, но он медленный и не пригоден для практического использования. Для конкретных групп существуют свои, более эффективные, алгоритмы:

– В кольце вычетов по простому модулю(не все эти алгоритмы обладают суб-экспоненциальной сложностью)
– В произвольном конечном поле
– В группе точек на эллиптической кривой

4 Схема Эль-Гамаля

Система Эль-Гамаля основана на сложности вычисления дискретных логарифмов в конечных полях. Основным недостатком систем RSA и Эль-Гамаля является необходимость выполнения трудоемких операций в модульной арифметике, что требует привлечения значительных вычислительных ресурсов. *Криптосистема Мак-Элиса* использует коды, исправляющие ошибки. Она реализуется в несколько раз быстрее, чем криптосистема RSA, но имеет и существенный недостаток. В криптосистеме *Мак-Элиса* используется ключ большой длины и получаемый шифртекст в два раза превышает длину исходного текста.

Для всех методов шифрования с открытым ключом математически строго не доказано отсутствие других методов криптоанализа кроме решения NP-

полной задачи (задачи полного перебора). Если появятся методы эффективного решения таких задач, то криптосистемы такого типа будут дискредитированы. Например, ранее считалось, что задача укладки рюкзака является NP-полной. В настоящее время известен метод решения такой задачи, позволяющий избежать полного перебора.

Контрольные вопросы

1. Для каких целей может применяться алгоритм RSA?
2. Опишите процесс шифрования с использованием алгоритма RSA.
3. Для каких целей может применяться *алгоритм Диффи-Хеллмана*?
4. Опишите последовательность действий при использовании *алгоритма Диффи-Хеллмана*.
5. Для каких целей может применяться алгоритм Эль-Гамаля?.
6. Опишите последовательность действий при использовании алгоритма Эль-Гамаля.

Лекция 12 Основные требования к цифровой подписи, прямая и арбитражная цифровые подписи, стандарты цифровой подписи ГОСТ 3410 и DSS

План изложения

1 Основные подходы к формированию цифровой подписи на основе различных алгоритмов с открытым ключом.

2 Отечественные и зарубежные стандарты на алгоритмы цифровой подписи, применяемые в настоящее время.

1 Основные подходы к формированию цифровой подписи на основе различных алгоритмов с открытым ключом

Аутентификация защищает двух участников, которые обмениваются сообщениями, от воздействия некоторой третьей стороны. Однако простая аутентификация не защищает участников друг от друга, тогда как и между ними тоже могут возникать определенные формы споров.

Например, предположим, что Джон посылает Мери аутентифицированное сообщение, и аутентификация осуществляется на основе общего секрета. Рассмотрим возможные недоразумения, которые могут при этом возникнуть:

- Мери может подделать сообщение и утверждать, что оно пришло от Джона. Мери достаточно просто создать сообщение и присоединить аутентификационный код, используя ключ, который разделяют Джон и Мери.
- Джон может отрицать, что он посылал сообщение Мери. Так как Мери может подделать сообщение, у нее нет способа доказать, что Джон действительно посылал его.

В ситуации, когда обе стороны не доверяют друг другу, необходимо нечто большее, чем аутентификация на основе общего секрета. Возможным решением подобной проблемы является использование цифровой подписи. Цифровая подпись должна обладать следующими свойствами:

- должна быть возможность проверить автора, дату и время создания подписи;

- должна быть возможность аутентифицировать содержимое во время создания подписи;

- подпись должна быть проверяема третьей стороной для разрешения споров.

Таким образом, функция цифровой подписи включает функцию аутентификации.

На основании этих свойств можно сформулировать следующие требования к цифровой подписи:

- подпись должна быть битовым образцом, который зависит от подписываемого сообщения;

- подпись должна использовать некоторую уникальную информацию отправителя для предотвращения подделки или отказа;

- создавать цифровую подпись должно быть относительно легко;

- должно быть вычислительно невозможно подделать цифровую подпись как созданием нового сообщения для существующей цифровой подписи, так и созданием ложной цифровой подписи для некоторого сообщения;

- цифровая подпись должна быть достаточно компактной и не занимать много памяти.

Сильная хэш-функция, зашифрованная закрытым ключом отправителя, удовлетворяет перечисленным требованиям.

Существует несколько подходов к использованию функции цифровой подписи. Все они могут быть разделены на две категории: прямые и арбитражные.

Прямая и арбитражная цифровые подписи

При использовании прямой цифровой подписи взаимодействуют только сами участники, т.е. отправитель и получатель. Предполагается, что получатель знает открытый ключ отправителя. Цифровая подпись может быть создана

шифрованием всего сообщения или его хэш-кода закрытым ключом отправителя.

Конфиденциальность может быть обеспечена дальнейшим шифрованием всего сообщения вместе с подписью открытым ключом получателя (асимметричное шифрование) или разделяемым секретным ключом (симметричное шифрование). В случае возникновения спора некая третья сторона должна просмотреть сообщение и его подпись. Если функция подписи выполняется над зашифрованным сообщением, то для разрешения споров придется хранить сообщение как в незашифрованном виде (для практического использования), так и в зашифрованном (для проверки подписи). Либо в этом случае необходимо хранить ключ симметричного шифрования, для того чтобы можно было проверить подпись исходного сообщения. Если цифровая подпись выполняется над незашифрованным сообщением, получатель может хранить только сообщение в незашифрованном виде и соответствующую подпись к нему.

Действенность рассматриваемой схемы зависит от безопасности закрытого ключа отправителя. Если отправитель впоследствии не захочет признать факт отправки сообщения, он может утверждать, что закрытый ключ был потерян или украден, и в результате кто-то подделал его подпись. Можно применить административное управление, обеспечивающее безопасность закрытых ключей, для того чтобы ослабить эти угрозы. Один из возможных способов состоит в требовании в каждую подпись сообщения включать отметку времени (дату и время) и сообщать о скомпрометированных ключах в специальный центр.

Другая угроза состоит в том, что закрытый ключ может быть действительно украден у X в момент времени T. Нарушитель может затем послать сообщение, подписанное подписью X и помеченное временной меткой, которая меньше или равна T.

Проблемы, связанные с прямой цифровой подписью, могут быть частично решены с помощью арбитра. В общем виде арбитражная подпись выполняется следующим образом. Каждое подписанное сообщение от отправителя X к получателю Y первым делом поступает к арбитру A, который проверяет подпись для данного сообщения. После этого сообщение датируется и посылается к Y с указанием того, что оно было проверено арбитром. Присутствие A решает проблему схем прямой цифровой подписи, при которых X может отказаться от сообщения.

Арбитр играет важную роль в подобного рода схемах, и все участники должны ему доверять.

Рассмотрим возможную технологию арбитражной цифровой подписи с шифрованием открытым ключом, когда арбитр не видит сообщение. Все обсуждаемые проблемы могут быть решены с помощью схемы открытого ключа.

$X \rightarrow A$: $ID_X \| EKR_x [ID_X \| EKU_y [EKR_x [M]]]$

В этом случае X осуществляет двойное шифрование сообщения M, сначала своим закрытым ключом KR_X, а затем открытым ключом Y KU_Y. Получается подписанная секретная версия сообщения. Теперь это подписанное сообщение вместе с идентификатором X шифруется KR_X и вместе с ID_X посылается A. Внутреннее, дважды зашифрованное, сообщение недоступно арбитру (и всем, исключая Y). Однако A может дешифровать внешнюю шифрацию, чтобы убедиться, что сообщение пришло от X (так как только X имеет KR_X). Проверка дает гарантию, что пара закрытый/открытый ключ законна, и тем самым верифицирует сообщение.

$A \rightarrow Y$: $EKR_a [ID_X \| EKU_y [EKR_x [M]] \| T]$

Затем A передает сообщение Y, шифруя его KR_A. Сообщение включает ID_X, дважды зашифрованное сообщение и отметку времени.

Эта схема имеет ряд преимуществ. Во-первых, никакая информация не разделяется участниками до начала соединения, предотвращая договор об обмане. Во-вторых, некорректные данные не могут быть посланы, даже если KR_X скомпрометирован, при условии, что не скомпрометирован KR_A. В заключение, содержимое сообщения от X к Y неизвестно ни A, ни кому бы то ни было еще.

2 Отечественные и зарубежные стандарты на алгоритмы цифровой подписи, применяемые в настоящее время

Национальный институт стандартов и технологии США (NIST) разработал федеральный стандарт цифровой подписи DSS. Для создания цифровой подписи используется алгоритм DSA (Digital Signature Algorithm). В качестве хэш-алгоритма стандарт предусматривает использование алгоритма SHA-1 (Secure Hash Algorithm). DSS первоначально был предложен в 1991 году и пересмотрен в 1993 году в ответ на публикации, касающиеся безопасности его схемы.

DSS использует алгоритм, который разрабатывался для использования только в качестве цифровой подписи. В отличие от RSA, его нельзя использовать для шифрования или обмена ключами. Тем не менее, это технология открытого ключа.

Рассмотрим отличия подхода, используемого в DSS для создания цифровых подписей, от применения таких алгоритмов как RSA.

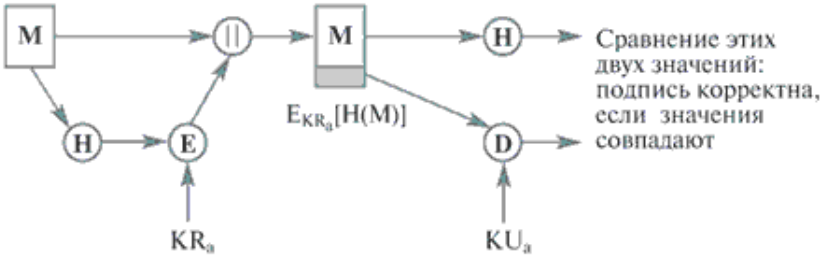


Рисунок 1 - Создание и проверка подписи с помощью алгоритма RSA

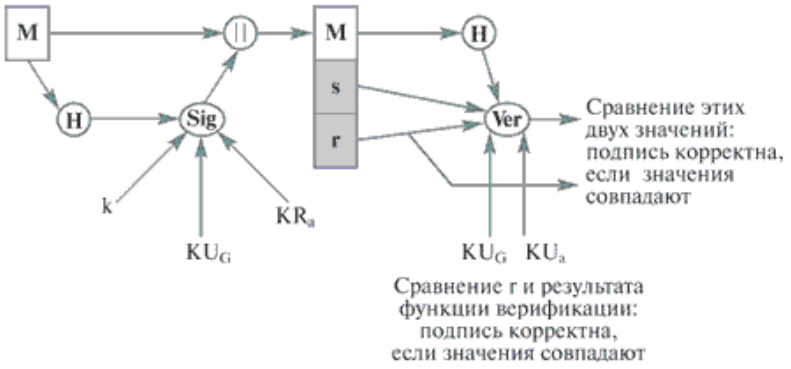


Рисунок 2 - Создание и проверка подписи с помощью стандарта DSS

В подходе RSA подписываемое сообщение подается на вход сильной хэш-функции, которая создает хэш-код фиксированной длины. Для создания подписи этот хэш-код шифруется с использованием закрытого ключа отправителя. Затем сообщение и подпись пересылаются получателю. Получатель вычисляет хэш-код сообщения и проверяет подпись, используя открытый ключ отправителя. Если вычисленный хэш-код равен дешифрованной подписи, то считается, что подпись корректна.

Подход DSS также использует сильную хэш-функцию. Хэш-код является входом функции подписи вместе со случайным числом k, созданным для этой конкретной подписи. Функция подписи также зависит от закрытого ключа отправителя KR_a и множества параметров, известных всем участникам. Можно считать, что это множество состоит из глобального открытого ключа KU_G. Результатом является подпись, состоящая из двух компонент, обозначенных как s и r.

Для проверки подписи получатель также создает хэш-код полученного сообщения. Этот хэш-код вместе с подписью является входом в функцию верификации. Функция верификации зависит от глобального открытого ключа KU_G и от открытого ключа отправителя KU_a. Выходом функции верификации

является значение, которое должно равняться компоненте r подписи, если подпись корректна. Функция подписи такова, что только отправитель, знающий закрытый ключ, может создать корректную подпись.

Теперь рассмотрим детали алгоритма, используемого в DSS.

DSS основан на трудности вычисления дискретных логарифмов и базируется на схеме, первоначально представленной ElGamal и Schnorr.

Существует три параметра, которые являются открытыми и могут быть общими для большой группы пользователей:

160-битное простое число q, т.е. $2^{159} < q < 2^{160}$.

Простое число p длиной между 512 и 1024 битами должно быть таким, чтобы q было делителем (p - 1), т.е. 2L-1 < p < 2L, где 512 < L < 1024 и (p-1)/q является целым.

$g = h^{(p-1)/q} \bmod p$, где h является целым между 1 и (p-1).

Зная эти числа, каждый пользователь выбирает закрытый ключ и создает открытый ключ.

Закрытый ключ x должен быть числом между 1 и (q-1) и должен быть выбран случайно или псевдослучайно.

x - случайное или псевдослучайное целое, 0 < x < q.

Открытый ключ вычисляется из закрытого ключа как $y = g^x \bmod p$. Вычислить y по известному x довольно просто. Однако, имея открытый ключ y, вычислительно невозможно определить x, который является дискретным логарифмом y по основанию g.

$y = g^x \bmod p$

Случайное число, уникальное для каждой подписи.

k - случайное или псевдослучайное целое, 0 < k < q, уникальное для каждого подписывания.

Для создания подписи отправитель вычисляет две величины, r и s, которые являются функцией от компонент открытого ключа (p, q, g), закрытого ключа пользователя (x), хэш-кода сообщения H (M) и целого k, которое должно быть создано случайно или псевдослучайно и должно быть уникальным при каждом подписывании.

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

$s = [\ k^{-1} (H (M) + xr)\ ] \bmod q$

Подпись = (r, s)

Получатель выполняет проверку подписи с использованием следующих формул. Он создает величину v, которая является функцией от компонент общего открытого ключа, открытого ключа отправителя и хэш-кода полученного сообщения. Если эта величина равна компоненте r в подписи, то подпись считается действительной.

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

$u1 = [H(M) w] \bmod q$

$u2 = r w \bmod q$

$v = [(g^{u1} y^{u2}) \bmod p] \bmod q$

подпись корректна, если $v = r$

Отечественный стандарт цифровой подписи ГОСТ 3410

В отечественном стандарте ГОСТ 3410, принятом в 1994 году, используется алгоритм, аналогичный алгоритму, реализованному в стандарте DSS. Оба алгоритма относятся к семейству алгоритмов ElGamal.

В стандарте ГОСТ 3410 используется хэш-функция ГОСТ 3411, которая создает хэш-код длиной 256 бит. Это во многом обуславливает требования к выбираемым простым числам p и q:

p должно быть простым числом в диапазоне

$2^{509} < p < 2^{512}$

либо

$2^{1020} < p < 2^{1024}$

q должно быть простым числом в диапазоне

$2^{254} < q < 2^{256}$

q также должно быть делителем (p-1).

Аналогично выбирается и параметр g. При этом требуется, чтобы $g^q \pmod p = 1$.

В соответствии с теоремой Ферма это эквивалентно условию в DSS, что $g = h^{(p-1)/q} \bmod p$.

Закрытым ключом является произвольное число x

$0 < x < q$

Открытым ключом является число y

$y = g^x \bmod p$

Для создания подписи выбирается случайное число k

$0 < k < q$

Подпись состоит из двух чисел (r, s), вычисляемых по следующим формулам:

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

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

Еще раз обратим внимание на отличия DSS и ГОСТ 3410.

Используются разные хэш-функции: в ГОСТ 3410 применяется отечественный стандарт на хэш-функции ГОСТ 3411, в DSS используется SHA-1, которые имеют разную длину хэш-кода. Отсюда и разные требования на длину простого числа q: в ГОСТ 3410 длина q должна быть от 254 бит до 256 бит, а в DSS длина q должна быть от 159 бит до 160 бит.

По-разному вычисляется компонента s подписи. В ГОСТ 3410 компонента s вычисляется по формуле

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

В DSS компонента s вычисляется по формуле

$s = [k^{-1} (H(M) + xr)] \bmod q$

Последнее отличие приводит к соответствующим отличиям в формулах для проверки подписи.

Получатель вычисляет

$w = H(M)^{-1} \bmod q$

$u1 = w\ s \bmod q$

$u2 = (q-r)\ w \bmod q$

$v = [(g^{u1}\ y^{u2}) \bmod p] \bmod q$

Подпись корректна, если $v = r$.

Структура обоих алгоритмов довольно интересна. Заметим, что значение r совсем не зависит от сообщения. Вместо этого r есть функция от k и трех общих компонент открытого ключа. Мультипликативная инверсия k (mod p) (в случае DSS) или само значение k (в случае ГОСТ 3410) подается в функцию, которая, кроме того, в качестве входа имеет хэш-код сообщения и закрытый ключ пользователя. Эта функция такова, что получатель может вычислить r, используя входное сообщение, подпись, открытый ключ пользователя и общий открытый ключ.

В силу сложности вычисления дискретных логарифмов нарушитель не может восстановить k из r или x из s.

Другое важное замечание заключается в том, что экспоненциальные вычисления при создании подписи необходимы только для $g^k \bmod p$. Так как это значение от подписываемого сообщения не зависит, оно может быть вычислено заранее. Пользователь может заранее просчитать некоторое количество значений r и использовать их по мере необходимости для подписи документов. Еще одна задача состоит в определении мультипликативной инверсии k^{-1} (в случае DSS). Эти значения также могут быть вычислены заранее.

Подписи, созданные с использованием стандартов ГОСТ 3410 или DSS, называются рандомизированными, так как для одного и того же сообщения с использованием одного и того же закрытого ключа каждый раз будут создаваться разные подписи (r,s), поскольку каждый раз будет использоваться новое значение k. Подписи, созданные с применением алгоритма RSA, называются детерминированными, так как для одного и того же сообщения с использованием одного и того же закрытого ключа каждый раз будет создаваться одна и та же подпись.

Контрольные вопросы

1. Какие асимметричные алгоритмы и как могут применяться для формирования и проверки электронной цифровой подписи?
2. Опишите процесс создания и проверки цифровой подписи с использованием различных *асимметричных алгоритмов*.
3. Какие стандарты действуют на алгоритмы формирования и проверки электронной цифровой подписи в России?
4. Какие цифровые подписи называются рандомизированными?
5. В чем заключается проблема сертификации открытых ключей?
6. Что включается в понятие инфраструктуры открытых ключей?
7. Каковы функции центра сертификации открытых ключей?
8. Что такое сертификат открытого ключа?
9. Какая схема распределения открытых ключей абонентов может использоваться в системе связи, имеющей в своем составе центр сертификации открытых ключей?

Список использованных источников

1. Владимир Шаньгин Информационная безопасность и защита информации / В. Шаньгин. - СПб.: Интермедия, 2017. – 702 с. [Электронный ресурс] – Режим доступа https://www.ozon.ru/context/detail/id/137767314/

2. Мельников В.П. Информационная безопасность и защита информации. (6-е издание) / В.П. Мельников, С.А. Клейменов, А.М. Петраков. - М.: Академия, 2012. - 336 с. [Электронный ресурс] – Режим доступа https://hi-media.ru/3079-informacionnaya-bezopasnost-i-zaschita-informacii-6-e-izdanie.html

3. Методы и средства защиты информации / В.А.Хорошко, А.А.Чекатков Под ред. Ю.С.Ковтанюка. - М.: Проспект, 2015. - 328 с. [Электронный ресурс] – Режим доступа http://asher.ru/security/book/mszi

4. Федоров Н.В. Криптографические методы и средства защиты информации. Наука и учеба / Н.В. Федоров. - СПб.: Интермедия, 2018. – 384 с. [Электронный ресурс] – Режим доступа http://abookz.net/books-skachat/books-training/95376-fedorov-nv-kriptograficheskie-metody-i-sredstva-zaschity-informacii.html

Защита информации. Инсайд [Электронный ресурс] электрон. журн. – Режим доступа http://www.inside-zi.ru/

2. Проблемы информационной безопасности. Компьютерные системы. [Электронный ресурс] электрон. журн. – Режим доступа http://jisp.ru/

3. Computerworld Россия [Электронный ресурс] электрон. журн. – Режим доступа: https://www.computerworld.ru/

4. Научно-технический вестник информационных технологий, механики и оптики [Электронный ресурс] электрон. журн. – Режим доступа: https://ntv.ifmo.ru/

5. CNews [Электронный ресурс] электрон. журн. – Режим доступа: http://www.cnews.ru/

Учебное издание

КОНСПЕКТ ЛЕКЦИЙ
по дисциплине
«МЕТОДЫ И СРЕДСТВА ЗАЩИТЫ КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ»
для студентов направления подготовки
Профессиональное обучение (по отраслям),
профиль
«Информационные технологии и системы»
(в 3-х частях). Часть 3

С о с т а в и т е л ь:
Ирина Владимировна Ганзенко

Печатается в авторской редакции.
Компьютерная верстка и оригинал-макет автора.

Подписано в печать _____
Формат 60x84 1/16. Бумага типограф. Гарнитура Times
Печать офсетная. Усл. печ. л. _____. Уч.-изд. л. _____
Тираж 100 экз. Изд. №_____. Заказ №_____. Цена договорная.

Издательство Луганского государственного
университета имени Владимира Даля

Свидетельство о государственной регистрации издательства МИ-СРГ ИД 000003 от 20 ноября 2015г.

Адрес издательства: 91034, г. Луганск, кв. Молодежный, 20а
Телефон: 8 (0642) 41-34-12, **факс:** 8 (0642) 41-31-60
E-mail: izdat.lguv.dal@gmail.com **http:** //izdat.dahluniver.ru/