Государственное автономное образовательное учреждение среднего профессионального образования РБ «Бурятский республиканский многопрофильный техникум инновационных технологий»

Костылева О. П.

УЧЕБНОЕ ПОСОБИЕ ПО МДК.02.02.

КРИПТОГРАФИЧЕСКИЕ СРЕДСТВА И МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Костылева О. П.

Учебное пособие по МДК.02.02. Криптографические средства и методы защиты информации для обучающихся по специальности среднего профессионального образования 090305 Информационная безопасность автоматизированных систем.

Костылева О. П. – Северобайкальск: мини-типография ГАОУ СПО РБ «БРМТИТ», 2013. 156 страниц.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Необходимость защиты информации, появилась, с развитием языка. Криптография, как тайнопись, возникла с распространением письменности. Необходимость передавать сообщения скрытно, требовала невозможности прочтения теми, кому это сообщение не было предназначено. Способы решения этой задачи можно разделить на несколько групп:

Организационные меры: действия, которые должны быть выполнены отправителем, посредниками и получателем, для того, что бы предотвратить перехват сообщения.

Например, отправитель не должен никому сообщать, о том, что сообщение отправлено, с кем, и в каком виде. Посыльный должен уничтожить сообщение (сжечь, разорвать, съесть) в случае, если он попал в плен. Сам он не должен знать содержимого сообщения. Получатель должен сохранить сообщение в надежном месте или уничтожить, если как документ оно не представляет ценности.

Организационные методы устанавливают правила обращения с информацией лиц имеющих к ней доступ, регламентируют порядок доступа, и должны предотвращать несанкционированный доступ. Далее будем считать, что злоумышленник смог получить частичный доступ к информации (по крайней мере, к каналам связи).

Технические средства: средства, препятствующие естественному распространению информации, циркулирующей в открытом (не зашифрованном) виде (речь, изображение, информация на экране компьютера или во время обработки), а также средства контроля доступа. Такими средствами являются звукоизоляционные и экранирующие материалы, генераторы электромагнитного и виброаккустического шума, средства защиты от прослушивания линий связи и др. Эти методы не имеют прямого отношения к криптографии и рассматриваться не будут. Будем считать, что злоумышленник смог получить доступ к каналам связи, снять ЭМ наводки или получить доступ непосредственно к передающей аппаратуре.

Правовые методы: устанавливается ответственность за несанкционированный доступ к защищаемой информации, или за ее разглашение. Будем считать, что стоимость информации значительно выше, грозящей ответственности.

Стеганография: маскировка или сокрытие передаваемого сообщения. При этом для перехватчика факт передачи сообщения может быть скрыт; передаваемое сообщение может выглядеть, как незашифрованное и не представляющее ценности; или как незашифрованное и ценное, но быть ложным и вводить перехватчика в заблуждение. Стеганографическими методами скрытия информации в древнем мире были: на обритую голову раба записывалось необходимое сообщение, а когда его

волосы отрастали, он отправлялся к адресату; сообщение наносилось на деревянную дощечку, а потом она покрывалась воском. С I и вплоть до XX в. были распространены симпатические (невидимые) чернила. Во время второй мировой войны активно использовались микроточки – микроскопические фотоснимки, вклеиваемые в текст писем. В настоящее время развивается компьютерная стеганография, информация скрывается в неиспользуемых областях дисков или файлов или используются младшие, малозначащие биты медиа форматов.

Стеганография редко применяется для защиты информации. В сравнении с криптографией – ее применение равносильно применению неизвестных криптографических алгоритмов, однако стеганография, совместно с криптографией может использоваться для скрытия факта передачи или хранения информации. В этом случае, зная алгоритм, но не имея ключа, невозможно проверить наличие скрытого сообщения. Будем считать, что злоумышленнику известен факт передачи, форматы, протоколы и алгоритмы шифрования, передаваемых данных.

Криптография: изначально – шифрование (обеспечение конфиденциальности) по неизвестному алгоритму или с неизвестным ключом (параметрами шифрования). На данный момент в задачи криптографии входит проверка целостности, достоверности, обеспечение юридической значимости, неотслеживаемости и др.

В данном учебном пособии рассматриваются вопросы криптографической защиты информации или имеющие к ним непосредственное отношение. Рассмотрены алгоритмы симметричного и ассиметричного шифрования, способы защиты сообщений от искажения, схемы электронно-цифровых подписей, способы построения инфраструктуры открытых ключей.

ТЕМА 1. ИСТОРИЯ КРИПТОГРАФИИ

В истории развития криптографии можно выделить четыре этапа [15, 16].

- Наивная криптография.
- Формальная криптография.
- Научная криптография.
- Компьютерная криптография.

Наивная криптография

Для **наивной криптографии** (с древнейших времен до начала XVI в.) характерно использование любых, обычно примитивных, способов запутывания противника относительно содержания передаваемых сообщений. Одни и те же способы шифрования использовались повторно, а ключи были короткими или вовсе отсутствовали. Это позволяло, однажды установив алгоритм шифрования, быстро расшифровывать сообщения. В этот период применяются в основном шифры перестановки и простой замены.

При шифровании заменой (подстановкой) символы шифруемого текста заменяются символами того же или другого алфавита с заранее установленным правилом замены.

В шифре простой замены каждый символ исходного текста заменяется символами того же алфавита одинаково на всем протяжении текста. Первые известные нам шифры такого типа появились в I в. до н. э. и связываются с именем Гая Юлия Цезаря, шифровавшего переписку заменой букв, на отстоящие от них в алфавите на 3 символа.

Некриптографическим устройством защиты информации был «диск Энея». Древнегреческий полководец Эней Тактика в IV веке до н. э. предложил устройство, названное впоследствии «диском Энея». Принцип его таков. На диске диаметром 10–15 см и толщиной 1–2 см высверливались отверстия по числу букв алфавита. В центре диска помещалась «катушка» с намотанной на ней ниткой достаточной длины. При шифровании нитка «вытягивалась» с катушки и последовательно протягивалась через отверстия, в соответствии с буквами шифруемого текста. Диск и являлся посланием. Получатель послания последовательно вытягивал нитку из отверстий, что позволяло ему получать передаваемое сообщение, но в обратном порядке следования букв. При перехвате диска недоброжелатель имел возможность прочитать сообщение тем же образом, что и получатель. Но Эней предусмотрел возможность легкого уничтожения передаваемого сообщения при угрозе захвата диска. Для этого было достаточно выдернуть «катушку» с закрепленным на ней концом нити до полного выхода всей нити из всех отверстий диска. На

основе этого устройства был позже основан метод шифрования. В одном из вариантов вместо диска использовалась линейка с числом отверстий, равных количеству букв алфавита. Каждое отверстие обозначалось своей буквой; буквы по отверстиям располагались в произвольном порядке. К линейке была прикреплена катушка с намотанной на нее ниткой.
Рядом с катушкой имелась прорезь. При шифровании нить протягивалась через прорезь, а затем через отверстие, соответствующее первой букве шифруемого текста, при этом на нити завязывался узелок в месте прохождения ее через отверстие; затем нить возвращалась в прорезь и аналогично зашифровывалась вторая буква текста и т. д. После окончания шифрования нить извлекалась и передавалась получателю сообщения. Тот, имея идентичную линейку, протягивал нить через прорезь до отверстий, определяемых узлами, и восстанавливал исходный текст по буквам отверстий. Такой шифр также можно отнести к шифрам замены [17].

При шифровании перестановкой, символы шифруемого текста переставляются по определенному правилу в пределах блока этого текста. Шифры перестановки являются самыми простыми и, вероятно, самыми древними шифрами.

Одним из первых известных шифров перестановки считается шифр «скитала» («сцитала»). Шифр применялся правителями Спарты с V в. до н. э. На стержень цилиндрической формы, который назывался сцитала, наматывали спиралью (виток к витку) полоску пергамента и писали на ней вдоль стержня несколько строк текста сообщения. Затем снимали со стержня полоску пергамента с написанным текстом. Буквы на этой полоске казались расположенными хаотично [14].

В эпоху возрождения получили распространение шифры табличной перестановки. Сообщение записывалось в таблицу в определенном порядке, в таблице могли переставляться столбцы и/или строки, после чего сообщение выписывалось в другом порядке.

Формальная криптография

Этап формальной криптографии (конец XV – начало XX в.) связан с появлением формализованных и относительно стойких к ручному криптоанализу шифров. В европейских странах это произошло в эпоху Возрождения, когда развитие науки и торговли вызвало спрос на надежные способы защиты информации. Важная роль на этом этапе принадлежит Леону Батисте Альберти, итальянскому архитектору, который одним из первых предложил *многоалфавитную подстановку.* Данный шифр, получивший имя дипломата XVI в. Блеза Вижинера, состоял в последовательном «сложении» букв исходного текста с ключом (процедуру можно

облегчить с помощью специальной таблицы). Его работа «Трактат о шифре» (1466) считается первой научной работой по криптографии. Другой печатной работой, в которой обобщены и сформулированы известные на тот момент алгоритмы шифрования, является труд «Полиграфия» (1508) немецкого аббата Иоганна Трисемуса. Ему принадлежит также предложение заменять в сообщениях пары букв.

Простым, но стойким способом многоалфавитной замены (подстановки биграмм) является шифр Плейфера, который был открыт в начале XIX в. Чарльзом Уитстоном. Уитстону принадлежит и важное совершенствование – шифрование «двойным квадратом». Шифры Плейфера и Уитстона использовались вплоть до первой мировой войны, так как с трудом поддавались ручному криптоанализу.

В XIX в. голландец Керкхофф сформулировал главное требование к криптографическим системам, которое остается актуальным и поныне: *секретность шифров должна быть основана на секретности ключа, но не алгоритма*.

Наконец, последним словом в донаучной криптографии, которое обеспечило еще более высокую криптостойкость, а также позволило автоматизировать (в смысле механизировать) процесс шифрования стали *роторные (дисковые) криптосистемы.*

Одной из первых подобных систем стала изобретенная в 1790 г. Томасом Джефферсоном, будущим президентом США, механическая машина.

Практическое распространение роторные машины получили только в начале XX в. Одной из первых практически используемых машин, стала немецкая *Enigma*, разработанная в 1917 г. Эдвардом Хеберном и усовершенствованная Артуром Кирхом. *Дисковые шифраторы* активно использовались во время второй мировой войны. Помимо немецкой машины Enigma использовались также устройства *Sigaba* (США), *Typex* (Великобритания), *Red*, *Orange* и *Purple* (Япония).

У дискового шифратора есть два типа ключей. Одни ключи – долговременные, это перепайки между контактами дисков, т. е. те подстановки, которые соответствуют каждому диску. Их смена означает смену самого диска, и производится довольно редко, например, раз в месяц или даже в год. Другие ключи – начальное расположение дисков друг относительно друга. Их можно менять гораздо чаще, делать различными для каждой телеграммы в зависимости от ее номера. Такие ключи называются сеансовыми или разовыми. Количество долговременных ключей для каждого диска составляет 26! различных вариантов его перепаек, а дисков несколько, иногда по 6, поэтому общее количество долговременных ключей недоступно для перебора и по сей день, $(26!)^6$, что сопоставимо с ключом длиной 530 бит. Разовых же ключей намного меньше, всего $(26)^6$ различных вариантов, такая работа была по силам ЭВМ с

момента их появления, но без знания подстановок надеяться дешифровать дисковый шифратор бесполезно.

Получить долговременные подстановки было возможно, только получив сами диски. Например, во время второй мировой войны американцами была захвачена немецкая подводная лодка U-571 с находящемся на ней дисковым шифратором «Энигма» только для того, чтобы получить неизвестные подстановки [26].

Роторные системы – вершина формальной криптографии, так как относительно просто реализовывали очень стойкие шифры. Успешные криптоатаки на роторные системы стали возможны только с появлением ЭВМ.

Научная криптография

Главная отличительная черта научной криптографии (1930–60-е гг.) – строгое математическое обоснование криптостойкости. К началу 30-х гг. окончательно сформировались разделы математики, являющиеся научной основой криптологии: теория вероятностей и математическая статистика, общая алгебра, теория чисел, начали активно развиваться теория алгоритмов, теория информации, кибернетика.

Своеобразным водоразделом стала работа Клода Шеннона «Теория связи в секретных системах» (1949), которая подвела научную базу под криптографию и криптоанализ. Шеннон ввел понятия «рассеивание» и «перемешивание», обосновал возможность создания сколь угодно стойких криптосистем.

В 1960-х гг. ведущие криптографические школы подошли к созданию блочных шифров, еще более стойких по сравнению с роторными криптосистемами, однако допускающих практическую реализацию только в виде цифровых электронных устройств.

Компьютерная криптография

С 1970-х гг. появились вычислительные средства с производительностью, достаточной для реализации криптосистем, обеспечивающих при большой скорости шифрования высокую криптостойкость. Первым классом криптосистем, практическое применение которых стало возможно с появлением мощных и компактных вычислительных средств, стали криптосистемы основанные на блочных шифрах. Был разработан американский стандарт шифрования DES (создан в 1972 г. и принят в 1978 г.) в течение более 20 лет позволявший обеспечивать приемлимый уровень безопасности. Один из его авторов, Хорст Фейстель, описал модель блочных шифров, на основе которой были построены другие,

более стойкие симметричные криптосистемы, в том числе отечественный стандарт шифрования ГОСТ 28147–89.

Появились понятия дифференциальный и линейный криптоанализ, причем DES уже разрабатывался стойким к дифференциальному криптоанализу, когда еще не было введено такое понятие. В настоящее время стоит вопрос об эффективности алгебраических атак (XSL-атак) на блочные шифры.

В середине 70-х гг. XX в. появляются асимметричные криптосистемы, которые не требовали передачи секретного ключа между сторонами. Здесь отправной точкой принято считать работу, опубликованную Уитфилдом Диффи и Мартином Хеллманом в 1976 г. под названием «Новые направления в современной криптографии». В ней впервые сформулированы принципы обмена шифрованной информацией без обмена секретным ключом.

Независимо к идее асимметричных криптосистем подошел Ральф Меркли. Несколькими годами позже Рон Ривест, Ади Шамир и Леонард Адлеман создали систему RSA, первую практическую асимметричную криптосистему, стойкость которой была основана на проблеме факторизации (разложения на множители) больших простых чисел. Асимметричная криптография открыла сразу несколько новых прикладных направлений, в частности системы электронной цифровой подписи и электронных денег. Создание ассиметричных криптосистем подтолкнуло математиков и криптоаналитиков к изучению способов факторизации, дискретного логарифмирования. Благодаря их успехам теперь для обеспечения безопасности при использовании RSA требуются ключи большого размера. Успехи в криптоанализе подтолкнули криптографов к изучению других областей, например операций над эллиптическими кривыми в конечном поле. Эллиптические кривые еще мало изучены и криптосистемы на их основе позволяют использовать относительно короткие ключи, но ситуация может измениться.

Относительно новым и малораспространенным методом является вероятностное шифрование. Вероятностное шифрование предложили Шафи Гольдвассер (Goldwasser) и Сильвио Микэли (Micali). Шифрование было названо «вероятностным» в связи с тем, что один и тот же исходный текст при шифровании с использованием одного и того же ключа может преобразовываться в совершенно различные шифротексты. При использовании криптосистем с открытым ключом существует возможность подбора открытого текста сопоставлением перехваченного шифротекста с результатом шифрования. Вероятностное шифрование позволяет на порядки увеличить сложность такого вида атаки.

Беннет (Bennet) и Брассард (Brassard), опираясь на работу Уиснера (Wiesner), разработали теорию квантовой криптографии, которая базируется на квантовой физике. Разработанная ими система, в отличие от

обычной криптографии, теоретически позволяет гарантированно защитить информацию от злоумышленника, даже если тот обладает самой современной технологией и неограниченными вычислительными мощностями. На данный момент, разрабатываются только тестовые варианты квантовых криптосистем.

В тоже время эффекты квантовой физики, возможно, смогут использоваться и для криптоанализа. Если будут построены квантовые компьютеры, то это поставит под вопрос существование современной криптографии.

ТЕМА 2. ОСНОВНЫЕ ПОНЯТИЯ

Прежде чем перейти к рассмотрению простых шифров, необходимо уточнить терминологию. Здесь даются только основные термины, необходимые для понимания следующих тем.

Криптография (от греч. κρυπτός скрытый и γράφω пишу) – наука о математических методах преобразования данных с целью сокрытия их информационного содержания, предотвращения их необнаружимой модификации, подтверждения подлинности авторства, невозможности отказа от авторства и решения других задач обеспечения информационной безопасности.

Пользователи и злоумышленники

Отправитель хочет послать сообщение **получателю**. Отправитель не может быть уверен в том, что сообщение не будет перехвачено, но он хочет быть уверенным, что оно не сможет быть прочитано.

Отправитель и получатель – это **законные пользователи** криптосистемы. В некоторых криптографических протоколах существует множество законных пользователей с различными ролями.

Криптографии противостоит **криптоанализ**, а законным пользователям – **криптоаналитики**.

Криптоанализ (от греч. κρυπτός скрытый и αναλυσις разложение) – наука о методах получения исходного значения зашифрованной информации, не имея доступа к секретной информации (ключу), необходимой для этого. В большинстве случаев под этим подразумевается нахождение ключа. Так же криптоанализ может позволить обнаружить слабые места системы.

Криптоаналитик – специалист по криптоанализу (так же иногда используются слова мошенник, злоумышленник, взломщик, враг, противник и т. д.).

Понятия криптоанализ и криптоаналитик могут относиться как к анализу защищенности системы, так и к взлому функционирующей криптосистемы.

Криптология – отрасль математики, охватывающая криптографию и криптоанализ.

Открытый текст и шифротекст

Сообщение – это упорядоченный набор из элементов алфавита. **Алфавит** – конечное множество используемых для кодирования информации знаков (символов). Необходимо различать понятия кодирование и шифрование (их иногда употребляют как синонимы). Кодирование – это только способ представления символов. Например, шестнадцатеричные коды, ASCII коды, коды азбуки Морзе и т. д.

Исходное, незашифрованное сообщение называется **открытым (исходным) текстом.** Открытый текст обычно обозначается M, T или P, от слов message, text и plaintext.

Процесс изменения вида сообщения с целью скрытия его содержимого называется **шифрованием**. Измененное таким образом сообщение называется **шифротекстом**. Шифротекст обычно обозначается C, от cipher text.

Процесс получения открытого текста из шифротекста, называется **расшифрованием**.

Криптографическая система

Алгоритм криптографического преобразования (или **шифр**) – набор математических правил, определяющих содержание и последовательность операций по шифрованию и расшифрованию информации.

Если безопасность алгоритма основана на сохранении в тайне самого алгоритма, то такой алгоритм называется **ограниченным**. Ограниченные алгоритмы представляют только исторический интерес, но они совершенно не удовлетворяют сегодняшним стандартам. Такие шифры неприменимы для использования большой или изменяющейся группой пользователей.

При разработке системы необходимо считать, что врагу известны подробности вашего алгоритма, потому что в конечном итоге они все равно станут ему известны. Август Керхкофф первым сформулировал это положение в 1883 г.: «В алгоритме нет никакой тайны, вся тайна в ключе». Тем не менее, ограниченные алгоритмы очень популярны для приложений с низким уровнем безопасности. Подобные алгоритмы иногда используются для генерации и проверки лицензионных ключей.

Ключ – конкретное состояние некоторых параметров алгоритма криптографического преобразования, обеспечивающее выбор одного преобразования из совокупности всевозможных, для данного алгоритма.

Множество возможных ключей называют **пространством ключей**.

Размер пространства ключей, является одним из основных параметров первичной оценки безопасности и дает информацию о том, какой объем работы необходим для полного перебора ключей. Однако если система шифрования содержит уязвимости, полный перебор ключей может не потребоваться.

Криптографическая система – система обеспечения безопасности информации криптографическими методами, включает совокупность систем шифрования, расшифрования, генерации и распределения ключей и других подсистем, необходимых для реализации криптографических протоколов. Иногда криптографическую систему называют синонимом алгоритма или шифра.

В результате шифрования открытого текста T алгоритмом E с ключом K_1 получаем шифротекст C (рис. 1):

$$E_{k1}(T) = C.$$

Расшифрование осуществляется обратным алгоритмом D с ключом K_2:

$$D_{k2}(C) = D_{k2}(E_{k1}(T)) = T.$$

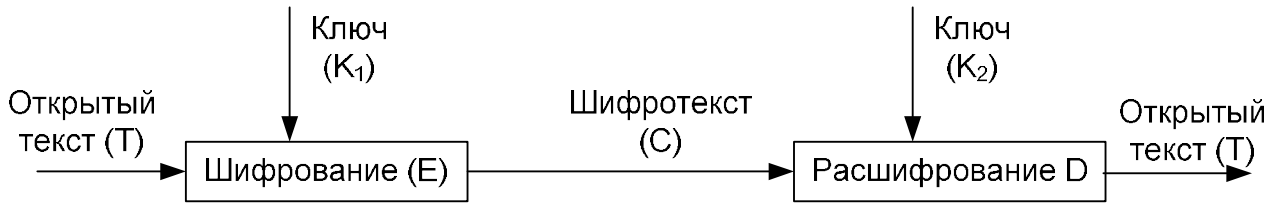


Рис. 1. Схема передачи данных в криптографической системе

Если для шифрования и расшифрования используется один и тот же ключ ($K_1 = K_2$), то криптографическая система называется **симметричной,** или системой с **закрытым (секретным) ключом**, иначе, если используются различные ключи ($K_1 \neq K_2$) – **ассиметричной** или системой с **открытыми ключами**. Ключ, используемый для шифрования, называют **открытым** или публичным, ключ для расшифрования – **закрытым (секретным)**. Закрытый ключ должен всегда сохраняться в тайне, открытый может быть свободно опубликован.

Виды атак

Атака – попытка взлома (вскрытия) части или всей криптосистемы.

На криптосистему могут производиться атаки различных видов.

Пассивная атака – атака на криптографическую систему, при которой противник наблюдает шифрованные сообщения, но не влияет на действия законных пользователей (рис. 2).

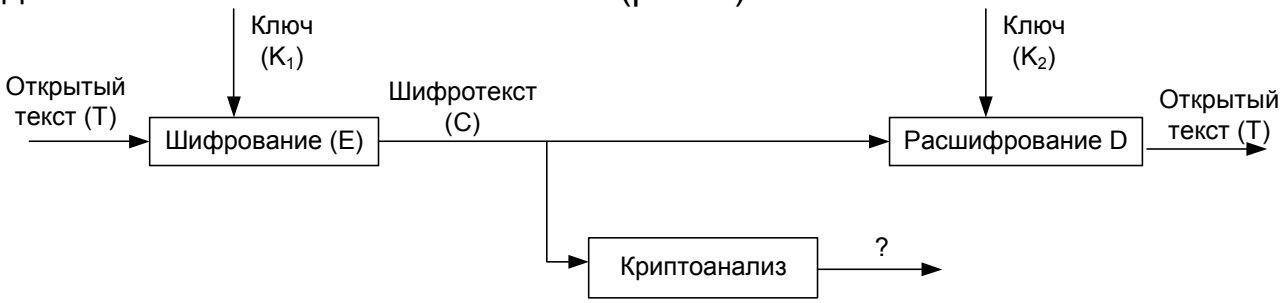


Рис. 2. Схема реализации пассивной криптоатаки

В результате криптоанализа передаваемого шифротекста, криптоаналитик может получить открытый текст, ключ, некоторые параметры сообщения (например его длину), или не получить ничего.

Активная атака – атака на криптосистему или криптографический протокол, при которой противник может изменять сообщения или влиять на действия законного пользователя.

На рис. 3 показана атака с модификацией шифротекста. В данном случае криптоаналитик имеет возможность не только прослушивать канал связи, но и изменять передаваемые сообщения. Если в системе не определено, что сообщение должно быть обязательно зашифровано (например, в почтовых протоколах), то злоумышленник может просто передать другой правдоподобный открытый текст. Если шифр может быть вскрыт за разумный период времени, то злоумышленник может подменить шифротекст так, что при расшифровании его получателем он будет осмысленным, но ложным. Также существует ряд атак на криптографические протоколы типа «человек посередине», например, при обмене открытыми ключами. В этом случае, для каждого участника обмена, злоумышленник представляется законным получателем.

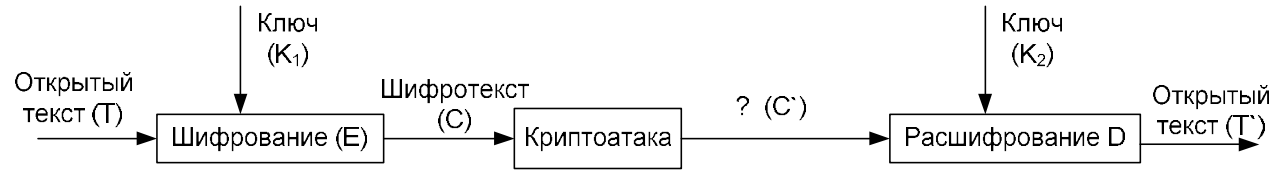


Рис. 3. Схема варианта реализации активной криптоатаки

При разработке криптографических систем необходимо исходить из принципа, что криптоаналитик может иметь доступ к каналам связи, знать, как устроена система, как генерируются и распределяются ключи, выступать в роли одной из сторон, знать какие ключи использовались

ранее, может предположить некоторые участки открытого текста или знать ранее использовавшиеся открытые тексты.

Большинство криптосистем решают только часть из этих проблем. Но в сложных криптографических протоколах существуют попытки обеспечения полной безопасности.

В зависимости от того к чему именно имеет доступ злоумышленник, выделяют несколько типов атак:

Атака с использованием только шифротекста. У криптоаналитика есть шифротексты нескольких сообщений, зашифрованных одним и тем же алгоритмом шифрования. Задача криптоаналитика состоит в раскрытии открытого текста этих сообщений или получении ключа (ключей), использованных для шифрования.

Вскрытие с использованием открытого текста. У криптоаналитика есть доступ не только к шифротекстам, но и открытым текстам сообщений, его задача состоит в нахождении ключа (или ключей), использованного для шифрования сообщений, для дешифрования других сообщений, зашифрованных тем же ключом (ключами).

В [26] приводятся примеры раскрытия открытого текста: почти каждая из шифрованных телеграмм в ближневосточные страны начиналась с перечисления многочисленных и всем известных регалий адресата, по которым вычислялось такое количество гаммы, которое иногда позволяло вскрывать шифр и читать телеграмму быстрее, чем она доходила до адресата. Или 1989 г., съезд народных депутатов. Генерал, стоя на трибуне со словами «Вот шифртелеграмма, которую я получил накануне!», показывает прямо в телекамеру содержание шифртелеграммы, тот самый открытый текст, по которому легко вычисляется гамма наложения.

Вскрытие с использованием выбранного открытого текста. У криптоаналитика есть доступ не только к шифротекстам и открытым текстам, но и возможность выбирать открытый текст, для шифрования. Это позволяет криптоаналитику выбирать для шифрования такие блоки, анализ которых даст больше информации о ключе.

Атаки этого рода срабатывали против немецких шифров: союзники сознательно допускали утечку определенной информации для того, чтобы получить зашифрованный текст, или провоцировали сообщения о событиях в городах с уникальными названиями, служащие особенно хорошими шпаргалками [10].

Вскрытие с использованием выбранного шифротекста. Криптоаналитик может выбирать различные шифротексты для дешифрования и имеет доступ к дешифрованным открытым текстам. Например, у криптоаналитика есть доступ к «черному ящику», выполняющему дешифрование. Его задача состоит в получении ключа.

Бандитский криптоанализ. Злоумышленник (слово криптоаналитик здесь не совсем уместно) угрожает, шантажирует или пытает пользова-

теля системы, пока не получит ключ. **Вскрытием с покупкой ключа** называют дачу взятки с целью получения ключа.

Так же требование выдать ключи может быть предъявлено в судебном порядке.

Существование этих способов взлома требует создания многосторонних протоколов, в которых ни один из пользователей не обладает всей ключевой информацией, либо использования стеганографии с целью сокрытия самого факта передачи какой либо секретной информации.

Задачи криптографии

Основными задачами криптографии и защиты информации в целом являются [21]:

- обеспечение конфиденциальности информации;
- обеспечение целостности информации;
- обеспечение достоверности информации;
- обеспечение оперативности доступа к информации;
- обеспечение юридической значимости информации, представленной в виде электронного документа;
- обеспечение неотслеживаемости действий клиента.

Конфиденциальность – свойство информации быть доступной только ограниченному кругу пользователей информационной системы, в которой циркулирует данная информация.

Целостность – свойство информации или программного обеспечения сохранять свою структуру и/или содержание в процессе передачи и хранения.

Аутентичность (Достоверность) – свойство информации, выражающееся в целостности и строгой принадлежности объекту, который является ее источником, либо тому объекту от которого эта информация принята.

Оперативность – способность информации или некоторого информационного ресурса быть доступным для конечного пользователя в соответствии с его временными потребностями.

Юридическая значимость – означает, что документ обладает юридической силой. С этой целью субъекты, нуждающиеся в подтверждении юридической значимости, договариваются о принятии некоторых атрибутов информации, выражающих ее способность быть юридически значимой. Например, о принятии электронной цифровой подписи.

Неотслеживаемость – способность совершать некоторые действия в информационной системе незаметно для других объектов, в том числе и администраторов. Цель данного требования – предотвращение тотальной слежки за пользователями ИС.

Конфиденциальность обеспечивается с помощью шифрования, целостность и аутентичность – с помощью ЭЦП и МАС (в зависимости от целей), юридическая значимость – с помощью ЭЦП. Примером решения задачи неотслеживаемости могут быть системы электронного голосования.

Электронной (цифровой) подписью (ЭЦП) называется присоединяемое к тексту его криптографическое преобразование, которое позволяет при получении текста другим пользователем проверить авторство и целостность сообщения. Для формирования ЭЦП используется закрытый ключ отправителя, а для проверки – открытый.

Имитозащита – защита от навязывания ложных данных. Для обеспечения имитозащиты к зашифрованным данным добавляется **имитовставка** (или MAC – Message Authentication Code – **код аутентичности сообщения**), представляющий собой последовательность данных фиксированной длины, полученную по определенному правилу из открытых данных и секретного ключа.

ТЕМА 3. МАТЕМАТИЧЕСКИЕ ОСНОВЫ КРИПТОЛОГИИ

Комбинаторика

Для того чтобы проводить элементарные вычисления, связанные со сложностью алгоритмов, длиной ключей, статистическим анализом, необходимо напомнить основы комбинаторики и теории вероятности. Теоремы комбинаторики и теории вероятности даны по [24].

Теорема о перемножении шансов. Основной принцип комбинаторики заключается в следующем: если первый элемент можно выбрать k-способами, а второй элемент – m-способами, то упорядоченную пару элементов можно составить km-способами.

Пусть множество $A=\{a_1,\ldots,a_k\}$ состоит из k элементов, а множество $B=\{b_1,\ldots,b_m\}$ – из m элементов. Тогда можно образовать ровно km пар (a_i,b_j), взяв первый элемент из множества A, а второй – из множества B.

Пример. Строка состоит из 6 символов. Первые 5 символов являются буквами английского алфавита, а последний – цифрой. Количество таких строк составит $26^5 \cdot 10$.

Общее количество различных наборов при выборе k элементов из n без возвращения и с учётом порядка равняется

$$A_n^k = n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot(n-k+1) = \frac{n!}{(n-k)!}.$$

Число A_n^k называется **числом размещений** из n элементов по k элементов, а сами результаты выбора – размещениями.

Пример. Если известно, что все символы ключа из 5 букв английского алфавита различны, то количество ключей составит $26 \cdot 25 \cdot 24 \cdot 23 \cdot 22 = \frac{26!}{21!}$.

В множестве из n элементов возможно ровно $n!$ перестановок этих элементов.

Пример. Количество вариантов записи всех символов алфавита без повторений равно $n!$, где n – длина алфавита.

Общее количество различных наборов при выборе k элементов из n без возвращения и без учёта порядка равняется

$$C_n^k = \frac{A_n^k}{k!} = \frac{n!}{k!(n-k)!}.$$

Число C_n^k называется **числом сочетаний** из n элементов по k элементов, а сами результаты выбора — сочетаниями.

Пример. В шифрах многозначной замены символы открытого текста заменяются произвольно одной из множества возможных шифрозамен. Если общее количество шифрозамен равно n, а количество шифрозамен для некоторого символа равно k, то количество вариантов выбора множества шифрозамен равно C_n^k .

Общее количество различных наборов при выборе k элементов из n с возвращением и без учёта порядка (число сочетаний с повторениями) равняется

$$C_{n+k-1}^k = C_{n+k-1}^{n-1}.$$

Число C_{n+k-1}^k называется **числом сочетаний** из n элементов по k **с повторениями**.

Пример. В шифре многозначной замены общее количество шифрозамен равно n, количество символов исходного алфавита равно k.

Количество вариантов составления множеств замен различного размера составит C_{n+k-1}^k. Речь идет только о размере этих множеств, без различия элементов этих множеств.

Теория вероятности

Пространством элементарных исходов называется множество Ω, содержащее все возможные взаимоисключающие результаты данного эксперимента. Элементы множества Ω называются элементарными исходами и обозначаются ω.

Событиями называются подмножества множества Ω. Говорят, что произошло событие A, если эксперимент завершился одним из элементарных исходов, входящих в множество A.

Элементарный исход – это мельчайший неделимый результат эксперимента, а событие может состоять из одного или нескольких исходов.

Объединением $A \cup B$ событий A и B называется событие, состоящее в том, что из двух событий A и B случилось хотя бы одно. Это событие включает как элементарные исходы из множества A, так и элементарные исходы из множества B.

Пересечением $A \cap B$ событий A и B называется событие, состоящее в том, что произошли сразу оба события A и B. Это событие содержит элементарные исходы, каждый из которых принадлежит и множеству A, и множеству B. Вместо $A \cap B$ часто пишут просто AB.

Дополнением $A \setminus B$ события B до A называется событие, состоящее в том, что произошло A, но не произошло B. Событие $A \setminus B$ содержит элементарные исходы, входящие в множество A, но не входящие в B.

Противоположным (или дополнительным) к событию A называется событие $\overline{A} = \Omega \setminus A$, состоящее в том, что A не произошло. Событие A состоит из элементарных исходов, не входящих в множество A.

События A и B называются несовместными, если они не могут произойти одновременно: $A \setminus B = \varnothing$.

Говорят, что событие A влечёт событие B, и пишут $A \subseteq B$, если всегда, как только происходит событие A, происходит и событие B. Это означает, что любой элементарный исход, входящий во множество A, одновременно входит в множество B, т. е. A содержится в B.

Пространство элементарных исходов назовём дискретным, если множество конечно или счётно: $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n, \ldots\}$.

В криптографии мы будем встречаться в основном с дискретными множествами.

Сопоставим каждому элементарному исходу ω_i число $p_i \in [0,1]$ так, чтобы $\sum_{\omega_i \in \Omega} p_i = 1$.

Вероятностью события A называется число $P(A) = \sum_{\omega_i \in \Omega} p_i$, равное сумме вероятностей элементарных исходов, входящих во множество A. В случае $A = \varnothing$, $P(A) = 0$.

Если все элементарные исходы равновероятны, то $P(A) = \frac{|A|}{|\Omega|}$, где $|A|$ – количество элементов (элементарных исходов) конечного множества

A. Вероятность события A равна отношению числа исходов, благоприятствующих этому событию, к общему числу **равновозможных** исходов.

В криптографии, при работе с числами, часто существует возможность выделить такие события, которые являются равновероятными.

Свойства вероятности:

1. $P(\varnothing) = 0$;

2. Для любого конечного набора попарно несовместных событий A_1, . . . , A_n имеет место равенство:

$$P(A_1 \cup \ldots \cup A_n) = P(A_1) + \ldots + P(A_n).$$

3. $P(\overline{A}) = 1 - P(A)$.

4. Если $A \subseteq B$, то $P(B \setminus A) = P(B) - P(A)$.

5. Если $A \subseteq B$, то $P(A) \le P(B)$.

6. $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

7. $P(A_1 \cup \ldots \cup A_n) \le \sum_{i=1}^{n} P(A_i)$

8. $P(A_1 \cup \ldots \cup A_n) = \sum_{i=1}^{n} P(A_i) - \sum_{i<j}^{n} P(A_i A_j) + \sum_{i<j<m}^{n} P(A_i A_j A_m) - \ldots$

$+ (-1)^{n-1} P(A_1 A_2 \ldots A_n)$.

Условной вероятностью события A при условии, что произошло событие B, называется число

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}.$$

События A и B называются независимыми, если $P(A \cap B) = P(A)P(B)$.

Вероятность события A при условии что произошло событие B остается такой же, как в случае если событие B не произошло.

При вычислении вероятности можно с помощью комбинаторики посчитать общее количество равновероятных событий и количество равновероятных событий удовлетворяющих заданным условиям, но обычно после первичных вычислений удобнее работать с вероятностями.

Примеры.

1. Посчитать вероятность того, что в строке длиной n символов символ *«A»* не появится ни разу (считая вероятности появления букв равными).

Пусть события $A_1,\ldots,A_n$ – появление символа *«A»* на соответствующих позициях.

$$P(A_1) = \ldots = P(A_n) = P(A) = p = 1/26.$$

P – вероятность непоявления символа *«A»* ни разу.

Вероятность непоявления символа на каждой позиции

$$q = P(\overline{A}) = 1 - p = 25/26.$$

Так как появление или не появление символа на каждой позиции – события независимые, то $P = q^n = \left(\frac{25}{26}\right)^n$.

2. Посчитать вероятность появления символа только на 1-й и 2-й позиции строки из 10 символов.

Оставив те же обозначения, и учитывая независимость событий, событие A должно произойти 2 раза и не произойти 8 раз, в заданном порядке.

$$P = p^2 q^8.$$

3. Посчитать вероятность появления символа k раз в строке из n символов.

Если позиции символов заданы, то $P = p^k(1-p)^{n-k}$. Количество равновозможных расположений этих символов определяется числом сочетаний из n *по* k. Таким образом:

$$P_n(k) = C_n^k p^k (1-p)^{n-k}.$$

Полученную формулу называют «k удач из n попыток» или формулой Бернулли; а распределение биномиальным.

Локальная теорема Лапласа позволяет вычислить значение вероятности для биномиального распределения приближенно:

$$P_n(k) \approx \frac{1}{\sqrt{2\pi npq}} e^{\frac{-t^2}{2}}, t = \frac{k-np}{\sqrt{npq}}.$$

Интегральная теорема Лапласа позволяет вычислить вероятность попадания k в диапазон:

$$P(k_1 \le k \le k_2) \approx \Phi(x_2) - \Phi(x_1);$$

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{\frac{-t^2}{2}} dt,\ x_1 = \frac{k_1 - np}{\sqrt{npq}}, x_2 = \frac{k_2 - np}{\sqrt{npq}}.$$

Функция $\Phi(x)$ – является табличной (в Excel значения вероятностей могут быть вычислены с помощью функции «БИНОМРАСП»).

Арифметика остатков

Многие операции в криптографии выполняются по модулю большого целого числа. В простых случаях это степень числа 2, используемая для получения последних бит. В других, модулем является большое простое число и на арифметике остатков, строится безопасность криптографических систем (особенно ассиметричных).

Остаток от деления обозначается *«mod»* и может использоваться два основных варианта записи:

$b = a \bmod m$ или $a = b \pmod m$.

Означает, что остаток от деления a на m равен b или $a = b + km$, для некоторого целого k. b называют **вычетом** a по модулю m, а называют

конгруэнтным b по модулю m. Операцию вычисления остатка также называют **приведением** по модулю m.

Арифметика остатков подчиняется тем же законам, что и обычная арифметика она коммутативна, ассоциативна и дистрибутивна. Кроме того, приведение каждого промежуточного результата по модулю n дает тот же результат, как и выполнение всего вычисления с последующим приведением конечного результата [11]:

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

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

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

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

Важное место в арифметике остатков занимает понятие обратного числа. Как и в обычной арифметике, обратным для числа a называется число a^{-1}, такое, что $a \cdot a^{-1} = 1 \ (mod\ m)$. Например, для решения уравнения $a \cdot x = b \ (mod\ m)$ относительно x, необходимо найти a^{-1}. $x = b \cdot a^{-1} \ (mod\ m)$. Кроме того, не для любого числа может быть найдено обратное. Например, для числа 3 и модуля 5 обратным является 2, так как $3 \cdot 2 \ mod\ 5 = 6 \ mod\ 5 = 1$, а для числа 3 по модулю 6 обратного не существует. В общем случае, для уравнения $a^{-1} = x \ (mod\ m)$ существует единственное решение, если a и m взаимно простые числа. Если m – простое, то все числа в диапазоне от 1 до n–1 имеют обратное по модулю m.

Обратные числа можно находить с помощью расширенного алгоритма Эвклида или с помощью обобщения Эйлера для малой теоремы Ферма.

Если m – простое и a не кратно m, то малая теорема Ферма утверждает:

$$a^{m-1} = 1 \ (mod\ m)$$

Функция Эйлера $\varphi(n)$ определяет количество чисел меньших n взаимно простых с n. Если n – простое, то $\varphi(n) = n - 1$. Если n является произведением чисел p и q, то $\varphi(n) = (p - 1)(q - 1)$. Для числа $n = p^k q^l$, $\varphi(n) = p^{k-1}(p - 1) \cdot q^{l-1}(q - 1)$.

Обобщением теоремы Ферма является утверждение, что

$$a^{\varphi(m)} = 1 \ (mod\ m).$$

Из этого выражения можно получить два полезных следствия:

$a^{\varphi(m)-1} = a^{-1} \ (mod\ m)$ – позволяет найти обратное, если известно разложение числа на множители.

$a^{\varphi(m)+1} = a^{k\varphi(m)+1} = a \ (mod\ m)$ – используется в схеме RSA.

Расширенный алгоритм Эвклида позволяет найти такие целые числа x и y, что $ax + by = d$, где a и b – целые неотрицательные числа и $a > b$, $d = НОД$(a,b) [25].

Применительно к задаче нахождения обратных чиселa – является модулем, b – числом для которого необходимо найти обратное, $y = b^{-1} \ (mod \ a)$, а НОД(a,b) должен быть равен единице, так как только для таких a и b может быть найдено обратное.

Алгоритм может быть представлен следующей последовательностью шагов:

```
Вход: a и b, a>b
1. Положить x2:=1, x1:=0, y2:=0, y1:=1
2. Пока b>0
     q:=[a/b], r:=a-qb, x:=x2-qx1, y:=y2-qy1
     a:=b, b:=r, x2:=x1, x1:=x, y2:=y1, y1:=y
3. Присвоить d:=a, x:=x2, y:=y2
```

Алгоритм может выдавать отрицательное значение y и положительное x. В этом случае можно заменить y на $a+y$, а x на $–b+x$, так как $a(–b + x) + b(a + y) = –ab + ax + ab + bx = ax + by$.

```
4. Если y<0, то y :=a+y, x :=-b+x
Выход: d,x,y.
```

Если требуется найти обратное для 3 по модулю 5, то на вход подается a = 5, b = 3; выходом являются d = 1, x = –1, y = 2. Таким образом, обратное для 3 по модулю 5 равно 2.

Количество выполняемых делений в этом алгоритме равно [11]

$$0{,}843 * log_2 n + 1{,}47.$$

Возведение в большие степени больших чисел по модулю, требует работы с числами значительно меньшего размера, чем в обычной арифметике. Если длины основания и степени сопоставимы с 2^{100}, то для получения результата требуется в среднем 150 умножений. Но в случае арифметики остатков (с аналогичной длиной модуля) промежуточные и конечный результаты будут того же порядка что и модуль, а в случае обычной арифметики результат будет сопоставим с $2^{100 \cdot 2^{100}} \approx 2^{2^{107}}$. Записать число такой длины не представляется возможным, так как оно превышает число атомов на земле (хотя и меньше числа атомов вселенной).

Для того чтобы возведение в степень было быстрым, и не требовало 2^{100} умножений, необходимо выполнять действия в правильной последовательности.

Например, для вычисления a^8 не обязательно вычислять $a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a \cdot a$, достаточно вычислить $((a^2)^2)^2$. А для возведения числа в степень 11 (11 = 1011_2) необходимо вычислить $a^{11} = ((((a^2)^2) \cdot a)^2) \cdot a$, то есть выполнить 5 операций умножения.

Такой способ позволяет сократить вычисление a^k до 1,5 n операций, где n – число двоичных знаков. Более сложные способы, с сохранением большого количества промежуточных результатов позволяют сократить число операций до 1,1 n. Способ выполнения операций одинаков, как для арифметики остатков, так и для обычной, но в арифметике остатков после каждой операции выполняется приведение по модулю. Для выполнения обратной операции: вычисления дискретного логарифма эффективных способов не найдено.

Рекомендуемая литература

Комбинаторика и теория вероятности: [24] или любой другой учебник по теории вероятности и математической статистике.

Арифметика остатков, теория чисел и другие вопросы математических основ криптографии – [17, 18, 20, 25].

Контрольные вопросы

1. Объясните получение формул комбинаторики
2. Ключ состоит из 3 букв и 3 цифр. Сколько существует таких ключей?
3. Ключ состоит из 3 букв в верхнем регистре, 3 букв в нижнем регистре, 2 цифр, знака подчеркивания и начинается с буквы. Сколько существует таких ключей.
4. Строка состоит из 5 букв. Какое существует количество строк содержащих хотя бы один символ *«A»*? Ровно один?

ТЕМА 4. АБСОЛЮТНАЯ И ВЫЧИСЛИТЕЛЬНАЯ СТОЙКОСТЬ

Теоретико-информационная стойкость

Система называется «абсолютно стойкой», если при бесконечной вычислительной мощности, и независимо от объемов шифротекстов у криптоаналитика информации для получения открытого текста недостаточно [8].

Пусть

M – множество открытых сообщений, X – сообщение;

C – множество шифротекстов, Y – шифротекст;

K – множество ключей, Z – ключ.

$Y = E_z(X);\ X = D_z(Y).$

Вероятность появления определенного значения x – $p(X = x)$. Будем считать, что события появления открытого текста X и ключа K – независимы.

Выразим вероятность появления шифротекста y, через вероятности появления открытых текстов и ключей.

$$p(Y = y) = \sum_{z \in K} p(Z = z) \cdot p(X = D_z(y)).$$

Вероятность появления определенного шифротекста с равна сумме произведений вероятности шифрования каждым ключом z на вероятность того, что при шифровании использовался открытый текст $D_z(y)$.

Открытый текст x в общем случае может быть преобразован в шифротекст у с помощью различных ключей z. Если можно построить таблицу соответствия ключей открытым текстам и шифротекстам, то

$$p(Y = y \mid X = x) = \sum_{z: x = D_z y} p(Z = z).$$

Вероятность появления шифротекста y, при условии, что был зашифрован открытый текст x, равна сумме вероятностей появления всех ключей, с помощью которых можно расшифровать y в x.

Для того чтобы вскрыть шифр, криптоаналитику понадобятся вероятности вида $p(x = X|y = Y)$, так, как ему известен шифротекст и он хочет узнать, из какого открытого текста он был получен.

$$p(X = x \mid Y = y) = \frac{p(X = x) \cdot p(Y = y \mid X = x)}{p(Y = y)}.$$

Криптографическая система обладает абсолютной стойкостью, если для всех открытых текстов $x \in M$ и всех шифротекстов $y \in C$ выполняется равенство

$$p(X = x|Y = y) = p(X = x).$$

или

$$p(Y = y|X = x) = p(Y = y).$$

Таким образом, наличие шифротекста (или сколь угодно большого количества шифротекстов) и бесконечных вычислительных мощностей никак не повлияет на знание об открытом тексте. Если криптоаналитик достоверно знает часть открытого текста, то он не сможет узнать ни одного нового бита открытого текста.

Из этого определения есть важное следствие. Мощности множеств M, C и K удовлетворяют

$$|K| \geq |C| \geq |M|.$$

Очевидно, что $|C| \geq |M|$. Это неравенство выполняется для любых систем, не обязательно абсолютно стойких, так как для любого ключа z: $y = E_z(x)$ и $|C(z)| = |M|$. А множество всех шифротекстов $|C|$ – не менее множества всех шифротекстов, полученных с помощью ключа z: $|C| \geq |C(z)| = |M|$.

Для совершенной системы, из $p(Y = y|X = x) = p(Y = y) > 0$ следует, что для любой пары x, y существует хотя бы один ключ z (может быть множество), такой, что $x = D_z(y)$. Зафиксировав x, мы можем сопоставить ка-

ждому y хотя бы одно z и для различных y – ключи z будут различными. Следовательно, $|K| \geq |C|$.

В частном случае, криптосистема, в которой $|K| = |C| = |M|$ является абсолютно стойкой, тогда и только тогда, когда:

- использование всех ключей равновероятно $p(Z = z) = 1/|K|$;
- для каждой пары x, y существует единственный ключ z такой, что $y = Ez(x)$.

Шифр, удовлетворяющий этим условиям, существует и называется одноразовый блокнот, но в связи с условием $|K| = |C| = |M|$ его использование затруднено и возможно только для коротких сообщений. Для больших сообщений его использование затруднительно, так как необходимо передавать ключ такой же длины, как и шифротекст. И хотя бы одно из двух: ключ или сообщение должны быть переданы по безопасному каналу.

На практике вместо абсолютно стойких шифров используются вычислительно стойкие или вычислительно безопасные, для которых невозможно построение вероятностей $p(Y = y \mid X = x) = \sum_{z: x=D_z y} p(Z = z)$ из-за большого количества требуемых вычислений.

Теория сложности и безопасность алгоритмов

Различные алгоритмы предоставляют различные степени безопасности, в зависимости от того, насколько трудно взломать алгоритм [11].

Алгоритм считается **вычислительно безопасным** (или **сильным**), если он не может быть взломан с использованием доступных ресурсов сейчас или в будущем. Обычно вычислительную сложность оценивают, как время необходимое для вскрытия (порядок количества операций), и для некоторых алгоритмов, требованиями к объему памяти.

Так же может быть оценена стоимость данных и стоимость вскрытия. Если стоимость данных меньше стоимости вскрытия, то данные можно считать защищенными. Если время взлома алгоритма больше времени в течение, которого данные должны храниться в секрете, то алгоритм можно считать безопасным. Например, для хранения личных данных на домашнем компьютере совсем не нужно использовать тройные шифры с ключами по 128 бит каждый. В случае кражи компьютера никто не станет тратить годы на вскрытие шифров. Но при повседневном использовании замедляются процессы чтения и записи. А при хранении государственных тайн и через сто лет открытие некоторых документов может повлечь международный скандал.

Обычно вычислительная сложность описывается порядком величины количества операций и записывается в виде $O(f(n))$, где $f(n)$ – сте-

пенная, логарифмическая или экспоненциальная функция от объема входных данных.

Например, рассмотрим временную сложность вычисления для полинома [28]

$$P_n(x) = a_n x^n + ... + a_i x^i + ... + a_2 x^2 + a_1 x + a_0.$$

При прямом вычислении для каждого слагаемого требуется i произведений, и после n сложений. Количество операций составит

$$\sum_{i=0}^{n} i + n = \frac{n(n+1)}{2} + n = \frac{n^2}{2} + \frac{3n}{2} + \frac{1}{2}.$$

Вычислительная сложность для такого количества операций определяется главной степенью полинома и записывается как $O(n^2)$.

Если при вычислении этого же полинома его записать в виде $P_n(x) = a_0 + x(a_1 + x(a_2 + ...(a_i + .. x(a_{n-1} + a_n x)))$, то на каждую скобку требуется одно сложение и одно умножение. Всего $2n$ операций. Вычислительная сложность в этом случае записывается как $O(n)$.

Если для вскрытия шифра требуется полный перебор ключей, и в алгоритме нет других уязвимостей, то сложность такого перебора $O(2^n)$, где n – длина ключа. При этом сложность проверки каждого ключа значительно меньше и может не учитываться.

Алгоритм называют **постоянным**, если его сложность не зависит отn. Записывается $O(1)$. Алгоритм является **линейным,** если его временная сложность $O(n)$. Алгоритм является **полиномиальным**, если его сложность $O(n^m)$, также выделяют квадратичные и кубичные. Алгоритм является **экспоненциальным**, если его сложность равна $O(t^{f(n)})$, где $t > 1$, $f(n)$ – полиномиальная функция. Если $f(n)$ возрастает, но медленнее, чем линейная, то алгоритм называется **суперполиномиальным** [11].

Проблемы, которые можно решить за полиномиальное время, называют решаемыми. Обычно для разумного объема входных данных, они могут быть решены за разумное время. Проблемы, которые не могут быть решены за полиномиальное время, называют нерешаемыми, или трудными.

Существуют так же принципиально неразрешимые проблемы, алгоритмов решения которых не найдено или не существует.

Рассмотрим сложность решения проблемы полного перебора ключей. Пусть ключом является бинарная строка длиной n бит, и любая строка может быть ключом. Тогда количество ключей равно 2^n. Перебор 2^m ключей соответствует перебору m бит ключа.

Если один процессор может проверять 1 000 000 ключей в секунду, то это соответствует перебору 20 бит. Если для перебора можно потратить год, то может быть проверено в 2^{25} раз больше ключей, чем за секунду. За 100 лет может быть проверено в 2^7 раз больше ключей, чем за год. Если можно задействовать одновременно 1 000 000 машин, то перебор будет произведен в 2^{20} раз быстрее. Если считать, что каждые

два года мощность машин удваивается, то может быть через 100 лет, за секунду на одной машине можно будет перебирать до 2^{70} ключей.

Исходя из этих цифр, можно сделать выводы, что длины ключа в 128 бит достаточно, если информация не составляет государственную тайну и на вскрытие шифра не будут направлены миллионы машин и затрачены годы. А длины ключа в 256 бит должно хватить и на ближайшие 100 лет.

Соответствие n бит – 2^n ключей, справедливо, только если любая битовая последовательность может быть ключом. Это соответствие не выполняется для ассиметричных алгоритмов, поэтому для обеспечения аналогичного уровня безопасности требуются ключи длиной 1024 или 2048 бит.

Также может быть рассмотрено требование к памяти, необходимой для тех или иных вычислений. Обычно ее сравнивают с числом атомов во вселенной. Для некоторых алгоритмов вскрытия, требование к памяти очень быстро превышает это число.

Если по прошествии времени в алгоритме обнаружится уязвимость, то возможно он будет вскрыт сразу и без использования огромных вычислительных мощностей, либо с большим объемом вычислений, но значительно меньшим, чем необходимо для полного перебора. Для некоторых алгоритмов можно предположить какие открытия в математике позволят упростить вскрытие, и на сколько. При проектировании и выборе алгоритмов, рекомендуется закладывать двойной запас прочности. Если для перебора недоступно 128 бит, то желательно использовать длину ключа в 256.

Также необходимо сказать о классах сложности проблем.

Класс **P** состоит из всех проблем, которые могут быть решены за полиномиальное время. Например, задача проверки одного ключа.

Класс **NP** состоит из всех проблем, которые можно решить за полиномиальное время на недетерминированной машине Тьюринга, которая может делать предположения и угадывать решение или перебирать все возможные решения. Например, задача перебора всех ключей решается за полиномиальное время, если все возможные ключи проверяются одновременно.

Предполагается, что квантовые компьютеры смогут проверять все предположения одновременно, то есть решать NP-полные задачи за полиномиальное время.

Контрольные вопросы

1. Пусть размер множества открытых текстов равен M, ключа – N, пара открытый текст, ключ однозначно определяет шифротекст. Какое количество шифротекстов можно получить для заданного открытого тек-

ста? Для всех открытых текстов? Какое количество шифротекстов, для всех сообщений обеспечит большую безопасность?

2. При каких условиях два различных ключа переводят открытый текст в один и тот же шифротекст?

3. Приведите примеры линейных, полиномиальных и экспоненциальных алгоритмов.

ТЕМА 5. ШИФРЫ ЗАМЕНЫ

Общие сведения

Шифром замены (или **подстановки**) называется алгоритм шифрования, который производит замену каждого символа (блока) открытого текста на символ (блок) шифротекста.

Символы открытого текста и шифротекста в общем случае могут принадлежать разным алфавитам, например, буквы могут быть заменены числами. В качестве символов могут рассматриваться так же сочетания букв или блоки информации, например, в целях криптоанализа блочные шифры могут рассматриваться как шифр замены, с очень большой таблицей замен.

Шифром моноалфавитной замены (или **простой замены**) называют шифр замены, в котором замена каждого символа осуществляется по одному и тому же правилу, независимо от его положения в тексте.

Шифром многоалфавитной замены называют шифр замены, в котором правило замены символов, может варьироваться в зависимости от положения в тексте, предшествующих символов или др. параметров.

Шифром многозначной замены называют шифр замены, в котором каждому символу из алфавита открытого сообщения сопоставлено множество возможных замен. При шифровании замена может выбираться произвольно.

Пусть шифруется сообщение T, результатом шифрования является шифрограмма C. Алфавит открытого сообщения A состоит из символов $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$; n – количество символов в алфавите открытого сообщения.

Формально в этом случае шифр замены можно описать следующим образом. Для каждой буквы α исходного алфавита строится некоторое множество символов $M(\alpha)$ так, что множества $M(\alpha)$ и $M(\beta)$ попарно не пересекаются при $\alpha \neq \beta$, то есть любые два различные множества не содержат одинаковых элементов. Множество $M(\alpha)$ называется множеством шифробозначений для буквы α [23].

Преобразование π символов открытого текста T в символы шифротекста C называют подстановкой.

Таблицу соответствия символов открытого текста символам шифрограммы называют таблицей замещения, или алфавитом подстановки. Такая таблица является ключом шифра замены.

Таблица 1

Общий вид таблицы замен

Символ открытого текста: a_i	A_1	A_2	...	A_N
Множество символов шифротекста: $M(A_1)$	$M(A_1)$	$M(A_2)$	...	$M(A_3)$

Расшифрование выполняется с помощью обратной подстановки, которая также может быть записана таблицей. Если символ шифротекста входит в $M(\alpha_i)$, то он заменяется на α_i.

Если процедура шифрования предусматривает последовательное выполнение нескольких подстановок, то результирующее преобразование так же является подстановкой.

$$\pi = \pi_1\pi_2.$$

Подстановки обладают свойством ассоциативности: Результат выполнения трех и более подстановок не зависит от порядка расстановки скобок [17]:

$$(\pi_1\pi_2)\pi_3 = \pi_1(\pi_2\pi_3).$$

Необходимо понимать, что последовательное использование нескольких подстановок не может увеличить сложность вскрытия, так как в итоге выполняется одна подстановка.

В современных блочных шифрах также используются таблицы подстановок и являются основным нелинейным элементом. Они обычно обозначаются S-блоки (от substitute).

Шифры простой замены

В **шифре Цезаря** каждая буква сообщения заменяется на другую, отстоящую от нее в алфавите на определенное число позиций. В оригинальном шифре Цезаря число позиций равнялось 3 [14], то есть можно говорить, что ключ отсутствовал, а замена символов открытого текста на символы шифротекста осуществлялась в соответствии с фиксированной таблицей (рис. 4).

T_i	A	B	C	D	E	F	G	H	I J	K		L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
C_i	D	E	F	G	H	I J	K		L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Рис. 4. Шифр Цезаря

Вскрытие такого шифра не представляло большого труда. Тем не менее вариации с различными сдвигами применялись и позднее.

Другой шифр, **полибианский квадрат**, авторство которого приписывается греческому писателю Полибию, использует произвольно заполненную таблицу замен. Для греческого алфавита размер таблицы составляет 5×5 (рис. 5). Каждая буква исходного текста заменяется на букву, стоящую в квадрате снизу от нее.

Ε	Ρ	Τ	Υ	Θ
Ι	Ο	Π	Α	Σ
Δ	Φ	_	Η	Ξ
Κ	Λ	Ζ	Χ	Ψ
Ω	Β	Ν	Μ	Γ

Рис. 5. Пример заполнения полибианского квадрата греческим алфавитом

Этот шифр для расшифрования требует знания не только способа шифрования (алгоритма), но и таблицы (ключа).

Стремление к упрощению привело к созданию уже в средние века шифрующей **системы Трисемуса**. В прямоугольную таблицу в начале записывалось некоторое слово или короткая фраза без повторений букв, а остальные клетки дополнялись алфавитом по порядку. Это позволяло использовать в качестве ключа не сложную таблицу, а только слово, которое могло быть передано устно (рис. 6).

T	R	I	S	E	M	U
A	B	L	C	D	F	G
H	J	K	N	O	P	Q
V	W	X	Y	Z	_	.

Рис. 6. Система Трисемуса

Пример:

Открытый текст: “SECRET_MESSAGE_TEXT”.

Ключ: “TRISEMUSTABLE”; размер таблицы: 7x4.

Шифротекст: “CDNBDAMFDCCHQDMADIA”.

В общем случае, максимальное число возможных вариантов подстановок соответствует числу перестановок из n элементов, то есть $n!$, где n – длина алфавита.

Для латинского алфавита из 26 букв количество ключей составляет $26! \sim 2^{88}$. То есть количество ключей для шифров замены с алфавитом, состоящим только из букв, превосходит количество ключей для распространенных до недавнего времени 64-битных блочных шифров.

В шифрах однозначной замены для одного и того же символа в открытом сообщении всегда записывается один и тот же символ шифротекста.

При шифровании текста на естественном языке между символами открытого текста существуют зависимости, которые остаются без изменений в шифротексте.

К таким зависимостям можно отнести известное статистическое распределение символов.

В табл. 2 буквам русского языка сопоставлена частота их появления в текстах [14].

Таблица 2

Частоты появления букв русского языка в текстах

Буква (символ)	Частота	Буква	Частота	Буква	Частота	Буква	Частота
ПРОБЕЛ	0.146	Р	0.042	Я	0.017	Ж	0.007
О	0.094	Л	0.039	З	0.016	Ш	0.006
Е	0.071	В	0.038	Ы	0.015	Ц, Ю	0.005
А	0.069	К	0.029	Г	0.014	Щ	0.004
И	0.064	М	0.027	Ь, Б	0.013	Ф	0.003
Н	0.057	П	0.026	Ч	0.012	Э	0.002
Т	0.054	Д	0.024	Й	0.010	Ъ	0.001
С	0.046	У	0.023	Х	0.008		

Существование устойчивых, часто повторяющихся сочетаний букв и сочетаний, которые не могут встречаться в тексте без ошибок.

Первую проблему успешно устраняют шифры многозначной замены. Обе проблемы устраняются шифрами многоалфавитной замены.

Шифры многозначной замены. Система омофонов

В системе омофонов, каждый символ исходного сообщения заменяется одной случайной шифрозаменой из множества возможных для этого символа. Количество шифрозамен выбирается пропорционально частоте появления символа в открытом тексте. В этом случае частота появления шифрозамен в шифротексте будет одинаковой.

Так, если взять алфавит замен состоящим из 1000 символов вида 000 – 999, то в соответствии с таблицей пробелу будет поставлено в

соответствие 1000*0,146 = 146 шифрозамен; букве О – 94, Е – 71,
Ъ – 1 шифрозамена.

Пример:

Открытый текст: "Шифр многозначной замены".

Ключ: ключом является таблица на рис. 7, приведенная в сокращенном виде. Количество строк по разным столбцам должно отличаться и соответствовать указанному в строке «всего».

	_	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш			...	Я	
1	567	257	421	024	358	514	782	756	746	146	895	561	345	450	740	221	80	36	116	561											
2	127	143	743	428	642	761	351	658	654	325	631	112	678	041	736	816	36	41	411	149											
3	869	863	796	324	118	834	419	324	359	547	482	647	124	320	910	642	14	24	175	925											
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
Всего	146	69	13	38	14	24	71	7	16	64	10	29	39	27	57	94	26	42	46	54	23	3	8	5	12	6	1				

Рис. 7. Система Омофонов

Шифротекст: 941325400 175869041 910221118 816359736 143004736 642631127 654869320 782740 …

Для одного и того же открытого текста при использовании одного и того же ключа может быть составлено множество различных открытых текстов. Такое свойство шифров так же затрудняет вскрытие.

Для шифров многозначной замены простой подсчет частот уже ничего не дает криптоаналитику. Однако если текст достаточно большой, то несколько помочь может информация о распределении пар и троек символов.

Шифры многоалфавитной замены

В многоалфавитных шифрах замены для шифрования используется несколько различных подстановок. Выбор подстановки осуществляется по определенному правилу. В простейших случаях – это циклическая смена предварительно определенных алфавитов; в более сложных – алфавиты могут быть вообще не определены в начале шифрования, а формироваться динамически.

Система шифрования Вижинера

Система шифрования Вижинера является логическим расширением простейшего шифра Цезаря на многоалфавитные шифры замены. В то же время, по сути, он является шифром гаммирования и при правильном применении может быть совершенным (см. тему 7. Аддитивные шифры).

Система шифрования Вижинера может быть представлена таблицей, в которой в первой строке записывается алфавит, а в каждой последующей алфавит со сдвигом на один символ [14].

Для шифрования выбирается ключ: слово, фраза или текст.

Для шифрования i-го символа, по i-му символу открытого текста в первой строке определяется столбец, а по i-му символу ключа в крайнем левом столбце – строка. На пересечении строки и столбца будет находиться i-й символ шифротекста. Если длина ключа меньше сообщения, то он используется повторно.

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	**Т**	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А
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	Ё	Ж	З	И	Й
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	Ё	Ж	З	И	Й	К
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	Ё	Ж	З	И	Й	К	Л
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	Ё	Ж	З	И	Й	К	Л	М
О	П	**Р**	**С**	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	**Ъ**	Ы	Ь	Э	Ю	**Я**	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П
С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	**Э**	**Ю**	**Я**	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р
Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	**А**	Б	В	Г	Д	Е	Ё	**Ж**	З	И	Й	К	Л	М	**Н**	О	П	Р	С
Ю	Я	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
Я	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю

Рис. 8. Система Вижинера

В данном случае для шифрования используется 33 заранее определенных алфавита, порядок их применения определяется ключом. Если длина ключа меньше длины сообщения, то шифр является циклическим (повторяется порядок смены алфавитов).

Пример:

Ключ: "строка".

Открытый текст: "многоалфавитный".

Шифротекст: "юаясщаэжррутянъ". Использованные символы выделены в таблице на рис. 8 жирным.

Если m – длина ключа, k – номер цикла, b – номер символа в цикле, то для каждого символа с индексом km+b применяется одинаковый алфавит подстановки, и для этих символов может быть проведен статистический анализ. Учитывая то, что алфавиты заранее известны, достаточно найти только самые часто встречающиеся символы, что бы восстановить алфавит для группы символов.

Роторные машины

Вплоть до середины XX в., для шифрования использовались роторные машины. Среди них – американская машина SIGABA (M-134), английская TYPEX, немецкая ENIGMA, японская PURPLE.

Главной деталью роторной машины является ротор (или колесо) с проволочными перемычками внутри. Ротор имеет форму диска. На каж-

дой стороне диска расположены равномерно по окружности 26 электрических контактов. Каждый контакт на передней стороне диска соединен с одним из контактов на задней стороне, как показано на рисунке. В результате электрический сигнал, представляющий знак, будет преобразован в соответствии с тем, как он проходит через ротор от передней стороны к задней (рис. 9).

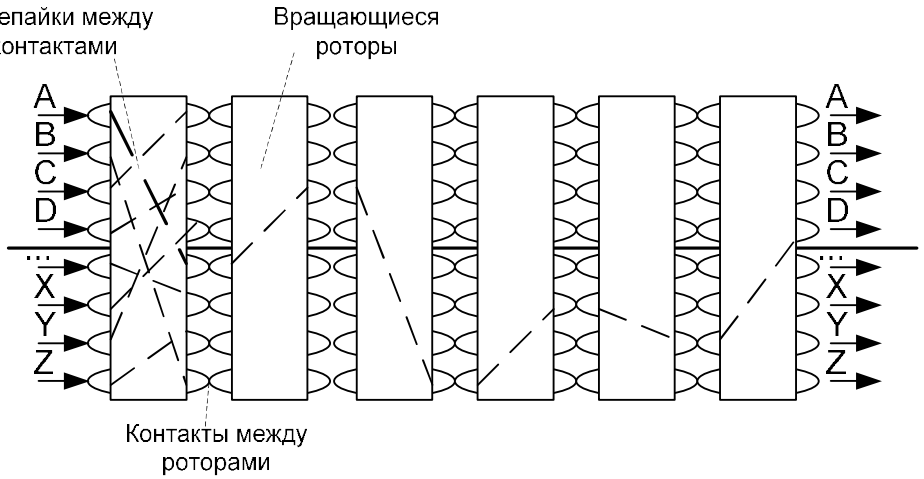


Рис. 9. Схема роторной машины

При повороте ротора из одного положения в другое изменяется осуществляемая машиной подстановка.

В шифровальных системах такого типа два вида ключей: постоянные и переменные. Постоянные – это перепайки между контактами ротора, они определяют алфавиты подстановки. Изменение этих ключей означает необходимость создания новых роторов и в реальных условиях эти ключи не изменялись. Без перехвата этих постоянных ключей, без ЭВМ вскрыть шифр было невозможно. Количество вариантов таких ключей для n роторов – $(26!)^n$. Переменные (временные, разовые) – определяют набор роторов, их количество и начальное положение. Количество разовых ключей при известном порядке роторов – 26^n.

Также неизвестным может являться алгоритм движения роторов, но обычно он определяется типом шифровальной машины и не может быть произвольно изменен. Так в немецкой шифровальной машине Enigma после шифрования одного знака правое крайнее колесо поворачивается на одну позицию. Когда это (и любое другое) колесо совершит полный оборот, колесо, расположенное слева от него, передвинется на одну позицию, и процесс будет повторяться. Этот процесс проведет банк роторов сквозь все его возможные положения, прежде чем цикл повторится. Период n-роторной машины составляет 26^n.

В истории уже приводился пример перехвата постоянных ключей машины Enigma. Их наличие позволяло вскрыть систему шифрования.

Поэтому надежность криптографической системы определяется в основном временными (сеансовыми) ключами. Но были также и ошибки в распределении сеансовых ключей. В начале каждого дня сеансовый ключ шифровался и передавался в начале сообщения и оставался неизменным в течение дня. Поэтому машину Enigma достаточно было вскрыть один раз, что бы читать сообщения до смены алгоритма распределения ключей.

Подробнее описание машины Enigma, ошибки в ее использовании и теоретические способы вскрытия можно прочитать в [8].

Контрольные вопросы

1. Сколько ключей, дающих различный результат шифрования, возможно при последовательном выполнении двух простых подстановок? Трех?
2. Сколько ключей, дающих различный результат шифрования/расшифрования, возможно для Полибианского квадрата (для фиксированного размера таблицы)?
3. Рассчитайте вероятность того, что в тексте из 100 символов буква «О» появится до 5 раз; в тексте из 1000 символов – до 50.
4. Составьте формулу расчета количества ключей в системе омофонов.
5. Как может быть взломана система Вижинера при шифровании длинного сообщения коротким ключем? Оцените необходимое количество циклов повторения ключа.
6. В первых версиях машины Enigma одновременно применялись 3 ротора из 5 возможных. Сколько возможно различных разовых ключей?

ТЕМА 6. ШИФРЫ ПЕРЕСТАНОВКИ

Общие сведения

Шифрами перестановки называются такие шифры, преобразования в которых приводят к изменению только порядка следования символов исходного сообщения [23].

В общем виде любой шифр перестановки можно представить в виде табл. 3.

Таблица 3

Таблица перестановок

Номер символа в открытом тексте	1	2	3	…	N
Номер символа в шифротексте	i_1	i_2	i_3	…	i_n

Также любую таблицу перестановок можно представить в виде перестановок пар элементов $\begin{pmatrix} a & b \\ b & a \end{pmatrix}$. Например, $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 3 \\ 3 & 1 \end{pmatrix}\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$.

Зная таблицу перестановок, можно выполнить как шифрование, так и расшифрование текста.

Шифры одинарной перестановки

Шифр простой одинарной перестановки описывается приведенной выше таблицей без изменений. Таблица в этом случае создается для всего сообщения. Применение такого шифра связано с необходимостью передавать большие таблицы. При больших n размер ключа превосходит размеры открытого текста. При этом шифр мог бы быть безусловно безопасным, но количество открытых текстов остается ограниченным.

Пример:

Открытый текст: “Шифр перестановки”.

Ключ: задан таблицей на рис. 10.

Шифротекст: “сенфвтршорипиеак”.

№ символа открытого текста	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
№ символа шифротекста	8	11	4	7	12	2	10	14	1	6	15	3	9	5	16	13
Открытый текст	ш	и	ф	р	п	е	р	е	с	т	а	н	о	в	к	и
Шифротекст	с	е	н	ф	в	т	р	ш	о	р	и	п	и	е	а	к

Рис. 10. Шифр простой одинарной перестановки

Проблему больших ключей устраняет **Шифр блочной одинарной перестановки**. Сообщение в этом случае разбивается на блоки по N символов. Перестановки замкнуты внутри каждого блока и одинаковы для всех блоков. Если длина сообщения не кратна длине блока, то сообщение дополняется произвольными символами [14].

Пример:

Открытый текст: “Шифр перестановки”.

Ключ: длина блока – 6 символов, перестановки указаны в таблице на рис. 11.

Шифротекст: реипшфтнеарсиувлок.

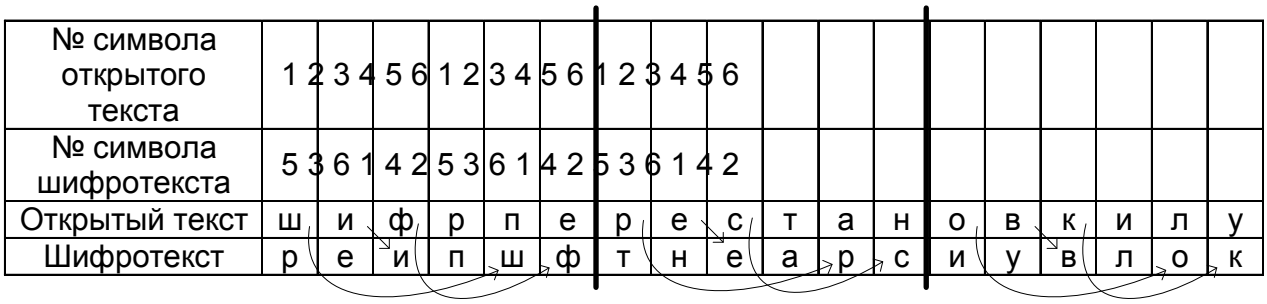


Рис. 11. Шифр блочной одинарной перестановки

В таком виде шифр удобно использовать, но и значительно легче вскрыть. Если блок короткий, то возможно вскрытие перебором вариантов, с учетом особенностей естественного языка. Также получение фрагмента открытого текста позволяет сразу же узнать ключ.

Шифры табличной перестановки

Исторически наиболее распространены были шифры **табличной перестановки.** В таких шифрах текст записывался в таблицу в определенном порядке, а затем результат выписывается в другом порядке [14].

В качестве ключа в шифрах табличной перестановки используются:

- размер таблицы;
- слово или фраза, задающие перестановку;
- особенности структуры таблицы.

Одним из самых примитивных шифров является простая табличная перестановка или **маршрутная табличная перестановка**. В этом шифре ключом является размер таблицы и маршруты вписывания и выписывания. Если остаются пустые клетки, то они заполняются произвольными символами.

Пример:

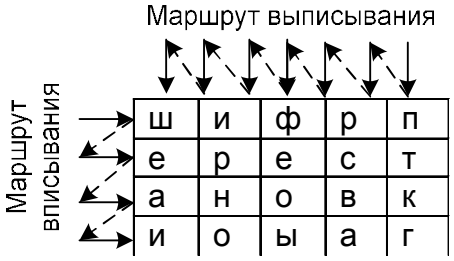


Рис. 12. Маршрутная табличная перестановка

Открытый текст: “Шифр перестановки”.
Ключ: размер таблицы – 5x4; маршрут вписывания – по горизонтали слева направо, сверху вниз ; маршрут выписывания – по вертикали, сверху вниз, справа налево.
Шифротекст: “пткгрсвафеоыирношеаи”.

Несколько более сложным является шифр **вертикальной перестановки.** В данном случае ключом является слово или таблица перестановок аналогичная блочному шифру. Количество столбцов определяется длиной ключа. Текст записывается по горизонтали, а выписывается по вертикали в соответствии с алфавитным порядком букв в алфавите. Если в ключе встречаются повторяющиеся буквы, то они нумеруются слева направо.

Пример:
Открытый текст: “Шифр вертикальной перестановки”.
Ключ: “таблица”.
Шифротекст: “иийарнсифкпнвлрвраеоштот”.

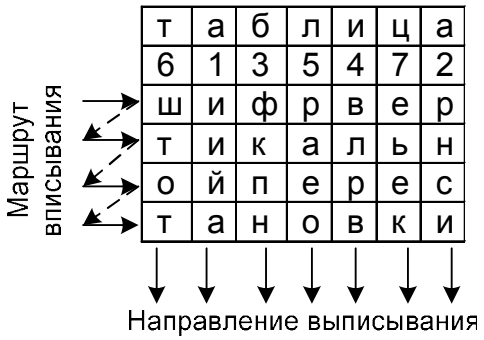


Рис. 13. Шифр вертикальной перестановки

Шифры множественной перестановки

В шифрах **множественной перестановки** повторно шифруется уже зашифрованный текст. Не имеет смысла повторно шифровать перестановкой сообщение, зашифрованное шифром простой одинарной перестановки, так как результат шифрования останется перестановкой, а количество возможных ключей шифрования не изменится. Аналогично для блочных шифров. Выполнение второй перестановки с той же длиной блока не усложнит процесс вскрытия, так как результатом окажется другая перестановка с той же длиной блока и той же длиной ключа. Если для повторной перестановки используется другой размер блока, то увеличивается и размер блока, и количество ключей. Если длина блока при первой перестановке – N, при второй – M, то длина блока – НОК(N, M); Количество ключей – N!*M!; но каждый символ может сдвинуться не более чем на N+M-1 символ.

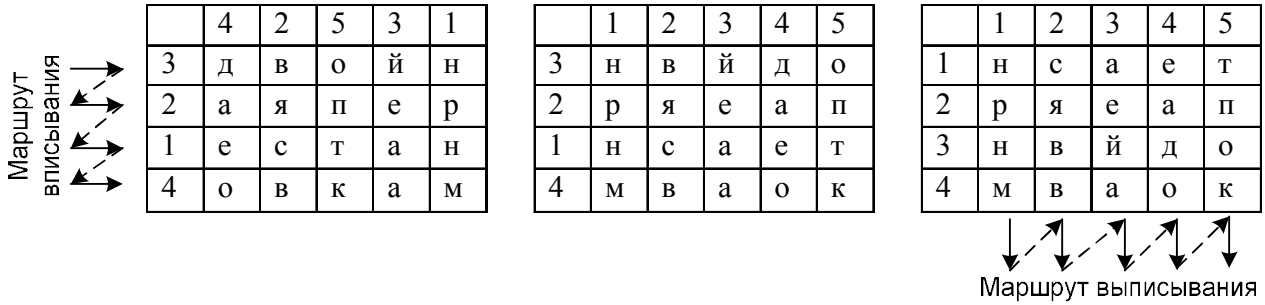
Повторное применение табличной перестановки с выписыванием и вписыванием в различных направлениях на каждом шаге, при таком же изменении количества ключей, позволяет перемешать символы по всему тексту.

В шифре **двойной табличной перестановки** в таблицу по определенному маршруту вписывается сообщение, переставляются столбцы и строки, затем сообщение выписывается по другому маршруту. После первой перестановки маршрут не изменяется.

Пример:

Открытый текст: "двойная перестановка".

Ключ: Размер таблицы: 5 x 4 ; Маршрут вписывания: по горизонтали, слева направо, сверху вниз ; маршрут выписывания: по вертикали, сверху вниз, слева направо ; перестановки столбцов: 4, 2, 5, 3, 1; перестановки строк: 3, 2, 1, 4.



	4	2	5	3	1
3	д	в	о	й	н
2	а	я	п	е	р
1	е	с	т	а	н
4	о	в	к	а	м

	1	2	3	4	5
3	н	в	й	д	о
2	р	я	е	а	п
1	н	с	а	е	т
4	м	в	а	о	к

	1	2	3	4	5
1	н	с	а	е	т
2	р	я	е	а	п
3	н	в	й	д	о
4	м	в	а	о	к

Рис. 14. Шифр двойной табличной перестановки

Шифротекст: "нрнмсявваейаеадотпок".

Количество ключей без учета маршрутов вписывания и выписывания и размера таблицы равно n!*m!, где m – количество столбцов, n – количество строк. В этом шифре, если подобрать маршрут выписывания и размер таблицы, мы получим запись в виде третьей таблицы. Символы, расположенные в одной строке таблицы, в открытом тексте находятся на расстоянии не более m-1 символов. Кроме того подбор порядка символов в одной из строк позволяет восстановить порядок символов во всех строках.

В настоящее время шифры перестановки применяются как части блочных шифров в виде блочной одинарной перестановки. Часто эти перестановки определены в самом алгоритме. Множественные перестановки не используются, так как размер блока фиксирован и определен алгоритмом.

Контрольные вопросы

1. Какое количество вариантов открытых текстов может быть составлено из символов шифротекста, если длина текста – N символов, количество вхождений каждого символа алфавита n_1, n_2, ...,n_m?
2. Поставьте в соответствие таблице перестановок в шифре блочной одинарной перестановки число в диапазоне от 0 до N!. Так, что бы ключ мог быть выражен одним числом.
3. В каких случаях множественные перестановки влияют на надежность алгоритма?
4. Можно ли менять порядок перестановок? В каких случаях?

ТЕМА 7. АДДИТИВНЫЕ ШИФРЫ

Общие сведения

Шифры, в которых шифрование может осуществляться побитово, называются **потоковыми**. Каждый бит исходного текста шифруется независимо от других с помощью гаммирования. В реальных системах одновременно шифруется по 8 бит или по 32, но при этом каждый бит шифруется независимо.

Аддитивный шифр – шифр, в котором для получения шифротекста используется бинарная операция аддитивного типа (сложение или сложение по модулю 2).

Гаммирование – преобразование открытого текста, при котором символы открытого текста складываются (по модулю, равному мощности алфавита) с символами псевдослучайной последовательности, вырабатываемой по определенному правилу.

Сложение по модулю N – остаток от деления суммы чисел на N.

Сложение по модулю 2 – побитовая операция «исключающее или», выполняемая для соответствующих битов данных. $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$.

Обычно, когда говорят «сложить два числа по модулю 2», то имеется в виду не получить остаток от деления на 2 суммы чисел, а сложить по модулю 2 соответствующие биты этих чисел.

Одноразовый блокнот

В 1917 г. Мэйджором Джозефом Моборном и Гилбертом Вернамом был изобретен идеальный способ шифрования. Он называется одноразовый блокнот (one-time pad) Этот шифр является единственной систе-

мой шифрования, для которого доказана абсолютная криптографическая стойкость [11].

Ключ в этом алгоритме представляет собой последовательность символов, которая удовлетворяет трем критически важным свойствам:

- Ключ должен быть истинно случайным. Для его получения не могут использоваться генераторы псевдослучайных чисел.
- Ключ должен совпадать по размеру с заданным открытым текстом;
- Ключ должен применяться только один раз.

В шифрах такого типа при шифровании используется сложение по модулю символов открытого текста с символами ключа. Если шифруется текст на естественном языке, то может использоваться сложение по модулю N, где N – длина алфавита. Если шифруются бинарные данные, то используется сложение по модулю 2.

Для того, что бы сложить по модулю N две буквы, необходимо определить их номера в алфавите (начиная с нуля), вычислить остаток от деления суммы на N и выполнить обратную замену полученного числа на букву. При расшифровании необходимо выполнить вычитание по модулю. Чтобы не получить отрицательных чисел к уменьшаемому можно прибавить N.

$C_i = T_i + K_i \; mod \; N.$

$T_i = C_i + N - K_i \; mod \; N.$

Таблица 4

Номера букв русского алфавита

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

Например:

Н+Ш mod 32 = 13+24 mod 32 = 37 mod 32 = 5 -> Е.

Также символы могут быть представлены в двоичном виде и сложены исключающим или, а затем преобразованы обратно в символы. В данном случае 32 буквы русского алфавита могут быть представлены 5-битными числами.

$C_i = T_i \oplus K_i.$

$T_i = C_i \oplus K_i.$

Например:

$Н \oplus Ш = 13 \oplus 24 = 01101 \oplus 11000 = 10101 = 21 \rightarrow Х.$

В отличие от рассмотренных ранее шифров, где для упрощения запоминания и передачи, в качестве ключей использовались слова или фразы, в этом шифре ключ большого размера передается записанным

(его запоминания не требуется), и использование осмысленных текстов в качестве ключа недопустимо.

Изначально шифр предполагалось использовать следующим образом.

1. Одной из сторон генерируется случайная последовательность символов и записывается на листки блокнота.

2. Копия такого блокнота передается другой стороне.

3. Отправитель шифрует открытый текст с помощью символов блокнота. Для одного символа открытого текста используется один символ блокнота. Страница, использованная для шифрования, уничтожается.

4. Получатель расшифровывает полученный текст с помощью блокнота и уничтожает шифротекст или использованные страницы блокнота.

Пример шифрования с сложением по модулю 32:

Открытый текст: "безопасный".

Ключ: "фзкцхпмсду".

Шифротекст: "хмтддпэюяь".

Таблица 5

Пример шифрования с сложением по модулю 32

Открытый текст (T)	Б	Е	З	О	П	А	С	Н	Ы	Й
№ символа T	1	5	7	14	15	0	17	13	27	9
Ключ (K)	Ф	З	Л	Ц	Х	П	М	С	Д	У
№ символа K	20	7	11	22	21	15	12	17	4	19
T + K mod 32	21	12	18	4	4	15	29	30	31	28
Шифротекст (C)	Х	М	Т	Д	Д	П	Э	Ю	Я	Ь

Пример шифрования с сложением по модулю 2:

Открытый текст: "безопасный".

Ключ: "фзкцхпмсду".

Шифротекст: "хвмшшпэьяъ".

Таблица 6

Пример шифрования с сложением по модулю 2

Открытый текст (T)	Б	Е	З	О	П	А	С	Н	Ы	Й
Bin-код символа T	00001	00101	00111	01110	01111	00000	10001	01101	11011	01001
Ключ (K)	Ф	З	Л	Ц	Х	П	М	С	Д	У
Bin-код символа K	10100	00111	01011	10110	10111	01111	01100	10001	00100	10011
T ⊕ K mod 32 (bin)	10101	00010	01100	11000	11000	01111	11101	11100	11111	11010
T ⊕ K mod 32 (dec)	21	2	12	24	24	15	29	28	31	26
Шифротекст (C)	Х	В	М	Ш	Ш	П	Э	Ь	Я	Ъ

Шифрование одноразовым блокнотом действительно является абсолютно стойким, так как для полученного шифротекста, без знания ключа равновозможными являются любые открытые тексты. Единственное что можно сказать об открытом тексте, это то, что его длина равна длине шифротекста. Более того, для любого шифротекста всегда существует такой ключ, который позволит получить любой заданный открытый текст заданной длины. Но для использования этого алгоритма, необходимо выполнить множество условий, которые делают шифр непригодным для практического использования.

Во-первых: Если мы допускаем возможность прослушивания канала связи, то ключ должен быть передан по абсолютно надежному каналу. Или достаточно знать, был ли ключ перехвачен в процессе передачи. Если был, то можно сгенерировать другой ключ и попытаться передать его. Пока не получили распространение квантовые каналы связи узнать факт перехвата остается невозможным. А если существует абсолютно надежный канал, то по нему может передаваться и открытый текст без изменения. Следствием в этом случае является передача ключа вручную (на блокноте или к современным условиям – на диске). Но и здесь нельзя избежать человеческого фактора (посыльный может продать ключи).

Во-вторых: Ключ должен быть абсолютно случайным. При использовании псевдослучайных генераторов можно добиться очень больших периодов, но, зная фрагмент псевдослучайной последовательности и алгоритм ее генерации, можно теоретически восстановить всю последовательность или данные инициализации генератора. Для некоторых алгоритмов восстановление последовательности требует недостижимого размера вычислительных ресурсов.

В-третьих: Ключ не может использоваться повторно. Если будут перехвачены два сообщения, зашифрованные одним и тем же ключом, то получив оба шифротекста, возможно они не будут сразу расшифрованы, но количество возможных осмысленных открытых текстов будет ограниченно или единственно (в зависимости от формата и длины сообщений). Если когда либо будет получен доступ к открытому тексту одного из сообщений, то все остальные сообщения, зашифрованные этим же ключом, будут сразу же расшифрованы.

Тем не менее, можно привести несколько примеров возможного использования одноразового блокнота. Если существует два канала связи с различной скоростью и различной степенью защищенности (медленный канал является защищенным). Ключ заранее передается по медленному каналу связи и при необходимости передачи сообщения, оно может быть передано по быстрому незащищенному каналу. Или другой аналогичный пример. Если существует два канала связи, с низкой вероятностью прослушивания и одновременное их прослушивание практически невозможно, то по ним может параллельно передаваться ключ и

шифротекст. В любом случае все отступления от классической схемы делают систему неидеальной.

Одноразовые блокноты используются и в настоящее время для шифрования сверхсекретных каналов связи с низкой пропускной способностью [11].

Шифры гаммирования

Самой сложной частью использования одноразового блокнота является передача ключа большого размера. В шифрах гаммирования вместо случайной последовательности используется псевдослучайная, полученная с помощью детерминированного генератора псевдослучайных чисел. То есть последовательность зависит только от параметров инициализации и никакие случайные параметры для генерации последовательности не используются. Запущенный дважды с одними и теми же параметрами инициализации, генератор должен выдать одинаковые последовательности.

Получение шифротекста из открытого текста и псевдослучайной последовательности происходит по тому же принципу что и при шифровании одноразовым блокнотом, но сама последовательность, называемая гаммой или потоком ключей, в этом случае не является ключом. Ключом являются параметры инициализации генератора ключевой последовательности.

На сегодняшний день создано огромное количество генераторов псевдослучайных чисел. Все они являются периодическими. Но их периоды могут значительно превышать размеры данных, которые будут когда-либо зашифрованы такими генераторами, либо возможности компьютеров сгенерировать последовательность такой длины. Период последовательности в 2^{256} считается достаточным для использования в самых серьезных криптографических приложениях.

Принцип работы шифров гаммирования можно показать схемой на рис. 15.

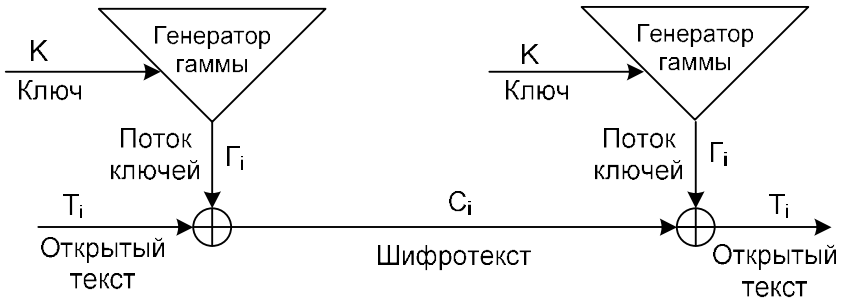


Рис. 15. Схема работы синхронных потоковых криптосистем

Шифр называют синхронным, так как каждый бит гаммы должен соответствовать своему биту шифротекста (или открытого текста). В случае пропуска одного бита, весь последующий текст будет расшифрован неправильно. Но в случае изменение одного бита неправильно будет расшифрован только этот бит.

Структура генератора гаммы представлена на рис. 16.

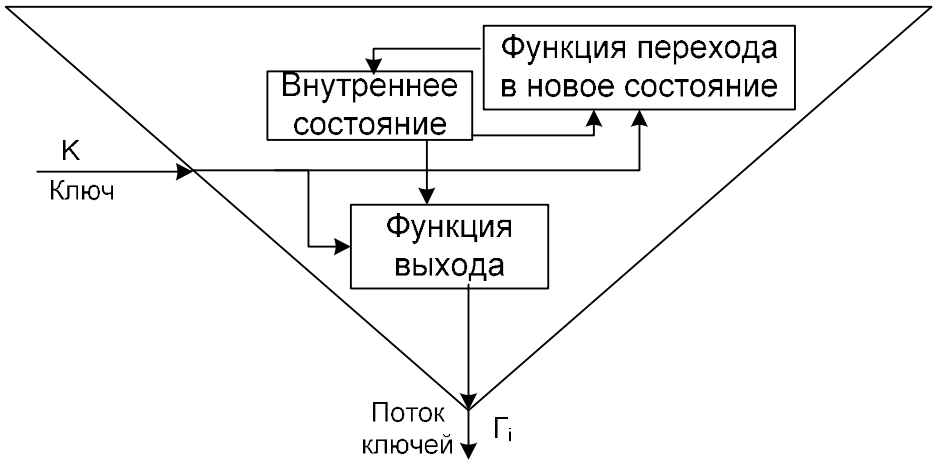


Рис. 16. Схема синхронного генератора гаммы

Пусть гамма генерируется в соответствии с следующими формулами:

$$X_n = (a \cdot X_{n-1} + b) \, mod \, m;$$

$$\Gamma_n = X_n \, mod \, 256,$$

где a, c и m – константы, являющиеся частью алгоритма; X_0 – ключ, задающий начальное состояние генератора; X_{n-1} – внутреннее состояние генератора в момент генерации; X_n. $X_n = (a \cdot X_{n-1} + b) \, mod \, m$ – функция перехода в новое состояние; $\Gamma_n = X_n \, mod \, 256$ – функция выхода. В данном случае функция выхода просто берет последние биты, внутреннего состояния, но может и осуществлять дополнительные преобразования, в том числе и зависящие от ключа.

Самосинхронизирующиеся шифры

Самосинхронизирующиеся шифры не требуют точного соответствия бит шифротекста, битам генерируемого ключа. Через некоторое время после начала работы дешифратор будет выдавать верный шифротекст. Схема работы самосинхронизирующихся генераторов представлена на рис. 17.

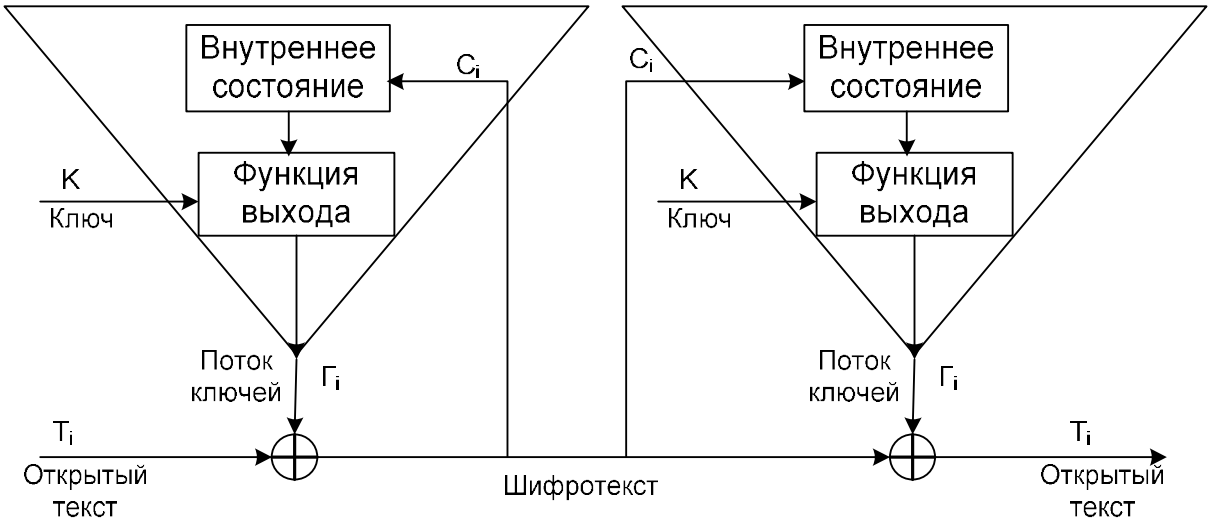


Рис. 17. Схема работы самосинхронизирующихся потоковых криптосистем

Внутреннее состояние такого генератора определяется N битами шифротекста и не зависит от внутреннего состояния и битов шифротекста, бывшего за N+1 шаг до этого (рис. 18).

Рис. 18. Внутреннее состояние самосинхронизирующегося генератора гаммы

В случае искажения или пропуска нескольких бит, внутреннее состояние генератора будет неверным в течение последующих N бит, и соответственно они будут расшифрованы неверно, после чего генераторы будут вновь синхронизированы.

Перед отправкой сообщения генераторы должны быть синхронизированы путем шифрования и расшифрования N бит произвольного текста, которые называются синхропосылкой.

В отличие от синхронных шифров, самосинхронизирующиеся уязвимы перед повторной передачей. Злоумышленник может не суметь расшифровать трафик, но может записать его и передать повторно. Тогда получатель, неверно расшифровав первые N бит, далее получит старый текст, который может заставить его выполнить повторно какие либо действия.

Для защиты от повторной передачи, необходимо использование меток времени или номеров сообщений. Наличие рассинхронизации при передаче должно приводить к повторной отправке сообщений.

Контрольные вопросы

1. Почему шифрование одноразовым блокнотом обладает абсолютной криптографической стойкостью?
2. За счет чего происходит синхронизация самосинхронизирующихся шифров?
3. Какие атаки могут быть осуществлены на шифры гаммирования?

ТЕМА 8. ГЕНЕРАТОРЫ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Требования к генераторам

Для получения последовательностей псевдослучайных чисел существует множество различных алгоритмов. Некоторые из них дают лишь видимость случайности и имеют короткий период либо предсказуемы. Некоторые могут использоваться в криптографических приложениях, требующих большой степени надежности.

Длиною периода гаммы называется минимальное количество символов, после которого последовательность начинает повторяться.

Случайность распределения символов по периоду означает отсутствие закономерностей между появлением различных символов в пределах периода.

Предсказуемость означает, что для восстановления всей гаммы необходимо относительно небольшое количество символов гаммы.

Если гамма имеет период 2^{256} бит, но при этом для восстановления всей гаммы требуется только 2000 символов открытого текста, то такой алгоритм не может быть использован в криптографических целях. И если, зная 2000 символов невозможно вскрыть всю последовательность, но при этом период составляет всего 4000 символов, то такой алгоритм также не может быть использован в криптографических целях. Если гамма имеет достаточный период и непредсказуема, но при этом 75 % битов равно «1»”, то злоумышленник может восстановить ~50 % бит, не взламывая генератор. Гамма должна иметь большой период, достаточный для шифрования всех сообщений, в течение всего срока службы, быть статистически случайной и непредсказуемой.

Линейные конгруэнтные генераторы

Линейными конгруэнтными генераторами называются генераторы вида $X_n = (a \cdot X_{n-1} + b) \bmod m$.

Здесь a, b и m – константы. a – называется множитель, b – инкремент, m – модуль. Ключом является значение X_0. Период такого генератора не превышает m. При этом для того, что бы период был близок к m, необходим правильный выбор констант. Существуют списки таких констант рекомендуемых для использования. Такие генераторы могут обеспечивать равномерное распределение, быструю генерацию чисел, но при этом, каждое число из этой последовательности однозначно определяет всю следующую последовательность.

Два линейных конгруэнтных генератора могут быть объединены. Генерируется две последовательности с различными константами и из первой вычитается вторая. В этом случае для модулей порядка 2^{31}, период объединенной последовательности может достигать 2^{60}. Но при этом внутреннее состояние такого генератора состоит только из двух чисел.

Сдвиговые регистры с линейной обратной связью (РСЛОС)

Сдвиговый регистр с обратной связью можно представить следующей схемой (рис. 19).

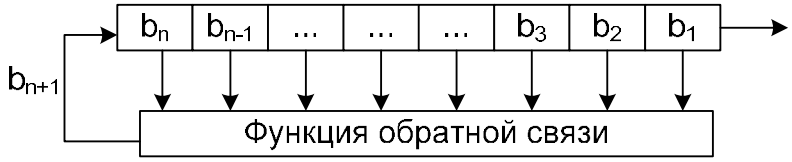


Рис. 19. Сдвиговый регистр с обратной связью

Регистр представляет собой последовательность бит. Если длина равна n битам, то регистр называют n-битовым сдвиговым регистром. Значения битов регистра поступают на вход функции обратной связи. Биты регистра сдвигаются, в старший записывается выход функции обратной связи, а младший становится элементом ключевой последовательности либо проходит дополнительные преобразования.

Если функция обратной связи представляет собой сложение по модулю 2 некоторых битов регистра, то обратная связь называется линейной. Перечень битов, участвующих в сложении называется отводной последовательностью, или отводами (рис. 20).

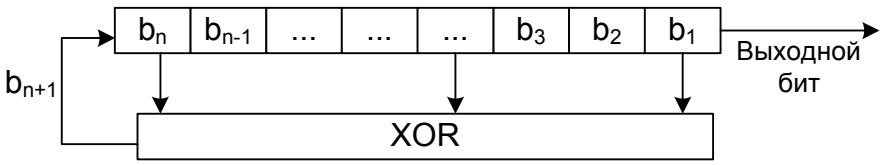


Рис. 20. Сдвиговый регистр с линейной обратной связью

Максимальное возможное число состояний сдвигового регистра равно 2^n–1 (если будет получена комбинация, состоящая только из нулей, то далее генератор будет выдавать только нули). Но, как и в случае линейных конгруэнтных генераторов, для получения последовательностей максимальной длины, необходимо задавать правильные позиции отводов. В [8] приведены математические основы вычисления позиций отводов, для получения РСЛОС с максимальным периодом, а в [11] – таблицы отводных последовательностей.

Ключом РСЛОС является начальное состояние регистра и отводная последовательность. Так как не все последовательности позволяют генерировать последовательности максимальной длины, то перед использованием они должны проверяться. Для того чтобы проверить, является ли РСЛОС максимальным, необходимо из отводной последовательности и 1 образовать многочлен и проверить его на примитивность.

Если отводы расположены на позициях n, k, l, m, то соответствующий многочлен будет $X^n + X^k + X^l + X^m + 1$. Многочлен является примитивным, если он является неприводимым, является делителем $X^{2^n-1} + 1$, но не является делителем $X^d + 1$, для всех d являющихся делителями $2^n - 1$. Поэтому для генерации примитивных многочленов степени n, необходимо знать разложение на множители числа $2^n - 1$ [8, 11]. Например, примитивным является многочлен $X^3 + X + 1$. Он соответствует РСЛОС на рис. 21.

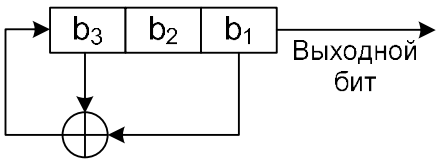


Рис. 21. 3-битовый максимальный РСЛОС, ассоциированный с многочленом $X^3 + X + 1$

Пусть регистр инициализирован последовательностью 0, 0, 1, тогда его состояния пройдут через следующие 7 состояний.

Таблица 7

Состояния РСЛОС ассоциированного с многочленом $X^3 + X + 1$

№	Состояние	Выход	№	Состояние	Выход
1	0 0 1	–	5	0 1 1	1
2	1 0 0	1	6	1 0 1	1
3	1 1 0	0	7	0 1 0	1
4	1 1 1	0	8(1)	0 0 1	0

Для генератора состоящего из одного РСЛОС, при известном положении отводов, достаточно определить n выходов, но если ключом является положение отводов, то для определения их положения достаточ-

но знать n+1 состояний, или 2n выходов. По ним может быть составлена система линейных уравнений вида

$$\begin{bmatrix} S_{n-1} & S_{n-2} & \dots & S_0 \\ S_n & S_{n-1} & \dots & S_1 \\ \dots & \dots & \dots & \dots \\ S_{2n-2} & S_{2n-3} & \dots & S_{n-1} \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_n \end{bmatrix} = \begin{bmatrix} S_n \\ S_{n+1} \\ \dots \\ S_{2n-1} \end{bmatrix},$$

где S_i – выходы; c_i – состояния отводов (0, если отвода на i-й позиции нет; 1 – если отвод есть). Система должна быть решена относительно $c_1 \dots c_n$.

Для приведенного выше примера может быть построена следующая система уравнений.

$$\begin{array}{l} 0 + 0 + c_3 = 1 \\ c_1 + 0 + 0 = 1 \\ c_1 + c_2 + 0 = 1 \end{array} \rightarrow \begin{array}{l} c_3 = 1 \\ c_1 = 1 \\ c_2 = 0 \end{array}.$$

Для того чтобы генераторы на основе РСЛОС можно было использовать в криптографических приложениях, выходы нескольких независимо работающих регистров пропускают через некоторую нелинейную функцию (рис. 22) [8].

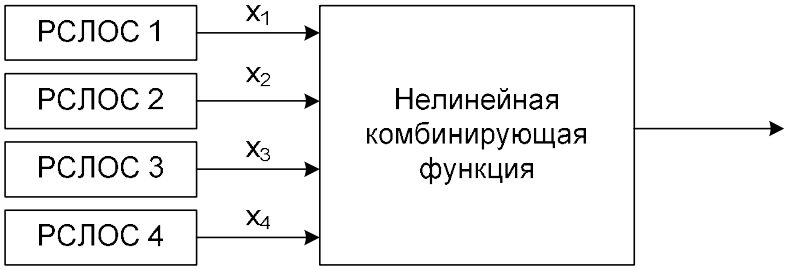


Рис. 22. Комбинирование РСЛОС

В качестве нелинейной комбинирующей, выбирают булеву функцию равную сумме различных произведений выходов РСЛОС. Например:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_2 x_3 x_4 \oplus x_1 \bar{x}_2 x_4 \oplus \bar{x}_1 x_2 x_4 \oplus x_1 x_3 \oplus x_2 x_3 \oplus \bar{x}_3 \oplus x_4.$$

Желательно, чтобы таблица истинности такой функции содержала примерно равное число нулей и единиц.

В качестве ключа такого генератора используется состояния инициализации регистров. В различных вариациях на тему регистров сдвига, в качестве ключа могут быть использованы любые изменяемые параметры.

Дополнительными усложнениями комбинации РСЛОС может быть также различная **частота генерации** битов различных регистров, в том числе и зависящая от общего выхода или выходов других регистров. Может использоваться **прореживание**, при котором на выходе генератора появляется сигнал, только если сигнал на одном из выходов

РСЛОС или их комбинации равен 1, **тактирование**, при котором новые биты, генерируются только на некоторых регистрах (говорят, регистр тактируется), остальные регистры остаются неизменными.

До сих пор многие потоковые генераторы основаны на РСЛОС и их вариациях. Некоторые из них давно вскрыты, некоторые возможно еще нет. В любом случае не стоит сильно доверять таким генераторам [11].

Аддитивные генераторы

Одним из недостатков рассмотренных выше генераторов является то, что они генерируют последовательности бит. На каждый цикл работы генератора на выходе мы получаем только один бит. При этом если процессор может обрабатывать одновременно 32 разряда, то битовые и 32-битовые генераторы будут выполнять один цикл за одинаковое время.

Аддитивные генераторы или запаздывающие генераторы Фиббоначи работают по тому же принципу, что РСЛОС, но вместо сложения по модулю 2 отдельных бит регистра, используется сложение n-битовых слов по модулю 2^n. Позиции слов участвующих в сложении выбираются по тем же принципам.

i-е слово последовательности получается в соответствии с следующей формулой:

$$X_i = (X_{i-a} + X_{i-b} + X_{i-c} + + X_{i-m}) \bmod 2^n .$$

Если учитывать только последние биты каждого слова, то мы получим классический РСЛОС. Соответственно период генерируемой последовательности будет не меньше периода соответствующего РСЛОС. При правильном выборе коэффициентов не менее $2^a - 1$.

На основе аддитивных генераторов также могут быть различные вариации.

Pike

Рассмотрим простой вариант комбинации аддитивных генераторов с тактированием [11, 32].

Pike использует три аддитивных генератора. Например:

$$A_i = (A_{i-55} + A_{i-24}) \bmod 2^{32}$$

$$B_i = (B_{i-57} + B_{i-7}) \bmod 2^{32}$$

$$C_i = (C_{i-58} + C_{i-19}) \bmod 2^{32} .$$

С предыдущего шага для каждого генератора должны быть сохранены биты переполнения (для первого, если $A_{i-55} + A_{i-24} > 2^{32}$, то 1, иначе 0). Если все биты переноса совпадают, то тактируются все три генера-

тора, если нет, то тактируются только два совпадающих, выход третьего остается неизменным.

Окончательным выходом является XOR выходов всех трех генераторов.

Алгоритм назван автором Россом Андерсоном «Pike» (Щука) в отличие от ранее созданного «Fish» (Рыба) от Fibonacci Shrinking Generator (Прореживаемый генератор Фиббоначи). После описания взлома «Fish» автор предлагает попробовать взломать броню «Щуки» [32]. Сведений о взломах пока нет, из чего вовсе не следует безопасность алгоритма.

Mush

Другой алгоритм «Mush» использует взаимное прореживание двух аддитивных генераторов:

$$A_i = (A_{i-55} + A_{i-24}) \bmod 2^{32}$$

$$B_i = (B_{i-52} + B_{i-19}) \bmod 2^{32}.$$

Тактируются оба генератора, и запоминаются биты переноса. Если для генератора A установлен, бит переноса, то тактируется генератор B. Если для генератора B установлен, бит переноса, то тактируется генератор A. Окончательным выходом является XOR выходов A и B.

Способы вскрытия такого генератора так же неизвестны.

RC4

RC – сокращение от Rivest's Cipher. Шифр RC4 разработан Рональдом Ривестом в 1987 г. для компании RSA Data Security. Шифр не был запатентован, но находился в частной собственности и оставался коммерческой тайной. Лицензии на использование алгоритма передавались после подписания соглашения о неразглашении. Тем не менее, в 1994 г. исходный код алгоритма был анонимно опубликован и быстро стал известен. С точки зрения компании RSA алгоритм и по сей день остается коммерческой тайной и для его использования необходима лицензия [11].

RC4 может использовать различные длины слов и различные длины ключей. Но кроме требований к лицензированию, он также подпадает под экспортные ограничения законодательства США, поэтому в России он не может быть законно использован с длиной ключа более 40 бит. С другой стороны, в России он и не является коммерческой тайной.

Итак, основным определяющим параметром является размер слова n (шифр работает не с битами, а со словами). В большинстве примеров это 8 бит, но на сегодняшний день может использоваться и 16. Количество ячеек внутреннего состояния равно 2^n, каждая по n бит. Необходимо, что бы все возможные слова были записаны в ячейках внутреннего

состояния. На первом этапе инициализации ячейки (S-блоки) заполняются последовательно значениями от 0 до $2^n - 1$.

```
For i = 0 to 2^n -1
 S[i] = i
```

Далее выполняется перемешивание блоков в соответствии с ключом.

```
j = 0
For i = 0 to 2^n -1
j = (j + S[i] + Key[i mod l]) mod 2^n
    Перестановка (S[i], S[j])
```

Здесь Key – ключ (инициализирующая последовательность) длины l бит. Для длины ключа здесь нет никаких ограничений (кроме законодательных).

Устанавливаются значения внутренних переменных – индексов

```
i = 0
j = 0
```

После завершения этих подготовительных этапов может выполняться непосредственно шифрование.

Для каждого цикла шифрования устанавливаются новые значения индексов:

```
i = (i + 1) mod 2^n
j = (j + S[i]) mod 2^n
```

Выполняется перестановка блоков с индексами i и j:

```
Перестановка (S[i], S[j])
```

Результатом является (рис. 23):

```
K = S[(S[i] + S[j]) mod 2^n].
```

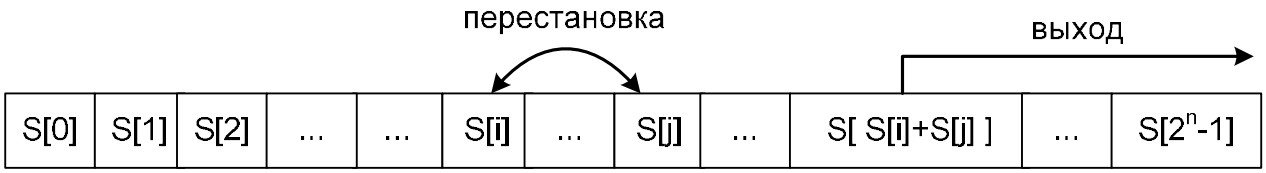


Рис. 23. RC4

Если длина слова составляет 8 бит, то количество различных внутренних состояний составляет 256! $256^2 \approx 2^{1700}$. Для 16 бит – 2^{954069}. Перебор такого числа состояний даже для 8 битовых слов невозможен и наверное никогда не станет возможным. Так же ничего не известно об успешных криптографических атаках на этот алгоритм. В результате самым узким местом является непосредственно ключ. Его длина должна выбираться из соображений невозможности полного перебора. Длина ключа в 40 бит явно недостаточна для обеспечения безопасности.

Алгоритмы, основанные на «нерешаемых» проблемах

Рассмотрим также несколько генераторов основанных на теории чисел, и на известных, нерешаемых на сегодняшний день задачах.

Blum-Micali

Генератор Blum-Micali использует известную проблему дискретного логарифмирования. Проблема является на сегодняшний день нерешаемой для больших чисел, в то время как для дискретного возведения в степень требуется относительно небольшое количество операций (но только относительно).

Дискретное возведение в степень ставит задачей найти a, зная g, x и p (g и p – простые числа).

$$a = g^x \bmod p$$

Эта задача решается за не более $2 log_2 x$ умножений по модулю. Например, для возведения числа в степень 11 (11 = 1011 $_2$) необходимо выполнить следующие операции:

$$g^{11} \bmod p = ((((g^2 \bmod p)^2 \bmod p)g \bmod p)^2 \bmod p)g \bmod p$$

При дискретном логарифмировании необходимо найти x, зная a, g и p. На сегодняшний день задача не решена математически, для ее решения требуется перебор значений x.

Генератор ПСЧ на основе этой проблемы строится в виде:

$$x_{i+1} = g^{x_i} \bmod p.$$

Если $x_i < \frac{p-1}{2}$*, то на выходе 1, иначе 0.*

Для того чтобы такой генератор был безопасен, необходимо, чтобы вычисление дискретных логарифмов было физически невозможным. Однако при этом для вычисления каждого бита требуется большое количество вычислений и алгоритм оказывается очень медленным.

RSA

Этот алгоритм генерации ПСЧ основан на тех же принципах факторизации больших чисел, что и ассиметричная система шифрования RSA. Проблема RSA будет рассмотрена в разделе «Криптосистемы с открытым ключом».

Параметрами системы являются числа $N = p \cdot q$, e взаимно простое с $(p-1)(q-1)$. Ключом является начальное состояние x_0.

Последовательность генерируется в соответствии с формулой:

$$x_{i+1} = x_i^e \ mod \ N .$$

Выходом может быть младший бит x_i. Время необходимое для генерации каждого следующего бита соответствует времени вычисления блока шифротекста в RSA.

Контрольные вопросы

1. Почему РСЛОС не могут применяться непосредственно для создания генераторов ПСЧ?
2. Что такое тактирование и прореживание, в чем смысл их применения?
3. В чем, на ваш взгляд, принципиальное отличие шифра RC4 от других приведенных здесь генераторов?
4. В чем состоит неудобство использования алгоритмов основанных на «нерешаемых» проблемах?

ТЕМА 9. БЛОЧНЫЕ ШИФРЫ

Общие сведения

Блочный шифр – шифр, в котором данные шифруются блоками одинакового размера, и результат функции шифрования очередного блока зависит только от значения входа функции и от значения ключа шифрования. Вход функции не обязательно является открытым тестом, а выход – шифротекстом. Важным структурным отличием от поточных является отсутствие внутреннего состояния.

Блочные шифры можно использовать в различных режимах работы, для решения различных задач (поточное шифрование, шифрование с обратной связью, независимое шифрование блоков, режимы вычисления хеш-функций и кодов аутентификации сообщений (имитовставок)).

В отличие от поточных шифров, с помощью блочных, в некоторых режимах, возможно обеспечение произвольного доступа к памяти. В частности, возможно шифрование разделов жесткого диска.

Большинство блочных шифров имеют удобную структуру для эффективной программной или аппаратной реализации.

Модель блочного шифра

Открытый текст M разбивается на блоки m_i, длиной по n бит (n может принимать для различных шифров значения 64, 128, 192, 256 бит и другие). Для того чтобы длина последнего блока открытого текста была

также равна n бит, применяется операция дополнения, которая должна быть однозначно обратимой. Ключом K является произвольная последовательность бит длиной s символов (диапазон длин ключей аналогичен – от 56 бит в DES, до 256 бит в ГОСТ и др.). Выходом функции шифрования E является блок шифротекста C_i, длина которого равна длине блока открытого текста.

Процедуры шифрования и расшифрования записываются:

$$C_i = E_K(m_i) \text{ и } m_i = D_K(C_i),$$

где E – функция шифрования; D – функция расшифрования; m – открытый текст; K – ключ; C – шифротекст (рис. 24).

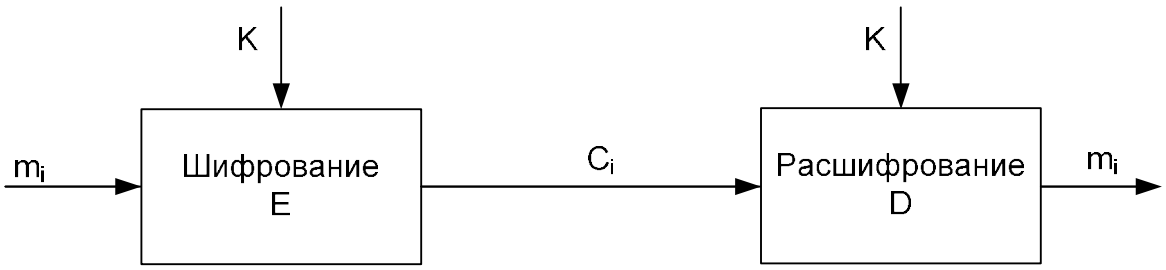


Рис. 24. Модель блочного шифра

Функция шифрования, оставаясь неизменной, может применяться в различных **режимах шифрования**. Вместо открытого текста на вход функции шифрования может подаваться функция открытого текста, предыдущего блока шифротекста и счетчика; а результатом шифрования может быть также некоторая функция выхода функции шифрования, предыдущего блока шифротекста, открытого текста и других данных.

$$C_i = f_1(E_K(f_2(m_i, C_{i-1}, nonce), m_i, C_{i-1})$$

Функции f_1 и f_2 обычно возвращают один из аргументов или сумму по модулю 2 двух аргументов. Распространенные режимы шифрования будут рассмотрены ниже. Если в режиме шифрования используется составляющая C_{i-1}, то $C_0 = IV$, где IV – вектор инициализации.

Дополнение

Так как исходное сообщение может быть любой длины (реально – кратно байту), то длина последнего блока может оказаться не равной длине блока, принятой в алгоритме. Чтобы избежать этого, к последнему блоку применяется операция дополнения. Требование к дополнению состоит в том, чтобы по дополненному сообщению можно было уникальным образом определить исходное. Если длина сообщения кратна длине блока без дополнения, то дополнение все равно должно выполняться, для обеспечения уникальности восстановления исходного сооб-

щения. Если неизвестно, было ли произведено дополнение, то всегда существуют сообщения, последние биты которых могут восприниматься либо как часть сообщения, либо как дополнение

Рассмотрим два примера выполнения дополнения.

1. Добавить к открытому тексту один байт с определенным значением.

2. Добавить такое количество нулевых байт, чтобы общая длина стала кратной b (количество добавляемых байт лежит в диапазоне 0 – n – 1).

Или:

1. Определим сколько байт необходимо добавить к сообщению, что добиться длины кратной b и обозначим количество дополнительных байт как n, где $1 \le n \le$ b и $n + l(M)$ кратно b.

2. Присоединим к открытому тексту n байтов, каждый из которых будет иметь значение n.

После выполнения дополнения длина последнего блока равна n и дополнение может быть снято единственным образом. Если после расшифрования, дополнение не может быть снято единственным образом или оно некорректно, то это следует расценивать как ошибку при передаче или попытку атаки с модификацией шифротекста.

Вектор инициализации

Как уже упоминалось, в некоторых режимах шифрования, для шифрования первого блока используется вектор инициализации IV (Initial Value). Так как вектор инициализации заменяет в схемах шифротекст нулевого блока, то он не является секретной информацией и может передаваться открыто. Рассмотрим некоторые подходы к его формированию и передачи.

Вектор-счетчик – для первого сообщения с данным ключом, значение IV принимается равным нулю или некоторому заранее оговоренному значению, для каждого следующего сообщения $IV_i = IV_{i-1} + 1$. Такой метод обеспечивает уникальность вектора, но если первые блоки сообщений одинаковы, то при объединении с вектором, шифротексты первых блоков могут быть коррелированны.

Случайный вектор инициализации – первое значение вектора IV генерируется случайно и посылается в качестве нулевого блока перед шифрованным сообщением.

Однажды используемое число – nonce (от number used once), иногда также называют меткой времени – решение, объединяющее первые два подхода. Число формируется на основе номера сообщения в системе, текущего времени или случайных чисел так, чтобы оно не могло быть сгенерированно повторно. Использование nonce позволяет решить две проблемы: нумерация сообщений позволяет определить, было ли отправлено старое сообщение или новое (защита от повторной передачи).

Если номер сообщения меньше номера полученного ранее, то такое сообщение должно быть отброшено. Уникальность номера не позволяет злоумышленнику найти одинаковые шифротексты, полученные из одинакового открытого текста.

Также необходимо учитывать, что номер должен быть уникален для всей системы, в рамках которой используется один ключ. Этого можно добиться, добавляя к номеру идентификатор отправителя. Для формирования nonce можно использовать следующую последовательность шагов.

1. Присвоить сообщению номер, в соответствии с рекомендациями.
2. Зашифровать номер с помощью блочного шифра, ключом, известным получателю. Это значение и будет вектором инициализации.
3. Перед началом сообщения вставить номер незашифрованным. Если в номере нет никакой случайности и зависимости от времени, то он может не передаваться совсем.
4. Использовать зашифрованный вектор, для шифрования сообщения.
5. Получатель, зная ключ, шифрует полученный номер сообщения и использует результат в качестве вектора инициализации для расшифрования.

Режимы шифрования

Электронная кодовая книга (ECB)

В режиме электронной кодовой книги (простой замены) блоки открытого текста шифруются независимо от других блоков. Шифротекст зависит только от блока открытого текста и ключа и не зависит от шифротекстов предыдущих блоков, вектора инициализации и других параметров (рис. 25).

$$C_i = E_K(m_i),$$
$$m_i = D_K(C_i).$$

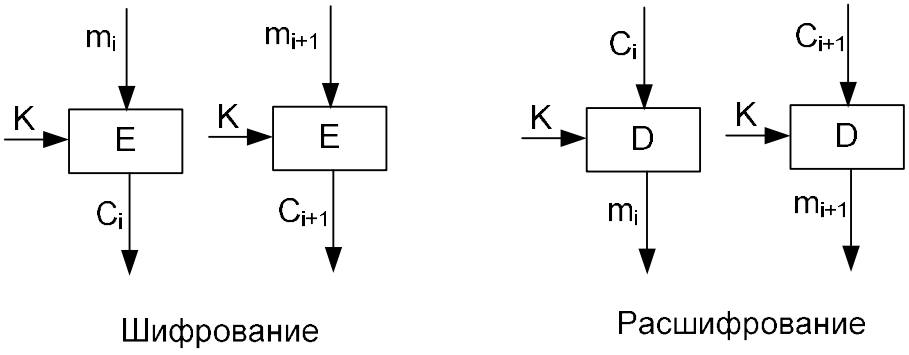


Рис. 25. Режим ECB

Этот режим назван режимом электронной кодовой книги (electronic codebook – ECB), так как теоретически существует возможность создать книгу (таблицу), в которой каждому блоку открытого текста будет сопос-

тавлен блок шифротекста. В этом случае, при длине блока в n бит, книга будет содержать 2^n записи, и каждая таблица будет соответствовать одному ключу.

Режим ECB редко используется для шифрования больших сообщений (ГОСТ запрещает использование этого режима, кроме случаев шифрования ключей). Если злоумышленник обнаружит повторяющиеся блоки шифротекста, он будет знать, и что соответствующие блоки открытого текста одинаковы и учитывая структуру сообщения сможет сделать догадки о значении этих блоков, что уже является серьезной проблемой. Кроме того, он может провести атаку с известным открытым текстом. Хороший блочный шифр должен быть стойким к таким атакам, но тем не менее, лучше не давать лишний раз злоумышленнику возможности проведения атак.

Сцепление шифрованных блоков (CBC).

Режим CBC (chipper block chaining) является наиболее популярным. Каждый блок открытого текста складывается по модулю 2 с предыдущим блоком шифротекста (рис. 26). Шифрование и расшифрование осуществляется следующим образом:

$$C_i = E(m_i \oplus C_{i-1}),$$

$$m_i = D(C_i) \oplus C_{i-1}.$$

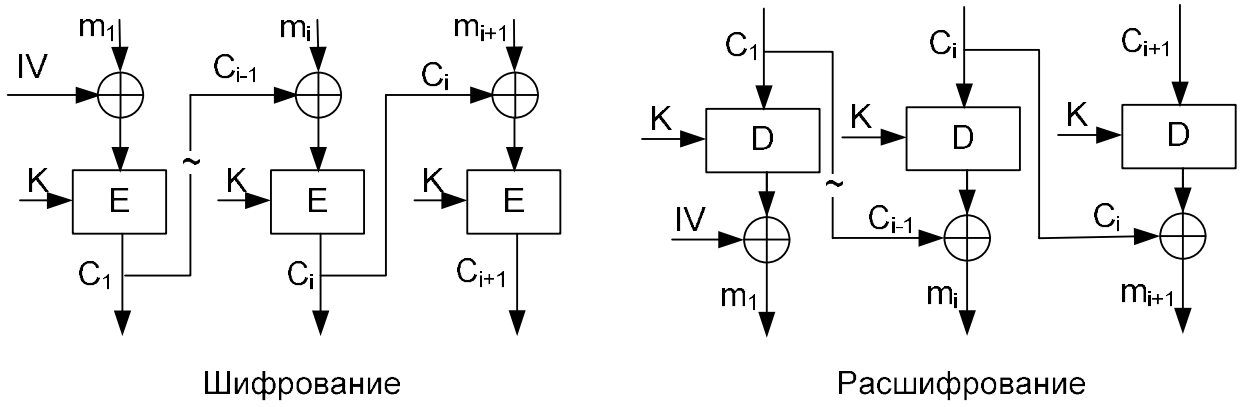


Рис. 26. Режим CBC

В режиме CBC одинаковые блоки открытого текста преобразуются в разные блоки шифрованного текста.

Режим CBC не применим для шифрования устройств с произвольным доступом к памяти для записи. Шифрование может выполняться только последовательно. Расшифрование может осуществляться в произвольном порядке. Такая схема может быть сравнима с шифрованием CD: шифрование и запись выполняется один раз и последовательно, но

расшифрование и чтение может выполняться с любого блока (достаточно знать шифротекст предыдущего блока).

Режим гаммирования

В режиме гаммирования алгоритм блочного шифрования задействован для усложнения предварительной гаммы (рис. 27).

$$C_i = m_i \oplus E_K(\Gamma_i),$$

$$m_i = C_i \oplus E_K(\Gamma_i).$$

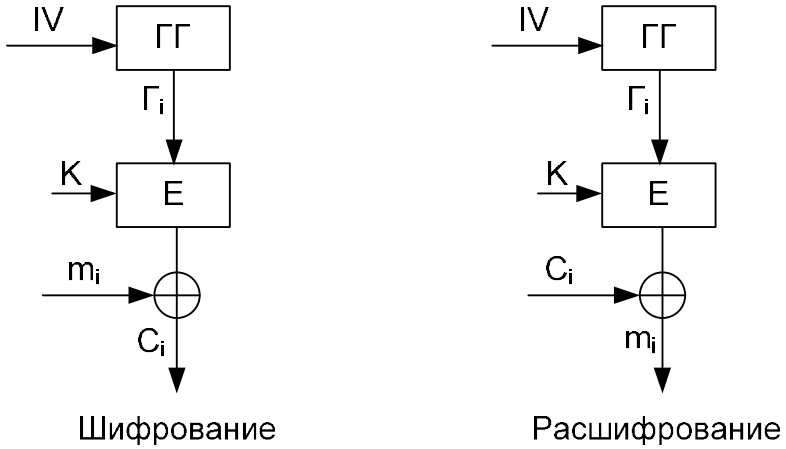


Рис. 27. Режим гаммирования

Некоторые генераторы гаммы вырабатывают гамму, которую можно вскрыть по достаточно коротким открытым текстам. В случае вычисления для гаммы функции шифрования, если злоумышленник может предположить открытый текст – это даст ему только значение шифрованной гаммы.

Ключ генератора гаммы, должен посылаться в зашифрованном виде. В этом случае будет обеспечиваться двойная надежность за счет объединения двух разнородных шифров. Если ключ будет посылаться в открытом виде, то данный режим аналогичен режиму CTR.

Генератор гаммы должен удовлетворять двум основным требованиям [31].

1. Необходимо, чтобы период повторения гаммы был достаточно большим. Он должен превосходить количество блоков, шифруемых с использованием одного и того же ключа.

2. Необходимо, чтобы соседние, или близкие блоки гаммы отличались хотя бы в нескольких битах.

При этом качество гаммы не имеет особого значения. Могут использоваться СРЛОС, аддитивные и даже конгруэнтные генераторы.

Режим гаммирования имеет все преимущества и недостатки шифров гаммирования. Генераторы гаммы могут быть как синхронными, так и с самосинхронизацией (достоинства и недостатки остаются прежними).

Обратная связь по выходу (OFB)

Режим OFB (output feedback) используется аналогично режиму гаммирования. Но в этом режиме нет узла выработки гаммы. Выход функции шифрования поступает на вход функции шифрования следующего блока. Шифротекст является суммой по модулю 2 выхода функции шифрования и открытого текста (рис. 28).

$$\Gamma_0 = IV,$$
$$\Gamma_i = E_K(\Gamma_{i-1}),$$
$$C_i = m_i \oplus \Gamma_i,$$
$$m_i = C_i \oplus \Gamma_i.$$

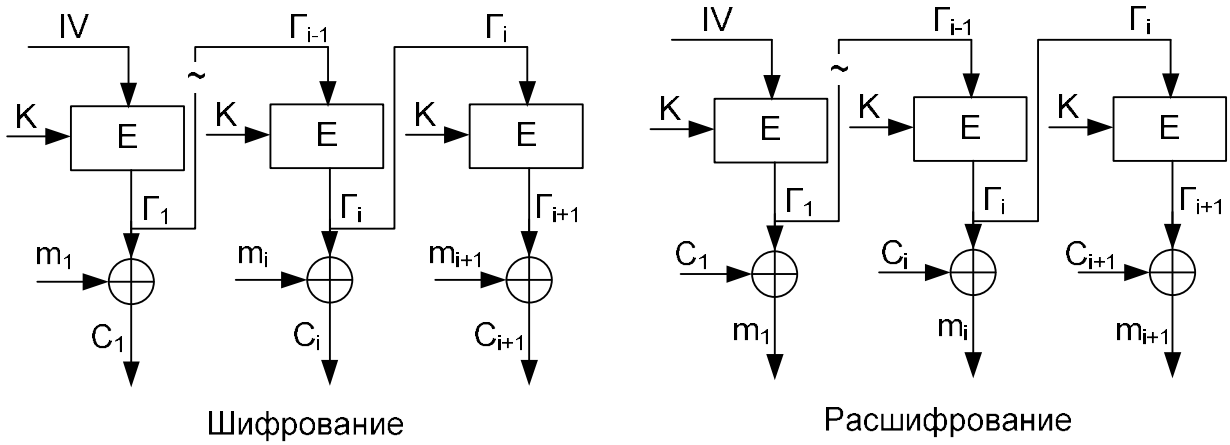


Рис. 28. Режим OFB

Вектор инициализации здесь должен быть уникальным. Если будет повторно использован один и тот же вектор инициализации, то будет сгенерированна точно такая же гамма, и следовательно можно будет найти различие между шифротекстами и открытыми текстами: $m_1 \oplus m_2 = C_1 \oplus C_2$.

В режиме шифрования OFB, как и в режиме гаммирования алгоритм шифрования полностью совпадает с алгоритмом расшифрования, и не требуется реализации функции расшифрования (при желании можно использовать даже однонаправленные функции, зависимые от ключа). Нет необходимости в дополнении, так как может использоваться столько байт гаммы, сколько необходимо. Также могут использоваться блоки произвольной (заранее определенной) длины, меньшей длины блока

функции шифрования. Как и в случае использования синхронных генераторов, ошибка в одном бите гаммы приведет к изменению одного бита шифротекста/открытого текста, а пропуск одного бита к нарушению всей следующей последовательности. Для шифрования или расшифрования произвольного блока сообщения в этих режимах необходимо генерировать гамму для всех предшествующих блоков.

Обратная связь по шифротексту (CFB)

В режиме CFB (Cipher-Feedback Mode) на вход функции шифрования подается предыдущий блок шифротекста. А шифротекст является суммой выхода функции шифрования и открытого текста (рис. 29).

$$C_0 = IV,$$
$$C_i = m_i \oplus E_K(C_{i-1}),$$
$$m_i = C_i \oplus E_K(C_{i-1})$$

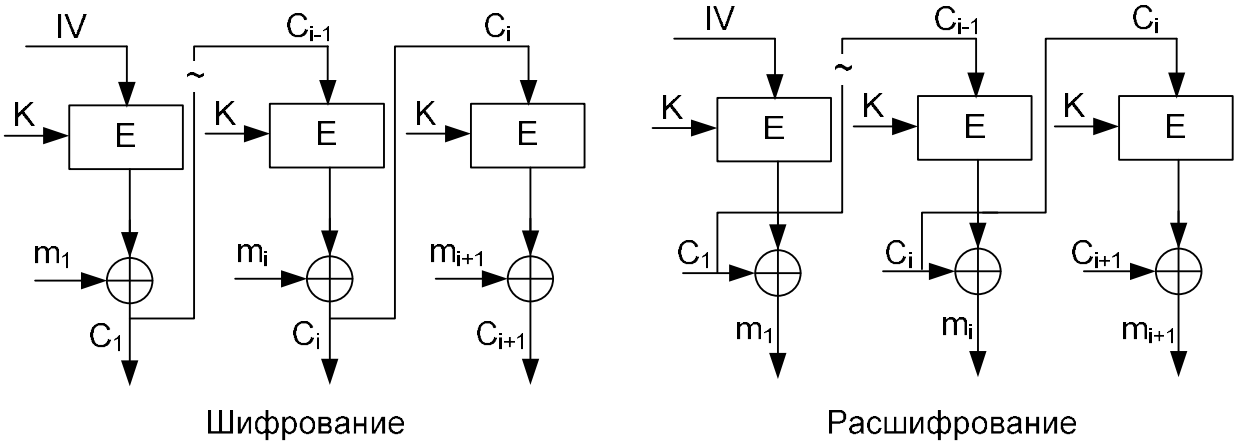


Рис. 29. Режим CFB

Этот режим можно сравнить с самосинхронизирующимися шифрами гаммирования. Ошибка при передаче приведет к тому, что только два блока открытого текста будут расшифрованы неверно, после чего ошибка не влияет на расшифрование. В этом режиме шифрование возможно только последовательно, а расшифрование – начиная с любого блока. Для расшифрования в этом режиме также не нужно реализовывать функцию расшифрования.

Счетчик (CTR)

Режим CTR (counter) основан на использовании неповторяющихся номеров блоков. В качестве номеров могут использоваться последовательные значения или значения, получаемые в соответствии с фиксиро-

ванным алгоритмом. Номера должны быть такими, чтобы корреляции между последовательными блоками были минимальны. В базовой форме, режим CTR представлен на рис. 30.

$$K_i = E_K(nonce \,||\, f(i)),$$

$$C_i = m_i \oplus K_i.$$

$$m_i = C_i \oplus K_i$$

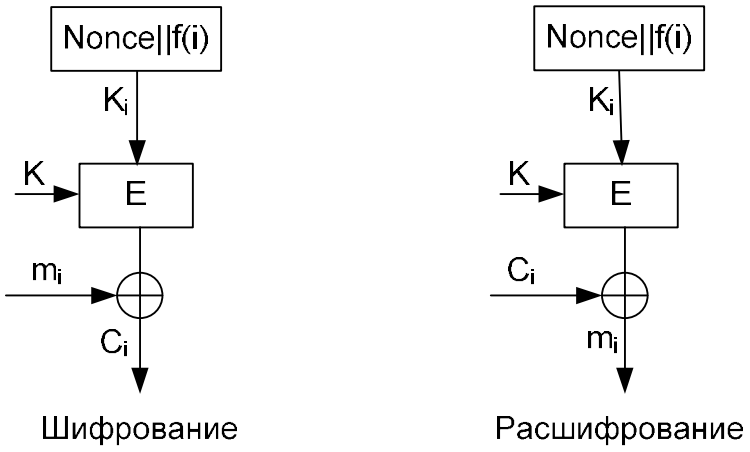


Рис. 30. Режим CTR

Здесь nonce – уникальный номер сообщения, i – номер блока в сообщении, || здесь и далее означает конкатенацию (соединение) строк. При расшифровании функция расшифрования также не требуется. При описании режима гаммирования уже было сказано, что не рекомендуется использовать блоки гаммы, отличающиеся только одним битом. Чтобы уменьшить зависимости между блоками, можно вместо $nonce||i$ использовать $nonce||f(i)$ или $f(nonce||i)$. Например функция $f(i) = a^i \bmod p$ ($a < p$, p – простое) не повторяется для i от 1 до p-1 и легко вычислима в одну сторону.

Nonce и i могут передаваться открыто и не являются секретными. Основное требование – уникальность результирующей строки.

Режим CTR обладает явными преимуществами перед другими шифрами, так как позволяет реализовать произвольный доступ к памяти (произвольным частям открытого текста), как для чтения, так и для записи. Для любого блока открытого текста или шифротекста может быть рассчитан номер блока и для него может быть рассчитан блок ключевого потока.

Существуют и другие режимы шифрования, применяемые в отдельных стандартах, но они менее изучены.

Накопление ошибок в различных режимах шифрования

Ошибка в бите данных – это изменение бита со значением «1» на бит «0» и наоборот. Ошибки могут возникнуть при сбое оборудования или быть вызваны злоумышленником искусственно. Ошибка может возникнуть в шифротексте, векторе инициализации, значении счётчика и может повлиять на результат шифрования, причем в различных режимах влияние на результат будет отличаться (табл. 8) [27].

Таблица 8

Эффект распространения ошибки

Режим	ошибка в бите шифротекста блока C_j	ошибка в бите IV (или счетчике)
ECB	ошибка в любом бите при дешифровании блока C_j	не используется
CBC	ошибка в любом бите при дешифровании блока C_j ошибка в одном бите при дешифровании блока C_{j+1}	ошибка в одном бите при дешифровании блока C_1
CFB	ошибка в одном бите при дешифровании блока C_j ошибка в любом бите при дешифровании блока C_{j+1}	ошибка в любом бите при дешифровании блока C_1
OFB	ошибка в одном бите при дешифровании блока C_j	ошибка в любом бите при дешифровании всех блоков
CTR	ошибка в одном бите при дешифровании блока C_j	ошибка в любом бите при дешифровании блока C_j

В любом режиме ошибка в пределах блока шифротекста, ведёт к неправильному дешифрованию этого блока. В режимах OFB и CTR ошибка в одном бите шифротекста приведет к ошибке в том же бите открытого текста.

В режиме CFB ошибка в одном бите шифротекста приведет к ошибке в одном бите того же блока открытого текста и к ошибкам в любом бите следующего блока.

В режимах же ECB и CBC повреждённым, в зависимости от силы преобразования используемого блочного шифра, может стать любой бит с вероятностью повреждения 50 %. В режиме CBC такая ошибка приведёт к ошибке только в том же бите при дешифровании следующего блока сообщения. В режиме CTR ошибка в бите счётчика также приведёт к

возможности изменения любого бита соответствующего блока с вероятностью 50 %.

В режимах ECB, OFB и CTR ошибка в бите отдельного блока шифруемого текста не повлияет на другие блоки при дешифровании.

В режиме OFB ошибка в одном бите вектора инициализации приведёт к полностью неверным результатам расшифрования, так как гамма зависит только от IV и ключа. В режиме же CFB эта ошибка так же, как и ошибка в шифротексте, заденет только первый блок.

Алгоритмы блочного шифрования

Практически все алгоритмы блочных шифров, используемые на практике, представляют собой несколько последовательных применений слабого блочного шифра. Каждое последовательное применение называется раундом. Несколько раундов слабого шифра могут в результате выдать стойкий к атакам шифр. Известно множество атак на модификации шифров с меньшим количеством раундов, однако успешные атаки на алгоритм целиком появляются редко.

Сеть Фейстеля

Большинство блочных алгоритмов являются сетями Фейстеля (Felstel networks). Сеть Фейстеля позволяет преобразовать произвольную функцию f в блочный шифр.

Шифруемый блок открытого текста разбивается на два подблока равной длины:

L_0 – *левый блок,*

R_0 – *правый блок,*

При шифровании на каждом шаге для i от 1 до n вычисляется:

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

$R_i = L_{i-1}$.

На последнем шаге перестановка L и R обычно не выполняется, чтобы обеспечить симметричность алгоритмов шифрования и расшифрования.

Шифротекстом являются блоки L_n и R_n.

Расшифрование выполняется в обратном порядке, с использованием той же функции. Для i от n–1 до 0 вычисляется:

$R_i = L_{i+1} \oplus f(R_{i+1}, K_{i+1})$,

$L_i = R_{i+1}$.

При использовании сетей Фейстеля, функция F может быть сколь угодно сложной, так вычисление обратной функции не требуется. Возможность расшифрования определяется только сетью Фейстеля. Для того, чтобы обратить алгоритм без последней перестановки (рис. 31) необходимо только изменить порядок ключей.

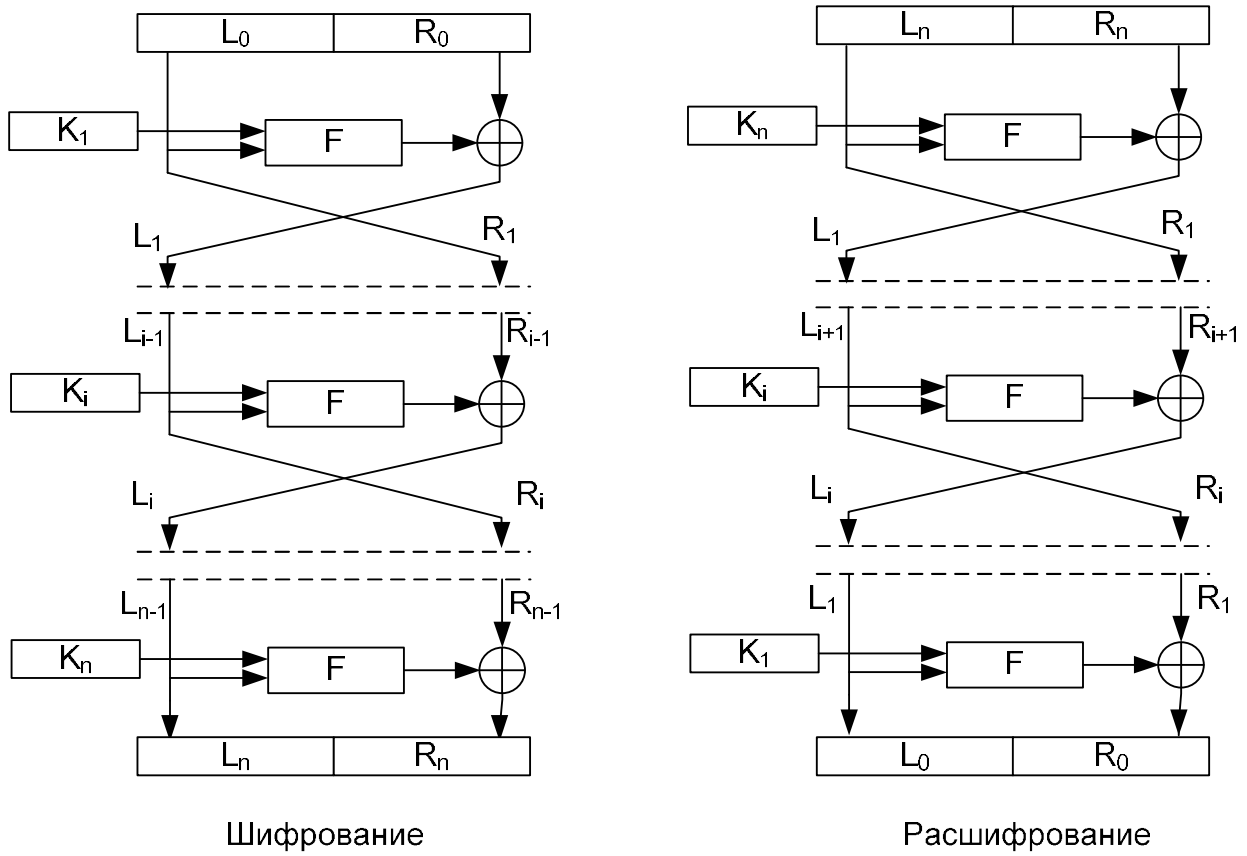


Рис. 31. Сеть Фейстеля

Безопасность шифра зависит от используемой функции F, и от количества циклов. Функция F строится так, чтобы максимально усложнить ее математическое описание и обычно состоит из замен и подстановок.

DES

DES (от Data Encryption Standard) – блочный алгоритм шифрования, принятый в качестве стандарта США в 1978 г.

DES является классической сетью Фейстеля. Данные шифруются 64-битными блоками. Ключом является 64-битная строка, в каждом байте которой, младший бит используется для контроля четности (требуется, чтобы сумма бит каждого байта была нечетной). Соответственно, длина ключа составляет 56 бит.

Процесс шифрования состоит из трех этапов.

1. Начальная перестановка (IP) 64-битного исходного текста.
2. 16 раундов сети Фейстеля с функцией подстановки.
3. Перестановка IP^{-1}, обратная начальной.

Подробное описание DES смотрите по [14].

Шифр был рекомендован для применения в режимах ECB, CBC, CFB и OFB, которые до сих пор являются основными режимами шифрования.

AES

История

К концу 90-х гг. стали сильно заметны недостатки DES. Основной недостаток – небольшая длина ключа – 56 бит. При существенно возросшей аппаратной производительности стал возможен полный перебор ключей. Пока не был принят новый стандарт, рекомендовалось использовать трехкратный DES (3DES), с тремя различными ключами. Это позволяло увеличить длину ключа до 168 бит, что недостижимо для полного перебора, однако такой способ требует трехкратного увеличения количества вычислений при шифровании и расшифровании.

В 1998 г. Американский институт стандартизации NIST – National Institute of Standards & Technology объявил конкурс на стандарт симметричного блочного шифрования.

В финал конкурса вышли алгоритмы:

- MARS – разработка корпорации IBM, основанная на классической сети Фейстеля и многочисленных математических операциях;
- RC6 – модификация блочного шифра RC5. Использует только основные математические преобразования, битовые сдвиги и, в качестве функции перемешивания, операцию T(X) = X*(X + 1);
- Serpent – шифр использует только операции табличных подстановок, исключающего «ИЛИ» и битовых сдвигов в тщательно подобранной очередности.
- TwoFish – достаточно сложная в реализации разработка компании Counterpane Security Systems, воплотившая много интересных идей из алгоритма предшественника BlowFish.
- Rijndael – разработан двумя специалистами по криптографии из Бельгии. Он является нетрадиционным блочным шифром, поскольку не использует сеть Фейстеля для криптопреобразований. Большинство вычислений являются работой с полиномами.

Rijndael, теперь известный как AES (по названию конкурса), стал победителем и был принят в качестве стандарта шифрования правительством США в 2000 г. Rijndael поддерживает различные длины ключей и блоков, от 128 до 256 бит, В AES в качестве стандарта принята длина блока 128 бит, а длины ключей могут быть от 128, до 256 бит.

Рассмотрим алгоритм AES, с длиной блока и ключа по 128 бит.

Блок открытого текста и ключ, представляются в виде матрицы 4×4 байта (4×6 и 4×8 для 192 и 256 бит), называемой состоянием (state). Состояние и ключ заполняются из исходных строк сверху вниз, слева направо. После шифрования, порядок байт остается тем же.

Как и в других алгоритмах, при шифровании множество раз выполняются повторяющиеся действия. Количество раундов зависит от размеров матриц и составляет 10 для минимальных размеров и 14 для максимальных. Будем обозначать Nn – число раундов, Nc – число колонок в матрице ключа.

Таблица 9

Число раундов AES

Nc	4	5	6	7	8
Nn	10	11	12	13	14

Раунд AES состоит из четырех операций.

- SubBytes – Нелинейная подстановка.
- ShiftRows – Циклический сдвиг вправо каждой строки матрицы на определённое число байт.
- MixColumns – Операция перемешивания байт каждого столбца матрицы путём умножения столбца на определённую матрицу.
- XorRoundKey – Каждый байт состояния складывается с ключом раунда (ключ каждого раунда получают из ключа шифрования, используя процедуру разворачивание ключа).

В последнем раунде шифрования отсутствует операция MixColumns.

Порядок выполнения операций можно представить псевдокодом:

```
XorRoundKey(state, k[0], Nc)
for (i=1; i< Nn; i++)
{
        SubBytes(state, Nc)
        ShiftRows(state, Nc)
        MixColumns(state, Nc)
        XorRoundKey(state, k[round], Nc)
}
SubBytes(state, Nc)
ShiftRows(state, Nc)
```

```
XorRoundKey(state, k[Nn], Nc)
```

В AES основным является полиномиальное представление чисел. Так байт {01100011} следует представлять, как $x^6 + x^5 + x + 1$. Операции умножения, вычисления остатков выполняются в конечном поле GF(2^8) по модулю неприводимого полинома $m(x) = x^8 + x^4 + x^3 + x + 1$. Коэффициенты полинома могут быть 0 или 1, соответственно сложения выполняются по модулю 2. Также на 3 шаге используется полином с коэффициентами из поля GF(2^8). То есть каждый коэффициент в этом случае сам является многочленом из GF(2^8).

Рассмотрим используемые в алгоритме операции.

Шаг 1. SubBytes.

Производится замена всех байт состояния. Функция замены имеет четкую математическую структуру и может быть описана формулами:

Каждый байт заменяется на мультипликативное обратное, по модулю $m(x)$. Мультипликативное обратное может быть найдено с помощью расширенного алгоритма Евклида, если реализовать сложение, умножение и вычисление остатков для многочленов, однако его нахождение удобнее выполнять по таблицам.

$$x = x^{-1} \; mod \; m(x).$$

Если x = 0, то обратного не существует и замена не производится.

Вычисляется:

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

Или для каждого бита y:

$$y_i = x_i \oplus x_{(i+4) \, mod \, 8} \oplus x_{(i+5) \, mod \, 8} \oplus x_{(i+6) \, mod \, 8} \oplus x_{(i+7) \, mod \, 8} \oplus c_i,$$

где $c = \{01100011\}$.

Байт y записывается вместо x.

Также значение байта после замены может быть определено по табл. 10 [7]. По вертикали указана первая шестнадцатеричная цифра байта, по горизонтали – вторая, внутри таблицы – замена для этого байта.

Таблица 10

Таблица замен, для байта {mn}

m\n	0	1	2	3	4	5	6	7	8	9	a	b	c	D	E	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	Fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	B2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	Fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	Ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	Bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Обратная операция также может быть выполнена по таблице замены или описана математически. Все необходимые таблицы приведены в [7].

Шаг 2. ShiftRows.

Циклический сдвиг вправо каждой строки матрицы на число байт, зависящее от номера строки и размера блока. Для длины блоков от 128 до 196 бит первая строка не сдвигается, вторая строка сдвигается на 1 байт, третья на 2, четвёртая на 3. Для длины блока 256 бит третья строка сдвигается на 3 байта, четвёртая на 4 байта.

Для рассматриваемой версии выполняется замена:

$$\begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} \rightarrow \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{pmatrix}$$

Обратная операция выполняется обратным сдвигом.

Шаг 3. MixColumn

Операция MixColumn, как можно догадаться из названия, выполняется по столбцам. Каждый столбец представляется, как многочлен степени 3, коэффициентами которого являются байты столбца, представленные в виде полиномов в поле $GF(2^8)$:

$$A(X) = a_0 X^3 + a_1 X^2 + a_2 X + a_3 .$$

a_0 соответствует $s_{0,?}$, a_3 соответствует $s_{3,?}$.

Фиксированный многочлен $C(x)$, также с коэффициентами из $GF(2^8)$:

$$C(X) = \{03\}X^3 + \{01\}X^2 + \{01\}X + \{02\}.$$

Здесь запись $\{b\}$ соответствует многочлену из $GF(2^8)$, который представим в виде числа b. $\{03\}$ соответствует x+1, {02} – x, {01} – 1.

Новое значение столбца соответствует произведению

$$B(X) = A(X) \cdot C(X) \; mod \; (X^4 + 1).$$

Коэффициенты $B(X)$ являются новыми значениями байтов столбца. При приведении по модулю $(X^4 + 1)$, коэффициенты при X^6, X^5 и X^4 могут быть прибавлены к коэффициентам при X^2, X и 1 соответственно. Вычитание коэффициентов из $GF(2^8)$ соответствует сложению.

Это же выражение, уже приведенное по модулю можно записать в матричном виде:

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

Для обращения этой операции, необходимо использовать многочлен обратный C(X) или обратную матрицу:

$$A(X) = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \{0e\} & \{0b\} & \{0d\} & \{09\} \\ \{09\} & \{0e\} & \{0b\} & \{0d\} \\ \{0d\} & \{09\} & \{0e\} & \{0b\} \\ \{0b\} & \{0d\} & \{09\} & \{0e\} \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Результаты умножений приводятся по модулю m(x).

Шаг 4. AddRoundKey

Текущее состояние складывается с ключом раунда.

$$\begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} \oplus \begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix}.$$

Ключ раунда получают из ключа шифрования, применением процедуры «разворачивание ключа» (key schedule).

Разворачивание ключа

На основе имеющегося ключа (произвольной длины более 128 бит) генерируется ключевая последовательность для всех раундов.

При генерации ключа используется операция SubBytes, определенная выше. RotBytes, циклически сдвигающая 32 битное слово влево.

Константа раунда: $RC_i = [x^{i-1} \bmod m(x), \{00\}, \{00\}, \{00\}]$, где $x =$ {02}, i – номер раунда.

Вычисления производятся по столбцам, и словом ключа является столбец.

Приведем псевдокод, показывающий процесс генерации ключей. Будем обозначать K – исходный ключ, W – генерируемый ключ, Nc – число столбцов ключа, Nn – число раундов.

```
for (kc=0; i<= Nc; kc++)  { W[kc]=K[kc]; }
for (i=1; i<= Nn; i++)
{
   T=RotBytes(W[Nc*i-1]);
   T=SubBytes(T);
   T=T⊕RC[i] ;
   W[Nc*i]=W[Nc*(i-1)] ⊕ T;     // W[4]=W[0]+T
   for (kc=1; i<=Nc; kc++){
        // W[5]=W[1]+W[4]
        W[Nc*i+kc]=W[Nc*(i-1)+kc] ⊕ W[Nc*i+kc-1] ;
   }
}
```

Благодаря своей четкой математической структуре алгоритм AES доступен для исследования математическими методами. Такие исследования могут как показать надежность алгоритма и отсутствие в нем закладок, так и найти методы его взлома. Этот алгоритм вызывает больше всего опасений в связи с возможностью алгебраических атак.

ГОСТ 28147-89

ГОСТ 28147-89 – советский и российский стандарт симметричного шифрования, введённый в 1990 г. Полное название – «ГОСТ 28147-89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования».

С момента опубликования ГОСТа на нём стоял ограничительный гриф «Для служебного пользования», и формально шифр был объявлен «полностью открытым» только в мае 1994 г.

Описание алгоритма

ГОСТ 28147-89 – блочный шифр с 256-битным ключом и 32 циклами преобразования, оперирующий 64-битными блоками. Основа алгоритма шифра – Сеть Фейстеля. На рис. 32 представлена схема одного раунда шифрования.

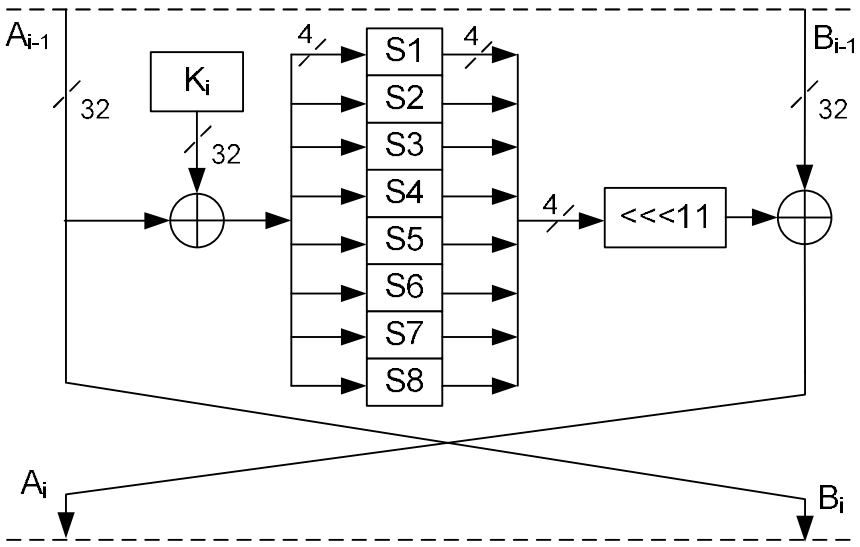


Рис. 32. Раунд алгоритма ГОСТ 28147-89

Ключ K разбивается на восемь 32-разрядных подключей $\{X_0,\ldots,X_7\}$. Блок открытого текста разбивается на два 32-разрядных подблока A и B. По ГОСТ они называются N_1 и N_2 и здесь переименованы, чтобы не путать индексы.

Для i от 1 до 32 выполняется:

1. В раунде шифрования i, значение A_i суммируется по модулю 2^{32} с подключом K_i.

$$t = A_i + K_i \ mod\, 2^{32}.$$

2. Результат суммирования преобразуется в блоке подстановки S. При выполнении замены, блок данных разбивается на 8 фрагментов по 4 бита, которые параллельно пропускаются через 8 различных таблиц замены $S1\ldots S8$. Таблица замены содержит в определенной последовательности значения от 0 до 15 (то есть все варианты значений 4-битного фрагмента данных); на вход таблицы подается блок данных, числовое представление которого определяет номер выходного значения. Например, если подается значение 5 на вход следующей таблицы: {13 0 11 74 9 1 10 14 3 5 12 2 15 8 6}, то на выходе будет значение 9 (поскольку 0 заменяется на 13, 1 – на 0, 2 – на 11 и т. д.). Таблицы замены не определены в ГОСТ и являются долговременными ключами.

3. Полученный вектор циклически сдвигается на одиннадцать бит влево.

$$t = S(t) << 11.$$

4. Результат сдвига суммируется поразрядно по модулю 2 с 32-разрядным значением N_2.

$$t = t \oplus B_i.$$

5. Значение A_i записывается в B_{i+1}; а результат суммирования в A_{i+1}

$$B_{i+1} = A_i;\ A_{i+1} = t.$$

В последнем 32 цикле A_i не изменяется, а результат заносится в B_i:

$$B_{32} = t;\ A_{32} = A_{31}.$$

Общая схема алгоритма соответствует схеме сети Фейстеля приведенной на рис. 31.

Подключи $\{K_1,\ldots,K_{32}\}$ соответствуют $\{X_0,\ldots,X_7\}$ по следующему правилу:

для $i = 1 \div 24$: $K_i = X_{(i-1) \bmod 8}$.

для $i = 25 \div 32$: $K_i = X_{(32-i+1)}$.

То есть для первых 24 раундов ключи выбираются циклически в прямом порядке, для последних 8 – в обратном порядке.

При расшифровании выполняется тот же алгоритм, но с обратной последовательностью ключевых элементов.

Режим простой замены (электронной кодовой книги), согласно ГОСТ может использоваться только для шифрования ключей. Для шифрования данных, ГОСТом предусмотрены режимы гаммирования и гаммирования с обратной связью. Их реализация несколько отличается от одноименных режимов рассмотренных выше. Также ГОСТом предусмотрен режим выработки имитовставки (кода аутентификации сообщений).

На основе данного алгоритма построен отечественный стандарт хеширования ГОСТ Р 34.11-94.

Режим гаммирования по ГОСТ

В данных режимах шифрование информации производится побитовым сложением по модулю 2 каждого 64-битного блока шифруемой информации с блоком гаммы. Гамма шифра – это псевдослучайная последовательность, вырабатываемая с использованием основного преобразования алгоритма ГОСТ 28147-89 (рис. 33).

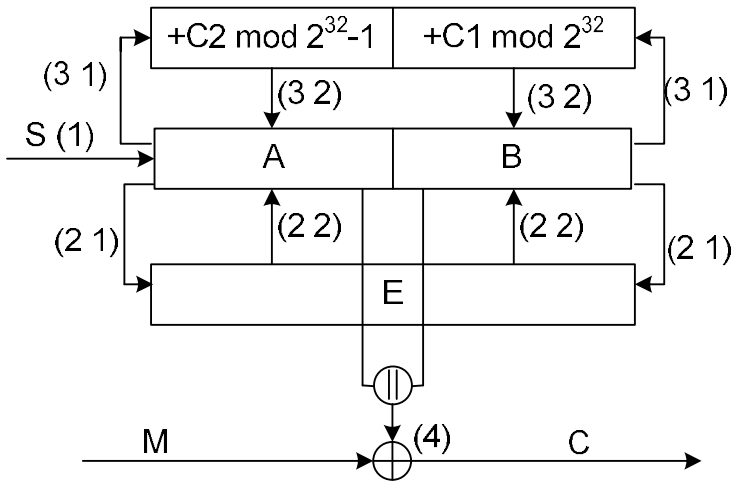


Рис. 33. Режим гаммирования

На рис. 33 цифрами обозначен порядок операций для первого блока знаком ⫴ обозначено объединение строк (регистров). Для первого блока данных генерация гаммы осуществляется в следующем порядке.

1. В регистры A и B записывается их начальное заполнение – 64-битная величина S, называемая «синхропосылкой».

2. Выполняется шифрование содержимого регистров A и B в режиме простой замены.

$$A||B = E_K(A||B).$$

3. Содержимое B складывается по модулю ($2^{32} - 1$) с константой $C_1 = 2^{24} + 2^{16} + 2^8 + 2^2$, результат сложения записывается в B. Содержимое A складывается по модулю 2^{32} с константой $C_2 = 2^{24} + 2^{16} + 2^8 + 1$, результат сложения записывается в A.

4. Содержимое регистров $A||B$ подается на выход в качестве 64-битного блока гаммы и складывается с блоком открытого текста.

Для следующих блоков данных, порядок шагов генерации гаммы меняется. Вначале выполняется сложение с константами (шаг 3), затем шифрование (шаг 2) и сложение с блоком открытого текста (шаг 4).

Снхропосылка в алгоритме может быть как секретным элементом, так и открытым. Существуют реализации алгоритма ГОСТ 28147-89, в которых синхропосылка также является секретным элементом, наряду с ключом шифрования. В этом случае можно считать, что ключ шифрования увеличивается на длину синхропосылки (64 бита), что увеличивает стойкость.

Данный режим почти полностью повторяет режим OFB, все его недостатки и преимущества. Дополнительным преимуществом может быть различие в шифровании первого блока и всех последующих.

Режим гаммирования с обратной связью по ГОСТ полностью совпадает с режимом CFB, примененным для базового алгоритма.

Достоинства ГОСТа

• При разработке алгоритма ГОСТ 28147-89 криптографами отечественных спецслужб была заложена избыточная стойкость – до сих пор не известны более эффективные методы взлома данного алгоритма, чем метод полного перебора возможных вариантов ключей шифрования. А полный перебор 2^{256} ключей (не считая секретной синхропосылки) при современном развитии компьютерной техники за реальное время осуществить невозможно.

• Алгоритм ориентирован на использование 32-битных слов, что позволяет выполнять быстрые аппаратные и программные реализации. Только для выполнения замен необходимо оперировать с 4-битными блоками.

• Возможно дополнительное повышение секретности, за счет сохранения в тайне блоков замен (долговременного ключа).

Недостатки ГОСТа

Основные проблемы ГОСТа связаны с неполнотой стандарта в части генерации ключей и таблиц замен. Тривиально доказывается, что у ГОСТа существуют «слабые» ключи и таблицы замен, но в стандарте не описываются критерии выбора и отсева «слабых». Отсутствие фиксированных таблиц замен поднимает ряд проблем:

- нельзя определить криптостойкость алгоритма, не зная заранее таблиц замен;
- реализации алгоритма от различных производителей могут использовать разные таблицы замен и могут быть несовместимы между собой;
- возможность преднамеренного предоставления слабых таблиц замен лицензирующими органами РФ.

Другим недостатком является слишком простая структура шифра и малый размер S блоков.

Многократное шифрование

Шифротексты образуют группу, если для алгоритма E, для любых ключей K_1 и K_2, существует ключ K, такой, что для любого открытого текста X выполняется $E_{K2}(E_{k1}(X)) = E_K(X)$.

Если шифротексты не образуют группу, то шифрование шифротекста позволяет повысить стойкость шифра. Если для шифрования используется один и тот же алгоритм, то шифрование называют многократным. Если различные – то каскадным.

Использовать двойное шифрование не рекомендуется, так как для него возможно использование вскрытия «встреча посередине» [11].

Для вскрытия «встреча посередине» необходимо иметь пару открытый текст – шифротекст (такое количество пар блоков, которое однозначно определит ключ).

Пусть криптоаналитику известны $P_1, C_1, P_2 \, и \, C_2$, такие, что:

$$C_1 = E_{K2}(E_{K1}(P_1)),$$
$$C_2 = E_{K2}(E_{K1}(P_2)).$$

Тогда для шифротекста C_1 вычисляем все возможные открытые тексты $E_{K1}(P_1)$: для всех $K2$ вычисляем $D_{K2}(C_1)$ и запоминаем. Для открытого текста P_1 вычисляем все шифротексты $D_{K2}(C_1)$: для каждого $K1$ вычисляем $E_{K1}(P_1)$ и ищем совпадения в таблице $D_{K2}(C_1)$. Если совпадение найдено, то возможно пара $K1$, $K2$ является ключами. Для того чтобы это проверить, можно вычислить $E_{K2}(E_{K1}(P_2))$ и сравнить с C_2 или попытаться расшифровать фрагмент текста. Если эта пара не является ключевой, то необходимо продолжать поиск. Надо заме-

тить, что уже при 64 битном ключе и блоке, размер таблицы составит $8*2^{64}$ байт или 128 млн. терабайт.

Также существуют способы вскрытия схем вида

$$C = E_{K1}(E_{K2}(E_{K1}(P_1))),$$

но они требуют еще большего количества памяти.

Значительно более надежной является схема с тремя различными ключами.

$$C = E_{K3}(E_{K2}(E_{K1}(P_1))).$$

Для 3DES был определен порядок шифрование – расшифрование – шифрование:

$$C = E_{K3}(D_{K2}(E_{K1}(P_1))).$$

Такая схема не имеет особых преимуществ, но используется для совместимости с однократным шифрованием. Если $K1 = K2$, то $C = E_{K3}(P_1)$.

Вскрытие трехкратного шифрования по схеме встреча посередине требует хранения 2^n шифротекстов и 2^{2n} вычислений (или наоборот).

Для усложнения криптоанализа, можно после каждого шифрования добавлять вставку, длиной, не кратной длине блока. В этом случае один блок открытого текста будет влиять на три блока шифротекста. Аналогично, может применяться перемешивание по большим блокам [11].

В случае вставок уравнение шифрования будет иметь вид:

$$C = E_{K3}(pad \,||\, D_{K2}(pad \,||\, E_{K1}(P))).$$

Вставка pad должна быть такой, чтобы ее снятие было однозначным, либо случайной, но известной длины.

На рис. 34 показано распространение зависимости шифротекста от открытого текста. При изменении одного бита открытого текста, изменится три блока шифротекста. Аналогично и в обратную сторону: блок (каждый бит блока) шифротекста зависит от трех блоков открытого текста.

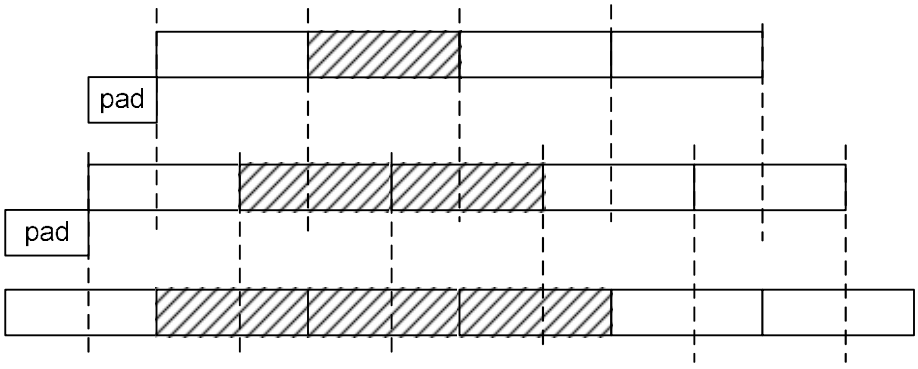


Рис. 34. Распространение зависимостей при смещении блоков.

Если есть опасения по поводу стойкости того или иного алгоритма, то целесообразно применять при каждом шифровании различные алгоритмы. Стойкость такого шифрования будет не менее стойкости каждого алгоритма.

Безопасность блочных алгоритмов

Блок открытого текста или шифротекста в общем виде может рассматриваться как один символ, а блочный шифр – как шифр замены. Соответственно возможное количество различных блоков открытого текста или размер алфавита для блоков длины n равен 2^n. Количество возможных таблиц замен равно $2^n!$.

Таблица замены определяется ключом K, длиной s бит. Пространство возможных ключей 2^s определяет лишь малую часть возможных таблиц замен (2^s несравнимо меньше $2^n!$, для любых разумных n и s). Множество таблиц замен определяется алгоритмом и теоретически доступно для криптоаналитика, но практически должно быть недоступно построение полностью даже одной такой таблицы. Практически криптоаналитик может построить только отдельные фрагменты таких таблиц для отдельных открытых текстов и отдельных ключей (если позволяют вычислительные возможности – то достаточно большие). Хорошо спроектированный блочный шифр не должен позволить найти закономерности между шифротекстами, полученными из одного открытого текста с помощью различных ключей; или полученными из различных открытых текстов, с помощью одного ключа. Малое изменение ключа или открытого текста должно приводить к значительным изменениям в шифротексте и эти изменения не должны подчиняться *простым соотношениям* [11]. Все пространство ключей должно определять множество таблиц замен, выглядящее статистически случайным. В этом случае для поиска ключа по известному открытому тексту может понадобиться полный перебор ключей.

Например, для DES выполняется простое соотношение:

Если $E_K(m) = C$, то $E_{K'}(m') = C'$, где K', m', и C' – побитовые дополнения K, m, и C. Побитовое дополнение – это отрицание, заменяющее 1 на 0, а 0 на 1. Такое простое соотношение позволяет в некоторых случаях вдвое уменьшить сложность атаки.

Простые соотношения можно представить в виде:

Если $E_K(m) = C$, то $E_{f(K)}(g(m,K) = h(C,k)$, где f, g и h – функции, вычисляемые значительно более быстро, чем функция шифрования.

Также необходимо, чтобы шифротексты, полученные с помощью различных ключей, не образовывали группу. Чтобы композиция двух шифротекстов не давала третий, и последовательное шифрование двумя ключами не давало такой шифротекст, который можно получить одним шифрованием.

Существует несколько общих методик, которые можно применять при взломе блочного шифра, например, полный перебор, опирающийся на предварительно вычисленные таблицы промежуточных параметров, или прием, условно называемый «разделяй и властвуй». Некоторые (не-

удачно спроектированные) блочные шифры могут оказаться беззащитными перед атакой с выбором открытого текста, где шифрование специально выбранного сообщения может выявить важные свойства секретного ключа.

Криптоанализ обычно делят на дифференциальный и линейный. В дифференциальном криптоанализе изучаются пары шифротекстов, исходные сообщения в которых имеют специфические различия. Результат применения логической операции исключающего ИЛИ к таким парам называется дифференциалом. Дифференциалы в зависимости от использованного ключа обладают теми или иными характерными свойствами. Если исследовать частоту появления дифференциалов, вычисленных при атаке с выбором открытого текста, то есть вероятность выявить основную структуру ключа. Линейный криптоанализ состоит в аппроксимации нелинейных компонент линейными функциями. А далее также по распределению вероятностей, получить информацию о ключе.

Важным параметром безопасности являются S-блоки (блоки замены), используемые почти во всех шифрах. Необходимо выбирать S-блоки так, чтобы при изменении одного бита на входе, на выходе изменялась половина бит, чтобы защититься от дифференциального криптоанализа. Также желательно использовать блоки большей длины, при этом случайно выбранные таблицы замены могут иметь достаточно хорошие характеристики. Если n – размер блока замены, то таблица должна содержать 2^n записей и иметь размер $n*2^n$ бит. Для аппаратной реализации использование n более 12 – проблематично. Для программной – могут использоваться и 16 битные блоки (таблица будет занимать 128 кбайт).

Относительно новым направлением криптоанализа являются алгебраические атаки или XSL-атаки. Цель таких атак – составить систему уравнений, в которой неизвестными являются элементы ключа, и попытаться решить их.

Очень долгое время такие атаки считали невозможным, однако в 2006 г. появились сообщения о возможностях взлома некоторого количества раундов различных шифров. Причем как с четкой алгебраической структурой (AES), так и не имеющих четкой структуры (DES). Ранее считалось, что XSL атакам в основном подвержены алгебраические шифры и основные опасения были связаны с использованием AES. Уязвимость DES с точки зрения XSL атак заключается в малом размере блоков замены. Причем от самих блоков успех мало зависит. Любые блоки данного размера могут быть взломаны для упрощенных шифров. Учитывая то, что алгоритм ГОСТ очень похож на DES, то можно опасаться и его взлома. 32 раунда ГОСТ и 256 бит ключа уже вовсе не кажутся паранойей.

Нет никаких оснований считать, что эти шифры еще не взломаны и секретами вскрытия этих алгоритмов не обладают спецслужбы.

Контрольные вопросы

1. Для каких целей используются различные режимы шифрования?
2. Почему шифры, основанные на сетях Фейстеля, оказываются обратимы, независимо от внутренней функции?
3. Оцените количество пар ключей, которые могут быть получены при атаке «встреча посередине» для одного блока, в зависимости от длин ключей и блоков. Все неопределенные распределения считайте равномерными.

ТЕМА 10. ВВЕДЕНИЕ В ПРОТОКОЛЫ

Общие сведения

Протокол – это полностью определенная процедура, выполняемая несколькими участниками, предназначенная для решения определенной задачи [11, 12].

- В протоколе участвует не менее двух сторон. Верхняя граница не ограничена (это могут быть, например, все пользователи некоторого сайта или системы распределенных вычислений).
- Каждый участник протокола должен знать протокол и последовательность составляющих его действий.
- Протокол должен быть непротиворечивым. Каждое действие должно быть определено так, что бы оно было понято однозначно.
- Протокол должен быть полным. Каждой ситуации должно соответствовать определенное действие.
- В криптографических протоколах в общем случае предполагается, что пользователи не доверяют друг другу. Необходимо учитывать, что один или несколько пользователей сообща могут попытаться нарушить работу протокола.
- В работу протокола могут вмешиваться незаконные пользователи, как выдавая себя за законных, так и подключаясь к линиям связи. При проектировании или анализе протоколов считают, что злоумышленник может не только слушать канал связи, но и модифицировать его.

Целью криптографического протокола является обеспечение возможности и безопасности выполнения санкционированных действий, и невозможности выполнения действий не определенных в протоколе.

Участники протоколов

Как правило, в протоколе участвует две **основные стороны**, которые собственно хотят выполнить какие либо действия и быть при этом уверены в своей безопасности. Это могут быть два пользователя обме-

нивающиеся секретными сообщениями или клиент и сервер платежной системы. Основных сторон может быть и больше, но такие протоколы встречаются реже. Основные участники, как и в большинстве книг по криптографии будут именоваться Алиса и Боб.

Так же часто в протоколе участвуют вспомогательные стороны, посредники или арбитры.

Типы протоколов

Можно выделить несколько основных типов протоколов.

Самодостаточные протоколы

В самодостаточных протоколах участвуют только стороны непосредственно заинтересованные в результате. Участие дополнительных сторон не требуется (рис. 35). Если одна из сторон попытается обмануть другую, то это станет известно другой стороне из проверочных действий предусмотренных протоколом.

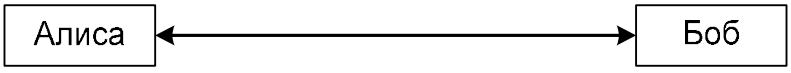


Рис. 35. Самодостаточный протокол

Протоколы с посредником

От посредников ожидается независимость поведения, следующая из его незаинтересованности в результатах протокола (в случае если злоумышленник не выдает себя за посредника). Посредники помогают реализовывать протоколы в случаях, когда основные стороны не доверяют друг другу (рис. 36).

Рис. 36. Протокол с посредником

Существует несколько проблем связанных с использованием посредников.

• Если две стороны относятся друг к другу с подозрением и хотят быть защищенными, то они с таким же подозрением, если не большим, будут относиться к любой третьей стороне.

• Если каждый участник доверяет посреднику и протокол не предусматривает проверки его действий, то посредник становится узким местом протокола при попытке его взлома.

В реальном мире, аналогией посредника может быть юрист. Например, юрист может присутствовать при подписании протокола или проверять документы и платежеспособность клиентов при совершении сделки купли-продажи.

Посредниками являются компании подтверждающие, выдающие, продлевающие и отзывающие сертификаты (например, SSL-сертификаты, необходимые для организации HTTPS соединений)

Протоколы с арбитром

Протоколы с арбитром состоят из двух протоколов: из протокола без посредника и протокола разрешения споров. Вторая часть выполняется только в случае появления разногласий после выполнения первой части.

Во время первой части протокола стороны накапливают доказательства выполнения другой стороной тех или иных действий, которые могут быть предоставлены арбитру (рис. 37).

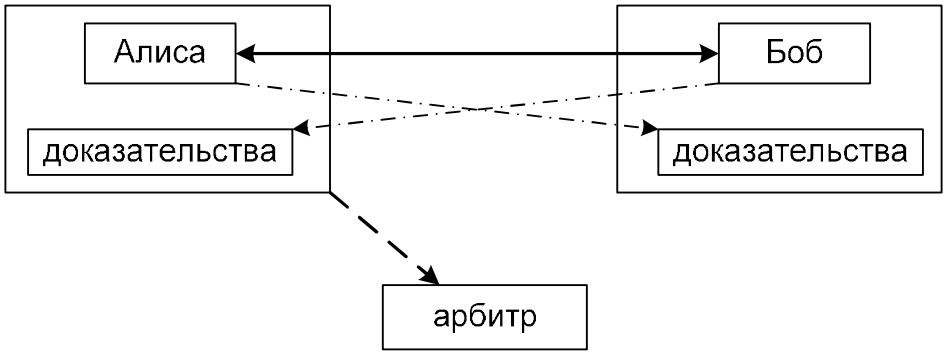


Рис. 37. Протокол с арбитром

Рассмотрим протокол подписания контракта с возможным участием арбитра [11].

1. Алиса и Боб договариваются об условиях контракта
2. Алиса подписывает контракт (у Боба появляются доказательства подписания контракта Алисой)
3. Боб подписывает контракт (у Алисы появляются доказательства подписания контракта Бобом)

В случае если появляются разногласия, одна из сторон может обратиться к судье, чтобы уличить другую в мошенничестве.

4. Алиса и Боб предстают пред судьей и предоставляют свои доказательства.

5. Судья принимает решение на основании доказательств.

Протоколы с арбитром могут быть как автоматические, когда система накапливает все данные необходимые для поиска мошенников и сама проверяет пользователей по запросу или после выполнения ими действий в системе; так и автоматизированные, когда окончательное решение принимает человек.

Возможности злоумышленников и типы атак

Попытки взлома протокола могут быть направлены против криптографических алгоритмов, используемых в протоколах, методов их реализации или собственно против протоколов.

При анализе протоколов предполагают, что используются идеальные невскрываемые алгоритмы и методы. Ошибки ищут при этом только в логике работы протоколов.

Может быть использовано множество различных способов вскрытия протокола.

Злоумышленник может прослушивать отдельные или все каналы связи и анализировать перехваченные данные. Такой тип атаки называется **пассивным.** Большинство криптографических протоколов могут успешно противостоять таким атакам.

В других случаях злоумышленник может не только слушать каналы связи, но и активно влиять на ход выполнения протокола. Он может выдавать себя за другого, передавать дополнительные сообщения, подменять в каналах связи сообщения других пользователей своими, разорвать канал связи, выступить в роли посредника, повторить ранее переданные сообщения. Такие виды атак называются **активными**. Активный взломщик не обязательно является посторонним по отношению к системе. Это может быть ее зарегистрированный пользователь или даже один из администраторов управляющей системы.

Если взломщиком является один из участников протокола, то такой взломщик называется мошенником. Пассивный мошенник следует всем правилам протокола, но все полученные им данные подвергает криптоанализу, с целью получения дополнительных сведений. Активный мошенник пытается изменить выполнение протокола и посылает модифицированные сообщения.

Чем больше количество мошенников и взломщиков, тем сложнее спроектировать протокол, функционирующий правильно в таких условиях.

ТЕМА 11. ОДНОНАПРАВЛЕННЫЕ ХЭШ-ФУНКЦИИ

Протокол подбрасывания монеты по телефону

Чтобы перейти к понятию однонаправленных функций, рассмотрим протокол подбрасывания монеты по телефону [12].

Алиса и Боб не могут договориться и решили положиться на волю случая. Если они находятся рядом, то они могут просто подбросить монетку:

1. Боб выбирает сторону – «орел» или «решка».
2. Алиса подкидывает монетку.
3. Алиса и Боб смотрят на результат и принимают решение в зависимости от результата.

Теперь предположим, что Алиса и Боб пытаются выполнить этот протокол по телефону. Если Боб скажет свой выбор, то он не сможет быть уверен, что Алиса честно сказала результат подбрасывания, если вначале Алиса подбросит монету и скажет результат Бобу, то она не будет уверена, что Боб не изменил своего решения, узнав результат подбрасывания.

Для того, что бы реализовать протокол по телефону, необходимо подтверждение неизменности выбора с момента его совершения, до момента разглашения результата. Эту задачу можно выполнить, сделав выбор бесконечным, например, сопоставив понятие «решка» – множеству нечетных чисел, а «орел» – множеству четных и использовав криптографическую функцию, отвечающую следующим требованиям.

- Для любого x легко вычислить f(x), но если дано f(x) – невозможно вычислить x или какую либо информацию о нем (в данном случае – узнать четное x или нечетное).
- Невозможно найти пару $x \neq y$, такую, что $f(x) = f(y)$.

Теперь протокол подбрасывания монеты по телефону можно записать в следующем виде.

1. Алиса выбирает большое x, вычисляет f(x) и сообщает результат функции Бобу (подкидывает монету и фиксирует результат, не сообщая сам результат).
2. Боб делает догадку о x – четное оно или не четное.
3. Алиса раскрывает истинное значение x Бобу.
4. Боб может проверить, значение f(x). Сравнить полученное на шаге 1 с вычисленным от x, которое получено на шаге 3.

Здесь первое требование к f(x) определяет то, что Боб не может вычислить x, зная f(x) и тем самым раньше времени определить, его четность. Второе требование определяет, что Алиса не может найти два числа x и y (одно четное, другое нечетное), такие, что $f(x) = f(y)$ и выбрать на 3-м шаге, которое из чисел раскрывать.

Понятие невозможности здесь дано применительно к протоколу (все криптографические функции рассматриваются как совершенные).

Виды однонаправленных функций

Под односторонней (однонаправленной) функцией понимается эффективно вычислимая функция, для обращения которой (то есть для поиска хотя бы одного значения аргумента по заданному значению функции) не существует эффективных алгоритмов [15].

Хеш-функция – функция, отображающая строку произвольной длины в строку фиксированной длины. Далее под хеш-функциями будут пониматься исключительно криптографические хеш-функции. Примером некриптографических хеш-функций могут быть различные алгоритмы вычисления контрольных сумм (например, CRC) которые позволяют обнаружить случайные изменения, но не защищают от намеренных фальсификаций.

Криптографическая хеш-функция должна обладать следующими свойствами [11].

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

2. Защищенностью от коллизий. **Коллизией** для хеш-функции h называется такая пара $x \neq y$, что $f(x) = f(y)$. Вычислительно невозможно найти два сообщения с одним и тем же значением хэш-функции.

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

Задачи восстановления прообразов и вторых прообразов вычислительно более сложные, чем задача поиска коллизий, поэтому требование невозможности нахождения коллизий является более сильным. Далее будет рассмотрен эффективный способ поиска коллизий.

В криптографических протоколах, для одностороннего преобразования, обычно используются именно хеш-функции. По сравнению с односторонними функциями, которые не скрывают длины исходной строки, можно выделить преимущества:

- Обычно удобнее работать со строками фиксированной длины
- Результат хеш-функции не позволяет узнать даже длины сообщения.

Коллизии

На первый взгляд, нахождение коллизий – процесс не менее сложный, чем обращение функции. Действительно, нахождение обратного значения, для определенного значения h(x), если в алгоритме нет уязвимостей, требует перебора открытых текстов. Количество текстов, которые нужно проверить соответствует 2^n, где n – длина выхода хеш-функции. Но если необходимо только найти коллизию между двумя сообщениями, то необходимо проверить $2^{\frac{n}{2}}$ открытых текстов.

Парадокс дней рождения

Это различие в количестве требуемых попыток носит название «Парадокс дней рождения» и может быть описано так.

Сколько человек нужно собрать, что бы с вероятностью в 50 % хотя бы у одного из них день рождения совпадал с вашим? Ответ – 365/2 ≈ 183. Но для того, что бы хотя бы у двух человек из группы с вероятностью в 50 % совпадал день рождения, достаточно собрать 23 человека.

С ростом числа возможных вариантов можно ожидать появлений совпадений после $\sqrt{N}$ проверок, где N – количество возможных вариантов.

Применительно к хеш-функциям из парадокса дней рождения следует, что если для $2^{\frac{n}{2}}$ различных текстов вычислить хеш-функции и записать, то можно ожидать появления коллизий.

Атака на основе парадокса дней рождений

В описании протокола подбрасывания монеты уже было сказано, что если Алиса знает два прообраза (четный и нечетный) с одинаковым хеш-значением, то она, после догадки Боба может выбрать одно из двух значений. Опишем подробнее более серьезный пример.

Пусть Алиса хочет подписать контракт с Бобом. Она готовит контракт и передает его Бобу. Он соглашается, вычисляет хеш-функцию и подписывает выход хеш-функции электронно-цифровой подписью. Но если Алиса хочет обмануть Боба, она может выполнить следующие шаги.

1. Алиса готовит две версии контракта: одну, выгодную для Боба, которую он согласится подписать и другую, приводящую его к банкротству

2. Алиса вносит несколько незначительных изменений в каждый документ и вычисляет хэш-функции. Эти изменения могут быть косметическими или вовсе невидимыми: замена одного пробела двумя в конце строки, замена перевода строки #0d0a на #0a0d, и т. д., в зависимости от формата документа. Делая или не делая по одному изменению в каждой из 32 строк, Алиса может легко получить 2^{32} различных документов.

3. Алиса сравнивает хэш-значения для каждого изменения в каждом из двух документов, разыскивая пару, из подлинного контракта и поддельного, для которой эти значения совпадают. Если выходом хэш-функции является 64-разрядное значение, Алиса, с большой долей вероятности сможет найти совпадающую пару, сравнив 2^{32} версий каждого документа.

4. Алиса получает подписанную Бобом выгодную для него версию контракта, используя протокол, в котором он подписывает только хэш-значение.

5. Спустя некоторое время Алиса подменяет контракт, подписанный Бобом, другим, который он не подписывал. Теперь она может убедить арбитра в том, что Боб подписал другой контракт.

Простой защитой от такого вида атаки может быть незначительное косметическое изменение текста перед подписанием. В этом случае Алисе придется искать уже не коллизию, а второй прообраз, что требует уже не 2^{32} вычислений, а 2^{64}.

Если же два различных документа с одинаковыми хеш-значениями находит Боб, то он в дальнейшем так же может отказаться от своих обязательств, предоставив ни к чему не обязывающий документ с такой же подписью, а настоящий назвать махинациями Алисы.

Сегодня 64-битные хеш-функции почти не используются, 128-битные функции еще некоторое время можно считать безопасными, если злоумышленник не может позволить себе загрузить вычислениями миллионы компьютеров в течение месяцев.

Так же в некоторых алгоритмах могут присутствовать уязвимости, позволяющие упростить поиск коллизий.

Применение хеш-функций

Можно выделить несколько классов задач, в которых используются хеш-функции.

Контроль целостности (контрольная сумма): для файлов вычисляется хеш-функция (часто используется md5) и ее значение хранится отдельно от файлов. Различные антивирусные приложения могут вычислять контрольные суммы исполняемых файлов и хранить в своей базе данных, что бы периодически проверять неизменность файлов. Также публикация хеш-значений файлов характерна для свободного программного обеспечения. Сам файл может быть получен из любого источника, которому не всегда можно доверять, но на сайте автора можно найти контрольные суммы и сравнить с вычисленными для имеющегося файла.

Коды аутентификации сообщений (MAC – Message authentication code): если при передаче, необходимо обеспечить только неизменность сообщений, то перед передачей вычисляется хеш-функция от пакета данных и секретного ключа известного отправителю и получателю и добавляется после пакета данных. См. тему 12. Коды аутентичности сообщений.

Цифровые подписи: если сообщение велико, то вместо него в схемах подписи используется хеш-значение. Кроме того, в некоторых алгоритмах ЭЦП подпись самого текста может быть не безопасна. См. тему 15. Электронно-Цифровые подписи.

В протоколах аутентификации, вместо пароля: хранение пароля на сервере или передача его в открытом виде, позволяют злоумышленнику легко получить к нему доступ. В некоторых схемах аутентификации вместо пароля хранится его хеш-значение. См. тему 17. Протоколы аутентификации.

Семейство алгоритмов MD4

Наиболее популярные на сегодняшний день алгоритмы хеширования имеют похожую структуру. Первым алгоритмом из этой группы был алгоритм MD4 (Message Digest 4) разработанный Роном Ривестом в 1992 г. Данный алгоритм используется в операционных системах MS Windows для хранения хеша пароля.

Приведем алгоритмы, которые можно отнести к этому семейству [8].

- MD4 – состоит из 3 раундов по 16 шагов в каждом. Длина выхода – 128 бит.
- MD5 – добавлен 4-й раунд и немного изменен алгоритм.
- SHA-1 – состоит из 4 раундов по 20 шагов в каждом. Длина выхода – 160 бит.
- RIPEMD-160 – состоит из 5 раундов по 16 шагов в каждом Длина выхода – 160 бит (также версии 128, 256, 320).
- SHA-256 – состоит из 64 одношаговых раундов. Длина выхода – 256 битов.
- SHA-512 – состоит из 80 одношаговых раундов. Длина выхода – 512 бит.
- SHA-384 – практически идентичен SHA-512 за исключением того, что его выход урезан до 384 бит.

Алгоритм MD4

Ривест описал цели, преследуемые им при разработке алгоритма MD4 [11].

- Прямая безопасность. Безопасность MD4 не основывается на каких-либо допущениях, например, предположении о трудности разложения на множители.
- Скорость. MD4 подходит для высокоскоростных программных реализаций. Она основана на простом наборе битовых манипуляций с 32-битовыми операндами.
- Простота и компактность. MD4 проста, насколько это возможно, и не содержит больших структур данных или сложных программных модулей.
- Удачная архитектура. MD4 оптимизирована для микропроцессорной архитектуры. Все структуры 32-битовые, все операции побитовые, либо сложение.

Рассмотрим подробно алгоритм MD4 [4].

Пусть имеется сообщение длиной b бит. Для вычисления хеш-значения (дайджеста) применяются пять шагов.

Шаг 1. Строка дополняется битами, до длины конгруэнтной 448 по модулю 512.

$$b` = 448 \ mod \ 512.$$

В начале сообщение дополняется одним битом «1», а затем последовательностью из «0» до нужной длины. Причем дополняется минимум один бит, максимум 512. Если длина сообщения уже равна 448 mod 512, то дополнение все равно происходит и дополняются 512 бит.

Шаг 2. Строка дополняется 64 битовым представлением исходной длины сообщения. Младшее 32-битовое слово при этом идет впереди. Порядок байтов также от меньших к старшим.

Например, если исходная длина сообщения 127 бит (7Fh), то на этом шаге дополнение будет иметь вид 7F 00 00 00 00 00 00 00.

После этого шага длина сообщения кратна 512 битам.

Шаг 3. Инициализация буфера, состоящего из четырех 32-битовых регистров A, B, C, D.

```
A: 67 45 23 01
B: ef cd ab 89
C: 98 ba dc fe
D: 10 32 54 76
```

Шаг 4. Определим основные используемые побитовые функции.

$$F(X,Y,Z) = (X \wedge Y) \vee (\overline{X} \wedge Z)$$

$$G(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$

$$H(X,Y,Z) = X \oplus Y \oplus Z.$$

Функция F может быть описана как если X то Y, иначе Z.

Функция G – если 2 бита равны «1», то «1», иначе «0».

Функция H – сложение по модулю 2 или функция четности.

Определим также функции для каждого из трех раундов.

$$R1(a,b,c,d,k,s):$$

$$a = (a + F(b,c,d) + X[k]) <<< s$$

$$R2(a,b,c,d,k,s):$$

$$a = (a + G(b,c,d) + X[k] + 5A827999h) <<< s$$

$$R3(a,b,c,d,k,s):$$

$$a = (a + H(b,c,d) + X[k] + 6ED9EBA1h) <<< s$$

Здесь a <<< s означает побитовый циклический сдвиг на s символов. Все сложения здесь выполняются по модулю 2^{32}.

Сообщение разбивается на блоки по 512 бит. Каждый блок состоит из 16 слов по 32 бита.

```
 /* обрабатываем сообщение по блокам из 16 слов */
For i = 0 to N/16-1 do
/* копируем блок в массив X (M – массив сообщения) */
   For j = 0 to 15 do
X[j] = M[i*16+j].
end /* для цикла по j */
/* сохраняем значения A,B,C,D на начало цикла */
AA = A; BB = B; CC = C; DD = D;
```

/* Первый раунд. В каждом раунде используем определенную выше процедуру с различным порядком аргументов A, B, C, D и различными k и s */

```
R1(A,B,C,D,0,3);     R1(D,A,B,C,1,7);
R1(C,D,A,B,2,11);    R1(B,C,D,A,3,19);
R1(A,B,C,D,4,3);     R1(D,A,B,C,5,7);
R1(C,D,A,B,6,11);    R1(B,C,D,A,7,19);
R1(A,B,C,D,8,3);     R1(D,A,B,C,9,7);
R1(C,D,A,B,10,11);   R1(B,C,D,A,11,19);
R1(A,B,C,D,12,3);    R1(D,A,B,C,13,7);
R1(C,D,A,B,14,11);   R1(B,C,D,A,15,19);
```

/* Второй раунд */

```
R2(A,B,C,D,0,3);     R2(D,A,B,C,4,5);
R2(C,D,A,B,8,9);    R2(B,C,D,A,12,13);
R2(A,B,C,D,1,3);    R2(D,A,B,C,5,5);
R2(C,D,A,B,9,9);    R2(B,C,D,A,13,13);
R2(A,B,C,D,2,3);    R2(D,A,B,C,6,5);
R2(C,D,A,B,10,9);    R2(B,C,D,A,14,13);
R2(A,B,C,D,3,3);    R2(D,A,B,C,7,5);
R2(C,D,A,B,11,9);    R2(B,C,D,A,15,13);
```

/* Третий раунд */

```
R3(A,B,C,D,0,3);    R3(D,A,B,C,8,9);
R3(C,D,A,B,4,11);    R3(B,C,D,A,12,15);
R3(A,B,C,D,2,3);    R3(D,A,B,C,10,9);
R3(C,D,A,B,6,11);    R3(B,C,D,A,14,15);
R3(A,B,C,D,1,3);    R3(D,A,B,C,9,9);
R3(C,D,A,B,5,11);    R3(B,C,D,A,13,15);
R3(A,B,C,D,3,3);    R3(D,A,B,C,11,9);
R3(C,D,A,B,7,11);    R3(B,C,D,A,15,15);
```

/* Выполняем сложение регистов с сохраненными ранее состояниями */

```
A = A + AA;  B = B + BB;  C = C + CC;  D = D + DD;
end /* для цикла по i */
```

Шаг 5. Выходом хеш-функции является объединение регистров A, B, C, D. Регистры записываются в приведенном порядке от младшего бита к старшему.

Для алгоритма MD4 быстро появились успешные криптоатаки на отдельные раунды. На сегодняшний день Hans Dobbertin опубликовал успешные результаты криптоанализа (быстрого поиска коллизий) не только алгоритма MD4, но и MD5. Эти функции сегодня уже не могут применяться в серьезных криптографических приложениях. Из данного семейства функций, нет сведений об атаках делающих SHA-1 и выше и RIPEMD-160 и выше непригодными.

Большинство существующих атак позволяют сократить время поиска коллизий в 2^n раз. Если длина хеша составляет 128 бит, то даже незначительная уязвимость позволит находить коллизии на специальном оборудовании, а если уязвимость серьезная, то и на обычном ПК. При использовании длины в 256 бит и выше, уязвимость должна быть очень серьезной, что бы сократить количество проверок хотя бы до 2^{64}. В настоящее время известны уязвимости позволяющие сократить количество проверок при поиске коллизий до 2^6 для md4, 10 часов на ПК (порядок не приведен) для md5, 2^{16} для RIPEMD (не RIPEMD-160), 2^{39} для SHA-0 и 2^{63} для SHA-1. В условиях постоянного появления новых уязвимостей для функций данного семейства можно рекомендовать использование функций с максимальной (512) длиной выхода.

Хеш-функции и блочные шифры

Хэш-функции можно строить с помощью n-битового блочного шифра. Есть несколько способов такого конструирования. Все они используют постоянное открытое начальное значение [8]. Идея в том, что если безопасен блочный алгоритм, то и однонаправленная хэш-функция будет безопасной [11]. Однако в некоторых случаях хеш-функции более уязвимы, так как известен ключ шифрования.

Общую схему хеш-функции для алгоритмов, в которых длина блока равна длине хеш-значения, можно описать в виде

$H_0 = IV$, где IV (Initial Value) – значение инициализации определенное алгоритмом. В описанных далее алгоритмах также подразумевается выполнение этого действия.

$H_i = E_A(B) \oplus C$.

A, B, C – зависят от алгоритма и могут быть блоками сообщения (M_i), значением хеш на предыдущем шаге (H_{i-1}), комбинацией ($M_i + H_{i-1}$) или константой. Три переменных могут принимать одно из четырех значений, следовательно, существует 64 основные комбинации. Все они были изучены и выделены четыре безопасных варианта и восемь почти безопасных.

Безопасные алгоритмы:

$H_i = E_{H_{i-1}}(M_i) + M_i$,

$H_i = E_{H_{i-1}}(M_i + H_{i-1}) + M_i + H_{i-1}$,

$H_i = E_{H_{i-1}}(M_i) + M_i + H_{i-1}$,

$H_i = E_{H_{i-1}}(M_i + H_{i-1}) + M_i$.

Если ключ больше длины блока, то может применяться модифицированная схема:

$H_i = E_{H_{i-1},M_i}(H_{i-1})$.

Здесь ключом является объединение хеш-значения на предыдущем шаге и блока сообщения. Такая схема считается безопасной.

Наоборот, при построении схем с длиной хеш-значения равным удвоенной длине блока, функции оказываются не безопаснее чем с равными длинами.

Для обеспечения безопасности таких функций необходимо последовательное выполнение преобразований над частями хеш. Удачная реализация была получена в схемах Davies-Meyer: Tandem и Abreast (одновременная). В них так же использованы алгоритмы шифрования с длиной ключа в два раза больше длины блока.

Tandem:

$$W_i = E_{G_{i-1},M_i}(H_{i-1}),$$

$$G_i = E_{M_i,W_i}(G_{i-1}) \oplus G_{i-1},$$

$$H_i = W_i \oplus H_{i-1}.$$

Abreast:

$$G_i = G_{i-1} \oplus E_{M_i,H_{i-1}}(\overline{G}_{i-1}),$$

$$H_i = H_{i-1} \oplus E_{G_{i-1},M_i}(H_{i-1}).$$

Выходом функции является объединение блоков G и H.

Хеш-функция ГОСТ Р 34.11-94

Рассмотрим подробно хеш-функцию, определенную российским ГОСТ. В своей основе она использует алгоритм шифрования ГОСТ 28147-89. Но в отличие от рассмотренных выше алгоритмов используется множество усложнений [2, 16, 28].

В описании используются следующие условные обозначения:

B* – множество всех бинарных слов. Нумерация символов в слове начинается с 1 и осуществляется слева направо.

|A| – длина строки A.

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

A||B или AB – конкатенация строк A и B.

A^k – конкатенация k экземпляров слова A. Например, запись $(0^4 1^8)^2$ означает строку «000011111111000011111111».

M – блок сообщения

H – стартовый вектор хеширования или промежуточное состояние.

Основой алгоритма является функция сжатий внутренних итераций κ, отображающая две строки, по 256 бит, в одну строку в 256 бит.

$$\kappa : V_{256}(2), V_{256}(2) \to V_{256}(2).$$

Вычисление функции состоит из трех следующих шагов.

1. Генерирования четырех ключей по 256 бит.
2. Шифрующего преобразования.
3. Перемешивающего преобразования.

Генерирование ключей

Любую строку $X \in V_{256}(2)$ можно представить в виде $X = x_4 \| x_3 \| x_2 \| x_1 = \eta_{16} \| \eta_{15} \| ... \| \eta_1 = \xi_{32} \| \xi_{31} \| ... \| \xi_1$, где $x_i \in V_{64}(2), i = 1..4$; $\eta_i \in V_{16}(2), i = 1..16$; $\xi_i \in V_8(2), i = 1..32$.

Тогда можно определить преобразования:

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

$$P(X) = \xi_{\phi(32)} \| \xi_{\phi(31)} \| ... \| \xi_{\phi(1)},$$

где $\phi(i + 1 + 4(k - 1)) = 8i + k$, $i = 1..3, k = 1..8$. Например, $\varphi(32) = \varphi(3+1+4(8-1)) = 8*3+8 = 32$, $\varphi(31) = \varphi(2+1+4(8-1)) = 2*8+8 = 24$.

Для генерации ключей используются следующие исходные данные:

Стартовый вектор H и блок сообщения M. $H, M \in V_{256}(2)$.

Константы C_2, C_3, C_4.

$$C_2 = C_4 = 0^{256},$$

C_3 = ff00ffff000000ffff0000ff00ffff0000ff00ff00ff00ffff00ff00ff00ff00.

1. $i = 1$, $U = H$, $V = M$.
2. $W = U \oplus V$, $K_1 = P(W)$.
3. i++
4. если $i = 5$, то перейти к 7
5. $U = A(U) \oplus C_i$, $V = A(A(V))$, $W = U \oplus V$, $K_i = P(W)$.
6. перейти к шагу 3.
7. Завершение алгоритма.

Выходом алгоритма являются ключи $K_i, i = 1..4$

Шифрующее преобразование

Строка H представляется в виде четырех 64 битных строк $H = h_4 \| h_3 \| h_2 \| h_1$, шифруемых ключами $K_i, i = 1..4$. Для шифрования используется алгоритм ГОСТ 28147-89 в режиме простой замены (ECB).

$$s_i = E_{K_i}(h_i), i = 1..4.$$

$$S = s_4 \| s_3 \| s_2 \| s_1.$$

Перемешивающее преобразование

$$H = \eta_{16} \| \eta_{15} \| ... \| \eta_1.$$

Пусть $\psi(H) = \eta_{16} \oplus \eta_{15} \oplus ... \oplus \eta_1 \| \eta_{16} \| \eta_{15} \| ... \| \eta_2$, $\kappa(M,H) = \psi^{61}(H \oplus \psi(M \oplus \psi^{12}(S)))$, где ψ^i – i-я степень преобразования. Например, $\psi^3(X) = \psi(\psi(\psi(X)))$.

Порядок вычисления хеш-функции

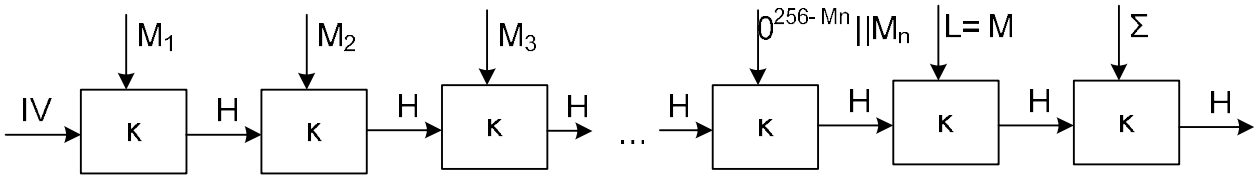


Рис. 38. Структура хеш-функции ГОСТ Р 34.11–94

Шаг 1. Инициализация.

$$M = M, H = IV, L = 0^{256}, \Sigma = 0^{256}.$$

Здесь IV – вектор инициализации, Σ – контрольная сумма, L – длина.

Шаг 2.

Если $|M| \le 256$ перейти к шагу 3.

M представить в виде $M_p \,||\, M_s$, $M_s \square V_{256}(2)$ и вычислить

$$H = \kappa(M_s, H),$$

$$L = L + 256 \, mod \, 2^{256},$$

$$\Sigma = \Sigma + M_s \, mod \, 2^{256},$$

$$M = M_p.$$

Повторить шаг 2.

Шаг 3.

Вычислить длину сообщения: $L = L + |M| \, mod \, 2^{256}$,

Добавить нули перед оставшейся частью сообщения до длины в 256 бит: $M = 0^{256-|M|} \,||\, M$

$$\Sigma = \Sigma + M \, mod \, 2^{256}.$$

$$H = \kappa(M, H), H = \kappa(L, H), H = \kappa(\Sigma, H).$$

Замечания по ГОСТ Р 34.11-94

Большим недостатком данного алгоритма, как и ГОСТ 28147-89 является его неполная документация. В алгоритме хеширования не определено значение вектора инициализации, а в используемом алгоритме шифрования не заданы таблицы замен. Вследствие этого каждая конкретная реализация отличается от другой. Для хеширования каждого блока используется 4 шифрования и многократное перемешивание, что делает функцию медленной. Медленнее других распространенных и медленнее шифрования. Преимуществом является 256 битный размер хеш-значения.

В 2008 г. появились сведения о возможности сокращения количества операций для поиска коллизий в алгоритме ГОСТ Р 34.11-94, в 2^{23} раз. Длина хеш-значения для данного алгоритма составляет 256 бит. Это оз-

начает, что для поиска коллизий требуется 2^{105} проверок, с сохранением результатов, вместо 2^{128}. Такие атаки, хоть и уменьшают доверие к алгоритму, но не сильно ослабляют его.

Контрольные вопросы

1. Чем отличается задача поиска коллизий от задачи поиска вторых прообразов?
2. Как можно рассчитать вероятность совпадений дней рождений в группе из n человек? (выведите формулу).
3. Почему при длине хеш-значения в n бит, задача поиска коллизий требует примерно $2^{\frac{n}{2}}$ вычислений? (приведите приближенные расчеты).
4. Оцените время и требования к памяти, для поиска коллизий на 64-битных и 128-битных хеш-функциях.
5. Почему длина сообщения также является параметром при хешировании?

ТЕМА 12. КОДЫ АУТЕНТИЧНОСТИ СООБЩЕНИЙ

Общие сведения

Коды аутентичности (или аутентификации) сообщений (MAC – Message Authentication Code) представляют собой дополнение к сообщению, позволяющее выявить попытку фальсификации сообщения. В российской терминологии, в частности в ГОСТ 28147-89, для обозначения кодов аутентичности используется термин **имитовставка**.

Применение только шифрования не позволяет решить проблему целостности сообщений полностью. Если злоумышленник знает несколько пар открытый текст – шифротекст, он может заменять в канале связи передаваемый шифротекст своим и надеяться на изменение смысла сообщения или появление сбоев.

Коды аутентичности сообщений вычисляются на основе сообщения и секретного ключа, известного отправителю и получателю и добавляются к сообщению. Наличие такого кода, для третьего лица знающего ключ, может свидетельствовать о том, что сообщение исходит либо от отправителя, либо от получателя. Например, Боб не может предъявить ключ и MAC арбитру с тем, чтобы доказать, что сообщение он получил от Алисы, так как это сообщение мог составить и он сам. Но никакое третье лицо, которому неизвестен ключ, не может незаметно изменить сообщение.

Код аутентичности сообщения – это функция, которая принимает на вход два аргумента: ключ K и сообщение m. Будем обозначать $MAC(K, m)$.

Существует два основных подхода к вычислению MAC. В первом используются блочные шифры, обычно в режиме CBC (CBC-MAC), во втором – хеш-функции (HMAC).

MAC и блочные шифры (CBC-MAC)

При вычислении CBC-MAC сообщение m шифруется ключом K в режиме CBC. Последний блок шифротекста и является кодом аутентичности [8, 9].

Сообщение дополняется так, чтобы его можно было разбить на блоки необходимой длины и представляется в виде $P_1 \dots P_k$. Значение MAC вычисляется следующим образом:

$$H_0 = IV,$$
$$H_i = E_K(P_i \oplus H_{i-1}),$$
$$MAC = H_k.$$

Значение вектора инициализации IV, здесь может быть фиксированным.

Существует ряд атак, специфичных для CBC-MAC на основе коллизий, которые сокращают безопасность до половины длины блока.

CBC-MAC не может быть использован совместно с шифрованием в режиме CBC (по крайней мере, с тем же ключом и вектором инициализации), так как значение кода аутентификации будет просто равно последнему блоку шифротекста.

Для режима CBC-MAC также характерно:

Если $MAC(a) = MAC(b)$, где длины a и b кратны длине блока, то $MAC(a||c) = MAC(b||c)$.

Злоумышленник может накапливать сообщения и их подписи пока не найдет коллизию $MAC(a) = MAC(b)$, после чего если он сможет заставить вычислить MAC сообщения $a||c$ одну из сторон, то он получит и MAC для сообщения $b||c$.

Для того чтобы избежать такой атаки в описанном варианте, через функцию MAC необходимо пропускать строку вида $a||l$, где l – длина сообщения.

Но и в таком виде функцию CBC-MAC нельзя считать безопасной. Более безопасными считаются MAC на основе хеш-функций.

Режим имитовставок по ГОСТ

В ГОСТ 28147-89 режим вычисления кодов аутентификации (или имитовставок) оговорен отдельно, но в целом совпадает с режимом CBC-MAC.

1. Первый 64-битный блок информации, для которой вычисляется имитоприставка, записывается в регистры $A||B$ и шифруется в сокра-

щенном режиме, в котором выполняется 16 раундов основного преобразования вместо 32-х.

2. Полученный результат суммируется по модулю 2 со следующим блоком открытого текста и сохраняется в $A||B$. Операция повторяется со всеми блоками до конца сообщения.

3. В качестве имитоприставки используется результирующее содержимое регистров $A||B$ или его часть (в зависимости от требуемого уровня стойкости).

$$H_1 = E_K(P_1),$$
$$H_i = E_K(P_i \oplus H_{i-1}),$$
$$MAC = H_k.$$

Использование 16 циклов преобразования вместо 32 позволяет избежать совпадения имитовставки и последнего блока шифротекста в режиме CBC (впрочем, он и не применяется в ГОСТ). Еще одним отличием от CBC-MAC является отсутствие вектора инициализации. Шифрование начинается сразу с первого блока.

MAC и хеш-функции, HMAC

Простейшим способом вычисления MAC на основе хеш-функции является вычисление хеш от конкатенации сообщения и ключа [8,9].

$$MAC = h(m||K).$$

Однако такой способ не считают особенно надежным. Сложность его взлома не выше нахождения K (или коллизии для K) по $h`(K)$. $h`$ – это та же функция h, только внутреннее состояние соответствует моменту, когда через внутреннюю функцию пропущено сообщение m. Использование функций вида $h(K||m)$ или даже $h(K||m||K)$ также считается ненадежным

Вместо них рекомендуют использовать более сложный вариант, называемый HMAC (hash MAC).

HMAC разрабатывалась так, чтобы избежать проведения атак с восстановлением ключа в автономном режиме. Для проведения успешной атаки злоумышленник должен выполнить $2^{\frac{n}{2}}$ взаимодействий с атакуемой системой, что значительно сложнее, чем тоже количество вычислений на собственном компьютере.

В [5] определен следующий формат MAC:

$$HMAC = h((K \oplus opad) || h((K \oplus ipad) || m)).$$

Здесь h – хеш-функция с выходом длиной L байт и длиной блока B байт (например, у md5 длина блока 512 бит или 64 байта, а длина выхода 128 бит или 16 байт); opad и ipad – константы длиной B байт. opad со-

стоит из байтов 5Ch, ipad – из байтов 36h; для выполнения сложения, K дополняется справа нулями. Если длина K больше B, то $K = h(K)$.

Если длина ключа меньше чем L байт, то это снижает безопасность функции. Но если длина ключа больше L байт, то это увеличивает безопасность незначительно. Однако если для получения ключа используется слабый генератор псевдослучайных чисел, то лучше использовать ключи длиннее L.

Возможно урезание выхода HMAC. Это уменьшает количество информации доступной криптоаналитику, но в то же время и уменьшает длину строки, которую нужно предсказать криптоаналитику. В RFC 2104 рекомендуется использовать не менее $L/2$ и не менее 80 старших битов.

Название алгоритма из семейства HMAC записывается в виде HMAC-H-t, где H – название алгоритма хеширования, t – размер выхода, если он был урезан. Например, HMAC-md5-80 означает, что используется алгоритм md5 и выходом являются первые 80 бит из 128.

UMAC

Также существует относительно новый алгоритм аутентификации сообщений – UMAC (Message Authentication Code using Universal Hashing) [6]. Этот алгоритм использует собственную хеш-функцию, скромно называемую универсальной (UHASH).

В общем виде алгоритм можно представить:

$$MAC = H_{K1}(m) \square F_{K2}(nonce),$$

где H – функция хеширования UHASH; F – псевдослучайная функция;

Nonce – метка времени. Поле nonce должно содержать уникальный номер сообщения. Этот номер может вырабатываться одинаковыми генераторами или передаваться с сообщением в открытом виде до тех пор, пока используются одинаковые ключи K1 и K2 этот номер не должен повторяться.

Длина выхода этой функции может быть 32, 64, 96 или 128 бит. Авторы считают, что даже коротких длин кодов аутентификации должно хватить, так как невозможно проверить все возможные варианты MAC, не взаимодействуя с получателем. Отправка же даже 2^{32} сообщений с различными подписями явно вызовет подозрение у получателя. Задача нахождения K1 и K2 по MAC, m и nonce не становится проще при сокращении длины выхода. Однако в [9] высказывается мнение, что надежность алгоритмов не должна зависеть от архитектуры. Алгоритм UMAC сложно взломать при коротких длинах выходов, только если криптоаналитик ограничен в возможностях проверки кодов.

Общие замечания по использованию MAC

В описании UMAC явно оговорено наличие поля nonce. В других стандартах наличие метки времени не всегда оговаривается явно. Но очевидно, что если такая информация отсутствует, то злоумышленник может просто отправить перехваченный ранее блок информации. Такой вид атаки может быть очень опасен. Например, в транзакциях между банкоматом и банком может быть запрошено повторное зачисление денег на счет. Или наоборот, если не будет доставлено несколько сообщений, то банк не получит сведений о снятии денег.

Очевидно, что для защиты от второго вида атаки, метки времени должны быть не только уникальными номерами, но и образовывать некоторую последовательность. Учитывая то, что номер сообщения информация открытая, то это может быть и последовательность натуральных чисел. Если после сообщения № 15 получено сообщение № 20, то должна запрашиваться повторная передача пропущенных сообщений, до тех пор, пока они не будут переданы корректно.

Номера сообщений должны быть уникальными до тех пор, пока не будут изменены ключи. При этом, чем чаще меняются ключи, тем меньше информации для взлома может собрать криптоаналитик.

В общем случае MAC вычисляется от конкатенации сообщения, метки времени и возможно других служебных полей.

$MAC = MAC_K(m||t)$.

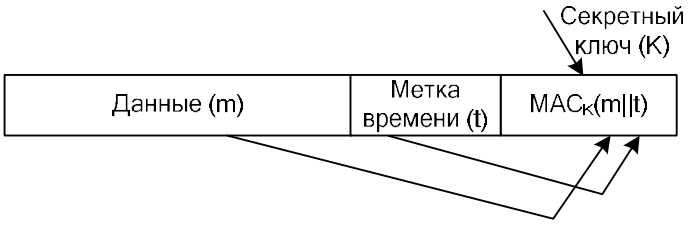


Рис. 39. Структура пакета при использовании кодов аутентичности сообщений

В большинстве хеш-функций длина сообщения участвует в вычислениях, поэтому ее не обязательно добавлять при вычислении MAC, однако если используется MAC на основе блочных шифров, то к сообщению должно быть добавлено поле «длина».

Контрольные вопросы

1. Могут ли коды аутентичности сообщений защитить от отказа от авторства?
2. Каким основным уязвимостям подвержены коды аутентичности сообщений?
3. Для чего нужны метки времени?

ТЕМА 13. КРИПТОСИСТЕМЫ С ОТКРЫТЫМ КЛЮЧОМ

Общие сведения

Если отправитель и получатель должны знать один и тот же ключ, то он должен быть каким либо образом передан. В некоторых случаях может существовать абсолютно защищенный канал связи, для передачи ключей, или канал связи, защищенный ранее согласованными ключами, или ключи могут быть переданы лично в условиях повышенной секретности. Эти условия не всегда могут быть выполнены. Именно из-за того, что распределение ключей не могло быть выполнено по защищенным каналам, была вскрыта немецкая Enigma.

Для решения проблем связанных с распределением ключей, были созданы ассиметричные криптосистемы или криптосистемы с открытым ключом. Впервые концепция таких систем была предложена Уитфилдом Диффи и Мартином Хеллманом в 1976 г. Вскоре появились и первые практические реализации [11].

В противоположность ассиметричным системам, все системы, использующие один ключ, стали называться симметричными, или криптосистемами с закрытым ключом.

В ассиметричных криптосистемах, для шифрования и расшифрования используются различные ключи.

Ключ, используемый для шифрования сообщений, называется **открытым** (Public Key). Этот ключ может быть передан по незащищенным каналам связи или быть опубликован в общедоступных базах. Далее считаем, что к нему может иметь доступ любой желающий.

Ключ, используемый для расшифрования, называется **секретным** или **закрытым** (Private Key). Получатель сообщения должен сохранить его в тайне. Открытый и закрытый ключи формируются получателем единовременно.

Шифрование и расшифрование записывается в виде:

$$C = E_{OK}(P),$$

$$P = D_{PK}(C),$$

где OK – открытый ключ; PK – закрытый ключ.

При этом шифрование с помощью открытого ключа осуществляется легко, а расшифрование с помощью открытого ключа и нахождение закрытого ключа, по открытому – сложные проблемы. Большинство алгоритмов ассиметричного шифрования имеют очевидную уязвимость – вскрытие подбором открытого текста. Перехватив шифротекст, злоумышленник может предположить точное содержимое сообщения, зашифровать его и проверить, действительно ли передавалось именно это сообщение. В симметричных криптосистемах также существуют атаки с известным открытым текстом, однако точно проверить, что был за-

шифрован предполагаемый текст в большинстве систем с закрытым ключом нельзя.

В некоторых протоколах назначение открытого и закрытого ключей меняются. Например, в алгоритмах цифровой подписи, сообщение подписывается закрытым ключом, после чего любой желающий может проверить подлинность с помощью открытого.

Легковычислимые и трудновычислимые задачи

Идея ассиметричного шифрования с точки зрения математики состоит в том, что существуют такие функции f, что их вычисление осуществляется быстро и легко, но имея уравнение $C = f(P)$, трудно найти обратную функцию f^{-1}, такую, что $P = f^{-1}(C)$. Это условие обеспечивает невозможность расшифрования сообщений злоумышленником. Для того, что бы сообщение могло быть расшифровано законным получателем, функция f^{-1} должна существовать, получатель должен знать эту функцию, и она должна быть легковычислимой.

Функции f и f^{-1} должны быть легковычислимы, то есть иметь полиномиальную сложность. Задача нахождения f^{-1} по f должна быть трудновычислимой, задача нахождения f по f^{-1} может быть легковычислимой, но тогда назначение функций уже не может быть изменено и такие функции не могут использоваться в ЭЦП.

При рассмотрении ассиметричных криптосистем, необходимо исходить из того, что криптоаналитик обладает открытым ключом, может сформировать любые открытые тексты и вычислить для них шифротексты.

На сегодняшний день существует несколько типов задач такого типа, которые могут быть успешно применены в криптографии. В первую очередь это задачи разложения на множители больших чисел, задачи дискретного логарифмирования и выполнения операций на эллиптических кривых.

Задача об укладке рюкзака

Проблема рюкзака может быть сформулирована следующим образом: легко вычислить массу рюкзака, зная, что в нем лежит, но сложно узнать, что в нем лежит, зная его массу.

При этом можно выделить две задачи вычисления набора предметов, легкую и сложную. Если имеется последовательность чисел, обозначающих массы, в которой каждое последующее число больше суммы

всех предыдущих, то такая последовательность называется сверхвозрастающей и соответствует легкой задаче.

Действительно, можно взять вес рюкзака и сравнить его с самым большим числом последовательности. Если вес меньше, то это число не входит в рюкзак, если больше, то входит, и мы уменьшаем вес рюкзака на это число. Далее повторяем проверки для следующих чисел.

Если имеется последовательность чисел {2, 3, 6, 13, 30, 67, 121} и вес рюкзака равен 88, то однозначно определяется набор чисел 67, 13, 6, 2.

Если последовательность чисел не является сверхвозрастающей, то очевидным решением является полный перебор возможных масс. Например, если дана последовательность {11, 82, 33, 6, 34, 41, 76} и вес рюкзака равен 91, то неочевидно какие числа входят в рюкзак, хотя для короткой последовательности вычислить это не сложно.

В начале формируется закрытый ключ: сверхвозрастающая последовательность $\{M_1,\ldots,M_n\}$ и числа k и n: k взаимнопростое с n и $n > \sum M_i$. Открытым ключом является обычная не сверхвозрастающая последовательность $\{m_1,\ldots,m_n\}$. Она может быть получена из закрытого ключа: $m_i = M_i \cdot k \mod n$.

Формально процесс шифрования можно представить в следующем виде: имеется битовая строка $b_1 b_2 \ldots b_n$ и набор весов $\{m_1,\ldots,m_n\}$. Сумма $s = \sum_{i=1}^{n} b_i m_i$, где $b_i \in \{0,1\}$ является шифротекстом для строки $b_1 b_2 \ldots b_n$.

Законный пользователь знает сверхвозрастающую последовательность числа k и n. Для получения отдельных весов была использована операция умножения по модулю. Для получения веса сверхвозрастающего рюкзака при расшифровании необходимо применить обратную операцию к сумме.

$$S = s \cdot k^{-1} \mod n.$$

k^{-1} может быть получено из k с помощью алгоритма Эвклида.

По сумме S и сверхвозрастающей последовательности $\{M_1,\ldots,M_n\}$ можно расшифровать строку и найти веса входящие в набор и соответственно номера битов равных «1».

Рассмотрим процесс формирования ключей, шифрования и расшифрования на примере.

Выберем секретный ключ:

{2, 3, 9, 17, 29, 65, 144, 291, 631, 1341, 2723, 5737, 11325, 23421, 48917, 99651}, n = 200536, k = 45863. С помощью алгоритма Эвклида вычислим $k^{-1} = 46143$.

Вычислим открытый ключ

$$m_i = M_i \cdot k \bmod n = M_i \cdot 45863 \bmod 200536.$$

Таблица 11

Вычисление открытого ключа

№ бита	0	1	2	3	4	5	6	7
РК (M_i)	2	3	9	17	29	65	144	291
ОК (m_i)	91726	137589	11695	178063	126811	173591	187120	110757
№ бита	8	9	10	11	12	13	14	15
РК (M_i)	631	1341	2723	5737	11325	23421	48917	99651
ОК (m_i)	62369	138267	151557	12799	10235	86507	84139	78373

Пусть требуется зашифровать следующую бинарную строку:

Открытый текст: "11001010 01011010 10001110 10101001".

Так как определен ключ из 16 чисел, то получаем два блока по 16 бит.

$s_1 = m_0 + m_1 + m_4 + m_6 + m_9 + m_{11} + m_{12} + m_{14}$ *=91726+137589+126811+187120+138267+ 12799+10235+84139 =788686*

$s_2 = m_0 + m_4 + m_5 + m_6 + m_8 + m_{10} + m_{12} + m_{15}$ *=91726+126811+173591+187120+62369+151557+10235+78373 = 881782*

Шифротекст: "788686 881782".

Для расшифрования необходимо домножить эти числа на k^{-1}.

$S_1 = 788686 \cdot 46143 \ mod\ 200536 = 67498,$

$S_2 = 881782 \cdot 46143 \ mod\ 200536 = 114570.$

Мы получили суммы сверхвозрастающих последовательностей. По ним теперь можно определить какие биты равны «1». Для этого в таблице сравниваем значение суммы с значением M_i. Если $\Sigma > M_i$ то в b_i записываем 1, а в следующую ячейку суммы $\Sigma - M_i$, иначе b_i – 0, а сумма не изменяется. Вычисления производятся справа налево.

Таблица 12

Расшифрование первого блока S_1

№ бита	8	9	10	11	12	13	14	15
Σ	178	1519	1519	7256	18581	18581	67498	67498
РК (M_i)	631	1341	2723	5737	11325	23421	48917	99651
Бит b_i	0	1	0	1	1	0	1	0
№ бита	0	1	2	3	4	5	6	7
Σ	2	5	5	5	34	34	178	178
РК (M_i)	2	3	9	17	29	65	144	291
Бит b_i	1	1	0	0	1	0	1	0

Таблица 13

Расшифрование второго блока S_2

№ бита	8	9	10	11	12	13	14	15
Σ	871	871	3594	3594	14919	14919	14919	114570
РК (M_i)	631	1341	2723	5737	11325	23421	48917	99651
Бит b_i	1	0	1	0	1	0	0	1
№ бита	0	1	2	3	4	5	6	7
Σ	2	2	2	2	31	96	240	240
РК (M_i)	2	3	9	17	29	65	144	291
Бит b_i	1	0	0	0	1	1	1	0

Расшифрованный открытый текст совпадает с исходным.

В реальных условиях использовались ключи, состоящие из 200 и более чисел. Сложность вскрытия рюкзака «в лоб» соответствует сложности перебора открытых текстов. Но задача была решена значительно быстрее полного перебора. Криптосистемы на основе задачи об укладке рюкзака вскрыты почти во всех их вариациях [11]. В [15] даны начальные установки для вскрытия.

RSA

О криптосистеме RSA сейчас, наверное, пишут больше всего. В мире криптографии с открытым ключом эта система стала стандартом де-факто. Ее стойкость основана на проблеме разложения на множители больших чисел (факторизации). Каждый раз, говоря о новых методах ускорения факторизации, упоминают и о связанных с ними ослаблениях криптосистемы. Вычислительная сложность факторизации не доказана, то есть не доказано, отсутствие эффективных методов разложения больших чисел на множители. Соответственно и не доказано их существование. Существующие методы факторизации не позволяют в приемлемые сроки разложить на множители числа длиной 1024 или 2048 бит. Также можно найти утверждения, что задача RSA проще задачи факторизации, то есть не обязательно раскладывать число на множители, что бы расшифровывать сообщения. Не смотря на эти опасения, RSA остается самой популярной криптосистемой с открытым ключом. Здесь рассмотрим задачу с традиционным объяснением ее стойкости [25].

Для генерации ключей выбираются два простых числа p и q и вычисляется их произведение $n = pq$. Вычисляется $\varphi(n) = (p-1)(q-1)$. Для того чтобы вычислить $\varphi(n)$, как раз необходимо знать разложение n на множители. Выбираем один из ключей – число e взаимнопростое с $\varphi(n)$. Вычисляем второй ключ из соотношения $ed = 1 \,(mod\ \varphi(n))$ или

$d = e^{-1} \ (mod \ \varphi(n))$. d в данном случае может быть вычислено с помощью расширенного алгоритма Эвклида. Ключи e и d могут меняться местами. То есть сообщение может быть зашифровано с помощью e и расшифровано с помощью d и наоборот. При этом одинаково сложно как по e и n вычислить d, так и по d и n вычислить e. Здесь будем считать, что e и n – открытый ключ, d – закрытый.

Для шифрования сообщение разбивается на блоки m_i, численное представление которых меньше n. Шифрование осуществляется по формуле

$$c_i = m_i^e \ mod \ n.$$

Расшифрование осуществляется по формуле

$$m_i = c_i^d \ mod \ n.$$

После выполнения шифрования и расшифрования, получаем исходный текст, так как:

$$ed = 1 \ (mod \ \varphi(n)) = k \cdot \varphi(n) + 1$$

$$c_i^d \ mod \ n = (m_i^e)^d \ mod \ n = m_i^{ed} \ mod \ n = m_i^{k \cdot \varphi(n)+1} \ mod \ n.$$

Обобщенная малая теорема Ферма утверждает, что

$$m^{\varphi(n)} = 1 (mod \ n).$$

Следовательно,

$$m_i^{k \cdot \varphi(n)+1} \ mod \ n = m_i^1 \cdot (m_i^{\varphi(n)})^k \ mod \ n = m_i.$$

Если при генерации ключей p или q окажется составным, то алгоритм работать не будет, так как $\varphi(n)$ будет вычислено неверно, и не будет выполняться $m^{\varphi(n)} = 1 (mod \ n)$.

Рассмотрим пример шифрования и расшифрования RSA на малых числах (чтобы можно было проверить его работу на калькуляторе).

Выберем два простых числа p и q:

`p = 31, q = 23.`

Вычислим n и φ(n):

`n = 713, φ(n) = 660.`

Выберем открытый ключ e; e взаимнопростое с $\varphi(n)$ меньшее n:

`e = 457.`

Вычислим закрытый ключ d; $d = e^{-1} \ (mod \ \varphi(n))$:

`d = 13.`

Пусть блок открытого текста представлен числом m < n:

`m = 215.`

Тогда шифротекст $c = m^e \ mod \ n = 215^{457} \ mod \ 713$:

`c = 151.`

Расшифровываем $m = c^d \ mod \ n = 151^{13} \ mod \ 713$:

`m = 215.`

Открытый текст до шифрования и после расшифрования не изменился.

Самым очевидным методом вскрытия является разложение n на множители, однако не доказано, что это необходимо, чтобы вычислить m по c и e. Но если способ вскрытия позволит найти закрытый ключ d, то это позволит решить и задачу разложения на множители.

У RSA имеются уязвимости, ограничивающие способы его применения в протоколах. В частности RSA нельзя использовать напрямую (без предварительного хеширования) в алгоритмах ЭЦП.

При использовании RSA только для шифрования опасности связаны с использованием в криптографической системе одинаковых модулей [8, 11]. Один законный пользователь в этом случае может вычислить по своим ключам значение $\varphi(n)$ и после этого может вычислить секретные ключи всех остальных пользователей. Внешний криптоаналитик может прочитать сообщение, если оно зашифровано различными ключами и отправлено двум разным получателям.

В этом случае криптоаналитик может получить два шифротекста и открытые ключи пользователей:

$$c_1 = m^{e_1} \bmod n,$$

$$c_2 = m^{e_2} \bmod n.$$

Криптоаналитик может получить доступ к $c_1, c_2, e_1, e_2 \text{ и } n$. Если e_1 и e_2 являются взаимно простыми, с помощью расширенного алгоритма Эвклида он может найти r и s из соотношения

$$re_1 + se_2 = 1,$$

и вычислить

$$c_1^r \square c_2^s = m^{re_1} m^{se_2} = m^{re_1+se_2} = m.$$

r или s является отрицательным, но c_1^r может быть представлено в виде $(c_1^{-1})^{-r}$.

При использовании схемы RSA необходимо следовать следующим рекомендациям [8].

• В протоколах сетей связи, применяющих RSA, не должен использоваться общий модуль.

• Знание одной пары показателей шифрования/дешифрирования для данного модуля позволяет взломщику разложить модуль на множители или вычислить другие пары показателей, не раскладывая модуль на множители.

• Показатели шифрования и дешифрования должны быть большими

• Для предотвращения вскрытия малого показателя шифрования сообщения должны быть дополнены случайными значениями. Дополнение блоков случайными значениями также позволяет повысить стойкость против большого количества атак.

При выборе длины ключа необходимо исходить из сложности имеющихся алгоритмов факторизации. Так, например Алгоритм Ленстры требует $e^{((2+O(1))\log p \log\log p)^{1/2}} \log^2 n$ операций, где n – составное число, которое необходимо разложить, а p – минимальный делитель [18]. Для n в 1024 бит и p в 512 бит – это $\approx e^{64} \approx 2^{20} \approx 2^{112}$ операций, что на сегодняшний день уже нельзя считать достаточным. Но длины n в 2048 бит и p в 1024 можно еще некоторое время считать вполне надежными. Факторизация потребует $\approx 2^{160}$ операций. В [9] рекомендуется в разрабатываемом программном обеспечении сразу предусматривать поддержку значений n до 8192 бит. Такие значения могут потребоваться в случае появления новых математических методов решения задачи факторизации или задачи RSA.

Уязвимыми могут быть также конкретные реализации генерации простых чисел p и q. Существует несколько алгоритмов получения простых чисел. Для их инициализации используются случайные числа. Соответственно для того, что бы подобрать p и q можно попытаться подобрать параметры, использовавшиеся для их получения.

El Gamal

Безопасность схемы El Gamal (Эль Гамаль) основана на сложности дискретного логарифмирования [11].

Выбирается простое число p и два случайных числа g и x меньше p. Вычисляется

$$y = g^x \bmod p .$$

Открытым ключом является y, g и p. Закрытым – x.

Для шифрования сообщения M сначала выбирается случайное число k, взаимно простое с p – 1. Затем вычисляются

$$a = g^k \bmod p ,$$

$$b = y^k M \bmod p .$$

Иногда также встречается вариант $b = y^k \cdot M \bmod p$ [14, 16].

Шифротекст состоит из двух частей, а служит для неявной передачи значения k, а b для передачи сообщения. При шифровании каждый раз должно использоваться новое значение k, так как если криптоаналитик узнает k, то он сможет расшифровывать все сообщения с тем же k.

Для расшифрования вычисляется $M = b / a^x \bmod p$.

Если использовалось сложение, то $M = b - a^x \bmod p$.

Действительно,

$$b / a^x \bmod p = y^k M / g^{kx} \bmod p = g^{xk} M / g^{kx} \bmod p = M .$$

При расшифровании, естественно, необходимо вначале найти $a^{-1} \bmod p$ и вычислить $M = b(a^{-1})^x \bmod p$.

Для того чтобы вскрыть шифр, криптоаналитику нужно, зная $y = g^x \bmod p$ и $a = g^k \bmod p$, найти $g^{kx} \bmod p$. Эту задачу называют проблемой Диффи-Хеллмана. Ее решение связано с проблемой дискретного логарифмирования, то есть нахождения a по g и $g^a \bmod p$, однако не доказано, что это единственный способ решения и возможно проблема Диффи-Хеллмана решается проще [20].

Криптосистемы с эллиптическими кривыми

Криптографические алгоритмы на основе эллиптических кривых сложнее для понимания и требуют знания теории полей. Но они считаются и наиболее надежными из ассиметричных алгоритмов и требуют значительно меньшей длины ключа. Эллиптические кривые применяются в криптографии с 1985 г., причем как для факторизации чисел и проверки простоты, так и для построения криптографических протоколов.

Здесь описание эллиптических кривых приведем лишь поверхностно, не вдаваясь в теорию групп и полей.

В общем случае уравнение эллиптической кривой E имеет вид:

$$y^2 + axy + by = x^3 + cx^2 + dx + e.$$

В криптографии применяется уравнение вида

$$y^2 = x^3 + ax + b \ \ mod\ p,$$

где a и b должны удовлетворять

$$4a^3 + 27b^2 \neq 0 \ \ mod\ p.$$

Инвариантом эллиптической кривой называется величина $J(E)$, удовлетворяющая тождеству [3]

$$J(E) = 1728\frac{4a^3}{4a^3 + 27b^2}.$$

Коэффициенты a, b эллиптической кривой E, по известному инварианту $J(E)$, определяются следующим образом

$$\begin{cases} a = 3k(mod\ p) \\ b = 2k(mod\ p) \end{cases},$$

где k определяется через инвариант

$$k = \frac{J(E)}{1728 - J(E)}\ (mod\ p), J(E) \neq 0, J(E) \neq 1728.$$

На эллиптической кривой определена операция сложения. Нулем является точка O, также называемая «бесконечно удаленная точка». В этой точке сходятся все вертикальные прямые. Сумма трех точек ле-

жащих на одной прямой равна O. Если P, Q и $R(x, y)$ лежат на одной прямой, то $P + Q = (x, -y)$.

Правила сложения можно записать в виде:

$$(x, y) + O = (x, y),\ O + O = O\ .$$
$$(x, y) + (x, -y) = O.$$

Пусть $P = (x_1, y_1)$, $Q = (x_2, y_2)$ и $x_1 \neq x_2$. Проведем прямую через точки P и Q. Ее уравнение будет выглядеть

$$y = y_1 + \lambda(x - x_1)\ (mod\ p),$$

где $\lambda = \dfrac{y_2 - y_1}{x_1 - x_2}\ (mod\ p)$.

Найдем точки пересечения прямой и кривой. Это будут три точки P, Q и $R(x_3, y_3)$.

$$(y = y_1 + \lambda(x - x_1))^2 = x^3 + ax + b\ (mod\ p)$$

Для этого уравнения по теореме Виета выполняется равенство

$$x_1 + x_2 + x_3 = \lambda^2\ (mod\ p).$$

Координатами точки R являются:

$$x_3 = \lambda^2 - x_1 - x_2\ (mod\ p),\ y_3 = y_1 + \lambda(x_3 - x_1)\ (mod\ p).$$
$$P + Q = (x_3, -y_3).$$

В случае, если необходимо удвоить точку $P(x_1,y_1)$, найти $P+P$, то в этой точке проводится касательная к кривой. Она имеет вид:

$$y = y_1 + \lambda(x - x_1)(mod\ p),$$

где $\lambda = \dfrac{3x_1^2 + a}{2y_1}\ (mod\ p)$.

Координаты точки R, лежащей на той же прямой:

$$x_3 = \lambda^2 - 2x_1\ (mod\ p),\ y_3 = y_1 + \lambda(x_3 - x_1)\ (mod\ p).$$
$$2P = P + P = (x_3, -y_3)\ .$$

Умножение точки на скаляр реализуется последовательностью сложений, аналогично возведению в степень по модулю. Поэтому обратную задачу называют дискретным логарифмированием на эллиптических кривых.

При шифровании выбирается кривая $E_{a,b}$, задаваемая коэффициентами a и b и точка G на ней. Выбирается число n, и вычисляется $P = nG$ [27].

E, G и P являются открытым ключом, n – закрытым.

Отправитель представляет блок открытого текста в виде точки M, выбирает случайное число k и вычисляет шифротекст C, состоящий из двух точек.

$$C = \{k \square G; M + k \square P\}.$$

Для дешифрования получатель вычисляет:

$$M = M + k \square P - (k \square G) \square n = M + k \square n \square G - k \square n \square G.$$

Криптоаналитику для взлома необходимо вычислить k, зная G и kG. Эта операция и называется дискретным логарифмированием на эллиптической кривой.

Рассмотренная схема аналогична схеме El Gamal. Существуют и другие схемы шифрования на основе эллиптических кривых. В некоторых схемах могут использоваться другие виды уравнения кривой, но обычно кривая рассматривается именно в таком виде.

Вероятностное шифрование

Одним из недостатков ассиметричных систем шифрования является то, что злоумышленник в большинстве случаев может получить доступ к открытым ключам, и когда к нему попадает шифротекст то он может пытаться зашифровать различные варианты открытого текста, пока не получит такой же шифротекст. Если длина блока шифрования 1024 символа и шифруемый текст является в некоторой мере случайным, такая атака не представляет большой опасности. Но если в системе передается одно из десяти возможных сообщений и злоумышленнику нужно узнать, которое именно, он без труда это сделает.

Вероятностное шифрование позволяет преобразовать один и тот же текст P, в один из множества возможных шифротекстов C. Открытому тексту P ставится в соответствие множество CP, C□CP.

Одним из подходов к вероятностному шифрованию является постановка в соответствии открытому тексту P множества открытых текстов PP. Например, нулю может быть поставлено в соответствие множество четных чисел, а единице – множество нечетных; к блоку открытого текста могут быть добавлены произвольные символы в начале или в конце сообщения; открытый текст может быть умножен на некоторое число по модулю, а само число добавлено к открытому тексту. Сообщение может быть даже зашифровано, а ключ добавлен к полученному тексту, хотя такой способ уже не повысит надежности. Наименее затратным способом является добавление к строке открытого текста, неповторяемой случайной строки, фиксированной длины.

Пусть длина блока при шифровании – n бит.

1. Разобьем сообщение P на блоки длиной $n–r$ бит $P_1,\ldots,P_k$.

2. К каждому P_i добавим случайную неповторяющуюся строку длиной r бит. Шифруем каждый блок.

3. После расшифрования отбрасываем последние r бит.

Если n – достаточно велико (> 512 бит), то r целесообразно брать 64–96 бит, чтобы гарантировать невозможность перебора случайных строк. Даже если злоумышленник будет знать, что была отправлена одна из двух возможных строк, он не сможет узнать, которая. Перебрать 96 бит не намного проще, чем разложить число на множители. Требова-

ние неповторимости не относится к вероятностному шифрованию, но его выполнение позволит обеспечить защиту от повторной передачи.

Схема El Gamal сама по себе уже является вероятностной, так как в ней для шифрования каждого блока выбирается случайное число k и зная открытый текст, проверить, что был зашифрован именно он – невозможно.

С вероятностным шифрованием связано также появление подсознательного канала (см. тему 15. Электронно-Цифровые подписи).

Дополнительные вопросы

Здесь не рассмотрены вопросы генерации простых чисел, проверки на простоту, алгоритмы факторизации и дискретного логарифмирования; способы выбора параметров эллиптических кривых. Эти вопросы можно изучить по [8, 18, 19, 20].

Контрольные вопросы

1. К чему приведет выбор составных p и q в RSA? Как должны формироваться ключи, если n раскладывается на 3 множителя? Почему такие схемы не используются?
2. Может ли шифрующая экспонента в системе RSA быть четной?
3. Чем отличается решение задачи RSA от разложения на множители?
4. Какую проблему нужно решить криптоаналитику, для вскрытия схемы El Gamal?
5. Почему для шифрования в схеме EL Gamal каждый раз должно использоваться новое значение k?
6. Сопоставьте схему El Gamal и приведенную схему на эллиптических кривых.

ТЕМА 14. ПРОТОКОЛЫ ОБМЕНА КЛЮЧАМИ

Общие сведения

Развитие ассиметричной криптографии не привело к отказу от симметричной. Невозможность решения проблем, на которых построена ассиметричная криптография не доказана а следовательно ей нельзя доверять в полной мере. Алгоритмы, применяемые в симметричной криптографии, являются значительно более надежными, хотя бы потому, что не всегда возможно определить, правильно ли подобран ключ. Но проблемой является передача ключа.

Рациональным решением, объединяющим достоинства ассиметричной и симметричной криптографии является передача ключа с по-

мощью ассиметричных алгоритмов для его дальнейшего использования в симметричных.

Ключ обычно является короткой строкой и от того, сколько времени будет потрачено на обмен ключами, не сильно зависит общая скорость. Поэтому для обмена ключами могут использоваться самые надежные, но медленные реализации.

Для того чтобы только передать ключ, может использоваться любой ассиметричный алгоритм. Например:

1) Алиса генерирует открытый и закрытый ключи, публикует и/или передает открытый ключ Бобу;

2) Боб генерирует ключ симметричного шифрования, шифрует открытым ключом Алисы и передает ей.

Если канал только прослушивается, но злоумышленник не может подменить передаваемые сообщения, то такой алгоритм может использоваться без опасений.

Протоколы распределения ключей в отличие от протоколов передачи сообщений позволяют использовать более простые операции, не требующие обратимости вычислений, либо сохранять в тайне не только закрытые ключи, но и части открытых.

Схема обмена ключами Диффи-Хеллмана

В схеме Диффи-Хеллмана генерируемый ключ зависит от случайных чисел, выбираемых участниками протокола.

В классическом варианте схема основана на проблеме дискретного логарифмирования. Схему El Gamal при этом считают ее незначительным расширением [8].

1. Алиса и Боб договариваются о модуле p и числе g (p – простое, $g < p$).

2. Алиса выбирает число a, вычисляет $g^a \bmod p$ и передает результат Бобу. Число a сохраняет в секрете.

3. Боб выбирает число b, вычисляет $g^b \bmod p$ и передает результат Алисе. Число b сохраняет в секрете.

4. Алиса вычисляет $(g^b)^a \bmod p$, возводя полученное от Боба $g^b \bmod p$ в степень a.

5. Боб вычисляет $(g^a)^b \bmod p$, возводя полученное от Алисы $g^a \bmod p$ в степень b.

В результате они оба получают одну и ту же строку $g^{ab} \bmod p$, из которой может быть каким-либо заранее определенным алгоритмом вычислен секретный ключ для симметричного шифрования.

Один из участников может заранее опубликовать $g^a \bmod p$, сократив этим число передач данных между участниками.

Данный протокол можно применить к группе точек эллиптической кривой [19].

Алиса и Боб согласуют между собой уравнение кривой E и точку на кривой Q. Это открытая информация.

1. Алиса выбирает число a, вычисляет aQ и пересылает его Бобу.
2. Боб выбирает число b, вычисляет bQ и пересылает его Алисе.
3. Алиса вычисляет $a(bQ)$.
4. Боб вычисляет $b(aQ)$.

В результате они получают координаты точки abQ, которые заранее определенным способом преобразуют в ключ.

Схема Hughes

Вариант схемы Диффи-Хеллмана Hughes позволяет передавать заранее определенный ключ, но не произвольный. Такая модификация может быть полезна, если ключ надо разделить между несколькими сторонами [11].

1. Алиса выбирает простой модуль n и число $g < n$. Выбирает случайное x и вычисляет ключ:

$$k = g^x \bmod n.$$

Сообщает Бобу g и n.

2. Боб выбирает случайное, большое y, вычисляет и передает Алисе

$$Y = g^y \bmod n.$$

3. Алиса вычисляет и передает Бобу

$$X = Y^x \bmod n.$$

4. Боб вычисляет $z = y^{-1} \bmod \varphi(n)$ и $k = X^z \bmod n$.

Трехпроходный протокол Шамира

Этот протокол позволяет передавать сообщения без предварительной передачи ключей. Вообще передача ключей не требуется. Если использовать алгоритм аналогичный RSA, то ни e ни d не передаются в открытом виде. Требуется только открытое согласование модуля.

Для этого протокола может использоваться любой коммутативный симметричный или ассиметричный шифр. То есть такой, для которого выполняется $E_A(E_B(M)) = E_B(E_A(M))$ [11]. Ключ шифрования может однозначно определять ключ расшифрования, но при этом по открытому тексту и шифротексту должно быть невозможно (сложно) вычислить

ключ. Не может использоваться коммутативный одноразовый блокнот, так как открытый текст и шифротекст сразу определяют ключ.

1. Алиса шифрует M своим ключом и передает результат Бобу

$$C_1 = E_A(M).$$

2. Боб шифрует полученный шифротекст C_1 своим ключом и передает результат Алисе

$$C_2 = E_B(C_1) = E_B(E_A(M)).$$

3. Алиса расшифровывает полученный шифротекст C_2 своим ключом и передает результат Бобу

$$C_3 = D_A(C_2) = D_A(E_B(C_1)) = D_A(E_B(E_A(M))) = E_B(M).$$

4. Боб расшифровывает C_3 своим ключом и получает M

$$M = D_B(C_3) = D_B(E_B(M)) = M.$$

Здесь злоумышленник может получить следующие пары шифротекст – открытый текст: $C_2 = E_B(C_1)$ или $C_1 = D_B(C_2)$ и $C_3 = D_A(C_2)$ или $C_2 = E_A(C_3)$. Если злоумышленник найдет один из ключей, то он найдет M.

Для данного протокола подойдут алгоритмы на основе RSA или эллиптических кривых.

Так, вариант на основе RSA независимо предложили Ади Шамир и Джим Омура.

Выбирается большое простое p.

Выбирается ключ шифрования e, и вычисляется ключ расшифрования $d = e^{-1} \bmod p - 1$.

Шифрование и расшифрование осуществляются по уже привычным формулам:

$$C = M^e \bmod p,$$

$$M = C^d \bmod p.$$

Числа e и d могут быть легко получены друг из друга, поэтому такой вариант шифрования нельзя считать ассимметричным, но при этом для получения e из C и M требуется решение задачи дискретного логарифмирования.

Атака «человек посередине»

Большинство протоколов обмена ключами подвержено атаке с модификацией сообщений типа «человек посередине». Возможность атаки основана на том, что если стороны не имеют общего разделяемого секрета, то они не могут узнать, с кем установлено безопасное соединение. Если все сообщения, посылаемые от Алисы к Бобу и от Боба к Алисе, могут быть изменены, злоумышленником (Малис), то может оказаться,

что установлено два безопасных соединения между Алисой и Малис и между Малис и Бобом.

Если перед началом связи Алиса и Боб не знают общего секрета, то все их сообщения могут быть модифицированы так, что они будут считать, что успешно обменялись ключами друг с другом, а на самом деле обменялись ключами с Малис.

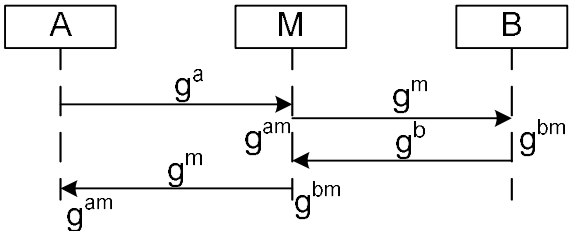


Рис. 40. Схема атаки «человек посередине»

Рассмотрим атаку на классический протокол Диффи-Хеллмана [12].

1. Алиса вычисляет g^a и отправляет Бобу.

2. Малис получает g^a. Вычисляет g^{am}. Отправляет Бобуg^m.

3. Боб вычисляет g^{mb}. Отправляет Алисе g^b.

4. Малис получает g^b. Вычисляет g^{bm}. Отправляет Алисе g^m.

5. Алиса получает g^m. Вычисляет g^{ma}.

Таким образом, Алиса и Малис имеют общий ключ g^{ma}, а Боб и Малис – общий ключ g^{mb}.

Для того чтобы решить проблему таких атак, необходимо использовать коды аутентификации сообщений, цифровые подписи или общий секрет, известный до начала взаимодействия.

В любом случае необходима некоторая, заранее известная информация. Если используются цифровые подписи, то открытый ключ для ее проверки также должен быть получен по каналам связи, которые также могут быть небезопасными.

В большинстве случаев ключ проверки подписи достаточно опубликовать.

Протокол «точка-точка»

Этот протокол предполагает, что у Алисы есть заслуживающий доверия открытый ключ Боба, а у Боба – открытый ключ Алисы [11].

Алиса генерирует число $x = g^a$ и посылает его Бобу.

Боб генерирует число $y = g^b$. Используя протокол Diffie-Hellman, он вычисляет общий ключk на основеx и b. Подписываетx и y и шифрует подпись ключомk. Затем он посылаетy и зашифрованную подпись Алисе.

$$y, E_k(S_B(x, y)).$$

Алиса также вычисляет k на основе y и a. Расшифровывает подпись Боба и проверяет ее. Затем Алиса посылает Бобу подписанное сообщение, состоящее из x и y, зашифрованных общим ключом k.

$$E_k(S_A(x, y)).$$

Боб расшифровывает сообщение и проверяет подпись Алисы.

Если открытые ключи действительно получены из надежного источника и подпись не может быть подделана за приемлемое время, то можно считать что этот протокол защищает от атаки «человек посередине». «Приемлемое время» – это время от момента публикации ключа, так как злоумышленник может попытаться заранее вычислить закрытый ключ по открытому.

Контрольные вопросы

1. Сравните протокол Диффи-Хеллмана и трехпроходный протокол Шамира, их преимущества и недостатки.
2. Какие условия должны быть выполнены, чтобы пользователи могли защититься от атаки «человек посередине»?

ТЕМА 15. ЭЛЕКТРОННО-ЦИФРОВЫЕ ПОДПИСИ

Общие сведения

Ассиметричная криптография позволяет любому пользователю системы (законному или незаконному) зашифровать свое сообщение открытым ключом получателя. Наличие шифрования при этом не дает никакой информации об отправителе, и «человек посередине» может перешифровывать все сообщения.

Защититься от этого, как уже было показано выше, позволяют цифровые подписи, вычисляемые на основе секретного клоча отправителя и проверяемые с помощью открытого ключа отправителя. Таким образом, при надежном функционировании системы, только отправитель может поставить свою подпись и любой желающий может удостовериться, что это именно его подпись.

Идея подписей была предложена в 1976 г. Диффи и Хеллманом, в первой же статье по ассиметричной криптографии. Можно считать, что без схем цифровых подписей, шифрование сообщений, открытыми ключами, не имело бы такого большого значения. Даже больше, в современном мире, проверка подлинности становится важнее секретности [10].

При использовании надежных схем ЭЦП получатель может быть уверен [8]:

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

• его оригинальности, то есть в том, что сообщение было послано именно указанным отправителем;

• отсутствии ренегатства (невозможности отказа): отправитель не сможет утверждать, что не посылал сообщения.

Схемы подписей можно разделить на два основных типа, с приложением и с восстановлением сообщения.

Схему подписи с приложением можно представить в виде:

$$M, s_k(h(M)),$$

где M – сообщение; h – некоторая хеш-функция; s_k – шифрование закрытым ключом.

Здесь сообщение передается в *открытом* виде, а подпись добавляется в конец сообщения. Из подписи может быть расшифровано хеш-значение и для открытого сообщения может быть вычислено хеш-значение. Если они совпадают, значит, сообщение не было изменено и принадлежит указанному отправителю.

Подпись с восстановлением сообщения – это сообщение зашифрованное закрытым ключом:

$$s_k(M).$$

Здесь, при расшифровании подписи открытым ключом, мы получаем сообщение и при этом удостоверяемся, что подпись действительно можно расшифровать открытым ключом отправителя. Однако дальнейшие выводы можно сделать, только если согласована структура сообщения и в ней есть избыточность. Почти любую строку, имеющую определенный формат, можно расшифровать. В большинстве случаев при этом будет получен набор символов выглядящий случайным.

Эти схемы являются только основой для конструирования реальных схем ЭЦП. В схемы может добавляться шифрование, метки времени, случайные биты, может изменяться формат сообщения и т. д.

Схема ЭЦП RSA

Подписи RSA аналогичны шифрованию. Ключи e и d генерируются также и в некоторых схемах могут использоваться те же ключи, что и для шифрования.

$$S = M^d \bmod n,$$

$$M' = S^e \bmod n.$$

Подпись проверяется сравнением

$$M' = M \bmod n.$$

Если подписываются блоки M меньше n, то подпись будет с восстановлением сообщения, иначе сообщение должно передаваться отдель-

но. При использовании подписей в таком виде имеются серьезные проблемы безопасности.

Пусть Алиса хочет, чтобы Боб подписал сообщение m', которое он в здравом уме никогда бы не подписал. Она выбирает произвольное x и вычисляет $y = x^e$ и $m = ym'$. Если Боб подпишет сообщение в таком виде, то Алиса получит $m^d = y^d m'^d = x^{ed} m'^d = xm'^d$. Вычислив $m^d x^{-1} = m'^d$, она получит подпись сообщения m'.

Если для расшифрования и подписи используется один и тот же ключ, злоумышленник сможет расшифровать отправленный законному получателю текст, попросив подписать ни к чему не обязывающую строку.

Пусть злоумышленник перехватил шифротекст $C = m^e$. Он вычисляет Cx^e и передает для подписи. В результате он получает $C^d x^{ed} = mx$, из которого без труда вычисляет x.

Использование различных пар ключей для шифрования и подписей не избавляет систему от всех недостатков, но требует дополнительной инфраструктуры ключей. Подпись хеш-значений позволяет обезопаситься от описанных атак, но добавляет возможность атак на хеш-функции.

Если используется хеш-функция, дающая выход в 256 бит, а длина модуля в RSA составляет 2048 бит, то хеш-значение должно быть «растянуто», так как использование коротких сообщений опасно. В случае RSA оно может быть просто дополнено случайными битами [9].

Таким образом, целесообразно использовать алгоритм ЭЦП RSA в виде:

$$M, s_k(h(M), rnd).$$

Схема ЭЦП El Gamal

Аналогично схеме шифрования El Gamal, выбирается простое число p и два случайных числа g и x меньше p. Вычисляется

$$y = g^x \bmod p.$$

Открытым ключом является y, g и p. Закрытым – x. Подписью сообщения M является пара a, b:

$$a = g^k \bmod p;$$

b вычисляется из уравнения

$$M = xa + kb \bmod p - 1.$$

Или $b = (M - xa)k^{-1} \bmod p - 1 = (M + (p - 1 - x) \square a) \square k^{-1} \bmod p - 1.$

Для проверки подписи нужно убедиться в верности

$$y^a a^b = g^M (\bmod p).$$

Как и при шифровании, каждый новый блок подписи требует нового значения k. Если злоумышленнику станет известно значение k, то он

сможет вычислить x. Если к злоумышленнику попадут два сообщения подписанных при помощи одного и того же k, то он сможет вычислить x, не зная k.

Эта схема не может быть использована как схема с восстановлением сообщения. Для того чтобы проверить подпись, нужно точно знать, что именно подписывалось. Если хеш дополняется случайными битами, то и они должны быть переданы в открытом виде, либо биты должны быть неслучайными.

Схема ЭЦП DSA

DSA (Digital Signature Algorithm) разработан в 1991 г. и был принят в качестве федерального стандарта США (DSS – Digital Signature Standard).

Выбирается простой модуль p длиной L кратной 64 бит в диапазоне от 512 до 1024 бит.

Выбирается q длиной 160 бит, являющееся множителем p–1.

Выбирается $h < p$.

Вычисляется $g = h^{\frac{p-1}{q}} \bmod p$.

Выбирается $x < q$.

Вычисляется $y = g^x \bmod p$.

Параметры p, q, g являются открытыми и могут быть общими для группы пользователей; y является открытым ключом, x – закрытым.

Отправитель вычисляет подпись, состоящую из двух частей.

Генерируется случайное $k < q$. Для каждой подписи k должно быть новым. Вычисляется пара r и s, являющаяся подписью:

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

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

Получатель, проверяя подпись, вычисляет:

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

$$u_1 = (H(m) \cdot w) \bmod q,$$

$$u_2 = rw \bmod q,$$

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

Как о российских государственных стандартах шифрования, так и об американских можно слышать, что в них оставлены «лазейки» (back-doors). Для данного стандарта выявлена возможность более простого взлома при некоторых слабых значениях p и q. Но алгоритм получения этих чисел открыт, и если следовать ему, то вероятность получения слабых чисел очень мала [11].

Обобщенная схема ЭЦП на проблеме дискретного логарифмирования

Схемы подписей El Gamal и DSA основаны на проблеме дискретного логарифмирования, но они являются только частными случаями более общей схемы ЭЦП.

Аналогично схеме DSA выбираем такие g, q и p, что p – простое, $g^q = 1 \,(mod\ p)$. Либо $q = p - 1$, а g – любое число, меньше p, либо q – множитель $p - 1$, а $g = h^{\frac{p-1}{q}} \ mod\ p$, где h – любое. Закрытым ключом является $x < q$, открытым – $y = g^x \ mod\ p$.

Для того чтобы подписать сообщение m, необходимо вычислить

$$r = g^k \ mod\ p;$$

найти s из уравнения

$$ak = b + cx \ (mod\ q),$$

где a, b и c могут быть связаны с различными значениями.

Подпись верна, если

$$r^a = g^b y^c \ (mod\ p).$$

Значения a, b и c могут быть выбраны из следующей таблицы. Здесь $r' = r \ mod\ q$.

Таблица 14

Возможные перестановки значений *a*, *b* и *c*

1	± r`	± s	m
2	± r`m	± s	1
3	± r`m	± ms	1
4	± mr`	± r`s	1
5	± ms	± r`s	1

Здесь шапка таблицы не фиксирована. Может быть сформировано 6 вариантов сопоставления строки таблицы и переменных a, b и c, без учета знака. Всего таблица определяет 120 вариантов схем подписей. Кроме того, каждая из них может быть приведена в вид DSA, то есть, добавлено приведение по модулю q [11].

Например, можно получить уравнение подписи $sk = m + r`x \ (mod\ q)$ и уравнение проверки $r^s = g^m y^{r'} \ (mod\ p)$. Или уравнение подписи $k = -r`m + msx \ (mod\ q)$ и уравнение проверки $r = g^{-r`m} y^{ms} \ mod\ p$. Первая соответствует схеме DSA, только способ проверки в DSA сложнее из-за использования двух модулей. Способ приведения любой из этих схем к виду DSA, смотрите в [11].

Российский стандарт на ЭЦП использует эллиптическую криптографию. Этот стандарт заменил использовавшийся ранее ГОСТ Р 34.10-94, основанный на проблеме дискретного логарифмирования. Эллиптическая криптография имеет несомненное преимущество: тот же уровень безопасности может быть получен при меньших длинах ключей, в сравнении с дискретным логарифмированием, и тем более RSA.

В стандарте ЭЦП используется уравнение кривой того же вида, и так же определена операция сложения, как и при шифровании, рассмотренном ранее.

p – модуль эллиптической кривой, удовлетворяющий неравенству $p > 2^{255}$.

Кривая E – $y^2 = x^3 + ax + b \mod p$, определяемая инвариантом $J(E)$ или коэффициентами a и b.

Кроме того, необходимо вычислить еще значения m и q.

Множество всех точек эллиптической кривой E, вместе с нулевой точкой, образуют конечную абелеву (коммутативную) группу порядка m. То есть m – количество всех точек эллиптической кривой.

Значение m находится в диапазоне

$$p + 1 - 2\sqrt{p} \le m \le p + 1 + 2\sqrt{p}.$$

Должно выполняться $m \neq p$.

Способ вычисления m не описан в ГОСТ, как и в большинстве книг по криптографии, но может быть найден в статьях посвященных этой проблеме. В [33] приведено несколько способов вычисления порядка. Простые для понимания, но не приемлемые по затратам времени способы, требуют для произвольной точки R, нахождения такого m, что $mR = O$. Для понимания алгоритма ЭЦП необходимо иметь в виду это соотношение.

Определяется простое число q – порядок циклической подгруппы группы точек эллиптической кривой E. Для q выполнены следующие условия:

$$\begin{cases} m = nq, n \in Z, n \ge 1 \\ 2^{254} < q < 2^{256} \end{cases}.$$

Если m – простое, то $q = m$. Если q не попадает в требуемый диапазон, то должны быть выбраны другие параметры кривой.

Порядок группы m (или подгруппы q) играет здесь такую же роль, что и $\varphi(p)$ при возведении в степень.

На кривой выбирается точка $P \neq O$, для которой $qP = O$. Такое равенство должно выполняться для всех точек. Если проводить аналогию с возведением в степень, то $g^{\varphi(p)} \mod p = 1$ и $qP = O$; $g^{\varphi(p)+1} \mod p = g$ и $(q+1)P = O + P = P$.

Параметры E, p, q и точка P являются общедоступными и могут быть фиксированными для группы пользователей.

Закрытым ключом является число $d < q$.

Открытым ключом является точка $Q = dP$.

Формирование ЭЦП можно представить следующей последовательностью действий.

1. Вычислить хэш-значение сообщения M: $a = h(M)$. Должна использоваться хеш-функция, определенная в ГОСТ Р 34.11-94. Длина ее выхода составляет 256 бит.

2. Определить $e = a \mod q$. Если $e = 0$, то $e = 1$.

3. Сгенерировать случайное целое число k, удовлетворяющее неравенству $0 < k < q$.

4. Вычислить точку эллиптической кривой $C = kP$ и определить $r = x_c \bmod q$, где $x_c - x$ координата точки C.

5. Если $r = 0$, то вернуться к шагу 3.

6. Вычислить значение $s = rd + ke \bmod q$. Если $s = 0$, то вернуться к шагу 3.

7. Определить цифровую подпись как конкатенацию двух двоичных векторов $\xi = \mathrm{r} \| s$. r и s должны быть выражены в виде 256-битных векторов.

Для проверки ЭЦП необходимо выполнить:

1. По полученной подписи ξ вычислить целые числа r и s. Если выполнены неравенства $0 < r < q$, $0 < s < q$, то перейти к следующему шагу. В противном случае подпись неверна.

2. Вычислить хэш-значение полученного сообщения M: $a = h(M)$.

3. Определить $e = a \mod q$. Если $e = 0$, то $e = 1$.

4. Вычислить значение $v = e^{-1} \mod q$.

5. Вычислить значения $z_1 = sv \mod q$, $z_2 = -rv \mod q$.

6. Вычислить точку эллиптической кривой $C = z_1 P + z_2 Q$ и определить $R = x_c \mod q$.

7. Если выполнено равенство $R = r$, то подпись принимается, в противном случае подпись неверна.

Подсознательный канал

В некоторых схемах подписи может присутствовать так называемый подсознательный канал, позволяющий передавать блок скрытого сообщения в цифровой подписи [11].

Получатель этого сообщения в большинстве случаев должен знать закрытый ключ отправителя или часть закрытого ключа. Подсознательный канал основан на том, что во многих алгоритмах используются случайные числа и размер подписи больше подписываемого сообщения. Избыточность позволяет скрыто передать блок сообщения. В схеме

подписи El Gamal, в качестве скрытого сообщения может использоваться k. Получатель этого сообщения для восстановления k должен знать секретный ключ x.

В схеме DSA если получатель знает ключ x, скрытое сообщение может передаваться в k (до 160 бит).

Если получатель не знает k, то несколько бит может передаваться в $a = g^k \bmod p$ для El Gamal или $r = (g^k \bmod p) \bmod q$ для DSA. Длина r – также 160 бит, но для того чтобы скрыть в нем несколько бит, необходимо перебирать k. Например, информационным может быть остаток по некоторому известному получателю модулю, в [11] описывается использование квадратичных вычетов. Учитывая, что r распределено равномерно, для подбора нужного значения r, n бит которого – информационные, понадобится порядка 2^n проверок. Подбор 16 информационных бит не будет проблемой и будет выполняться за несколько секунд.

Опасность подсознательного канала, для доступа к которому не нужен ключ, состоит в том, что программа или микросхема, выполняющая шифрование и подпись сообщений, при недобросовестной реализации, может раскрывать несколько бит ключа за подпись. Эта информация, при получении подписанных документов будет доступна производителю программы или микросхемы.

Контрольные вопросы

1. Покажите, как может быть раскрыто x при известном k в схеме El Gamal. Как может быть раскрытоx, если два сообщения подписаны с помощью одного и того же k?
2. Выполните преобразования, показывающие, что DSA работает.
3. Выполните преобразования, показывающие, что работает алгоритм ГОСТ 34.10-2001.
4. Преобразуйте схему подписи ГОСТ 34.10-2001 к схеме подписи на основе дискретного логарифмирования.
5. Оцените возможности использования подсознательного канала в алгоритме ГОСТ 34.10-2001.

ТЕМА 16. ИНФРАСТРУКТУРА ОТКРЫТЫХ КЛЮЧЕЙ (PKI)

Общие принципы PKI

До тех пор пока мы допускаем возможность атаки типа «человек посередине», мы не можем быть уверены в источнике полученных сообщений. Если используется ассиметричная криптография, и отправитель передал свои открытые ключи, мы также не можем быть уверены в том,

кто сформировал и отправил эти ключи. Необходимо, чтобы эти ключи были получены из источника, которому можно доверять.

Вообще, если считать, что все передаваемые сообщения могут быть модифицированы, то доверять нельзя никому, точнее нельзя доверять ни одному сообщению, полученному через такой канал связи. Если злоумышленник может модифицировать не все каналы связи доступные пользователю, то все же есть способ выявить достоверные и недостоверные сообщения.

Инфраструктура открытых ключей (PKI – Public Key Infrastructure) построена на идее, что если у нас есть всего один ключ центра сертификации, в подлинности которого мы уверены, то мы можем обмениваться ключами с множеством людей, имеющих сертификаты.

Пусть Алиса хочет передать Бобу сообщение, подписанное его открытым ключом, или проверить его подпись. Для этого она должна получить открытый ключ Боба. При этом существует доверенная третья сторона (TTP – trusted third party), которой доверяют Алиса и Боб. При рассмотрении инфраструктуры открытых ключей, доверенной стороной является центр сертификации (ЦС или CA – Certificate Authority). Чтобы использовать инфраструктуру открытых ключей, все пользователи должны быть уверены, что обладают открытым ключом ЦС. Основания для такой уверенности могут быть различными. Они могли получить его вместе с шифрующим программным обеспечением или запросить у самого ЦС с использованием различных каналов связи. Например, с браузером может идти несколько десятков SSL сертификатов, с помощью которых проверяется подлинность сервера при установке https соединений. В любом случае процесс распределения ключа корневого центра сертификации не может быть полностью защищен криптографическими методами, если все имеющиеся каналы связи ненадежны.

Получение сертификата

Алиса генерирует пару открытый ключ – закрытый ключ и отправляет открытый ключ и сведения о себе в ЦС. Центр сертификации проверяет сведения и если ничего не вызывает подозрений, то выдает сертификат, в котором указан, открытый ключ Алисы, и то, что он действительно принадлежит Алисе. Весь сертификат подписан ЭЦП центра сертификации.

При большом количестве пользователей системы, корневой (главный) центр сертификации (root CA) может иметь дочерние. Сертификаты может выдавать любой ЦС. Если сертификат выдается дочерним ЦС, то кроме собственного сертификата, Алиса должна получить сертификат дочернего ЦС, подписанный корневым ЦС.

Обмен ключами

Теперь Алиса может передать Бобу свой сертификат и сертификат ЦС (рис. 41).

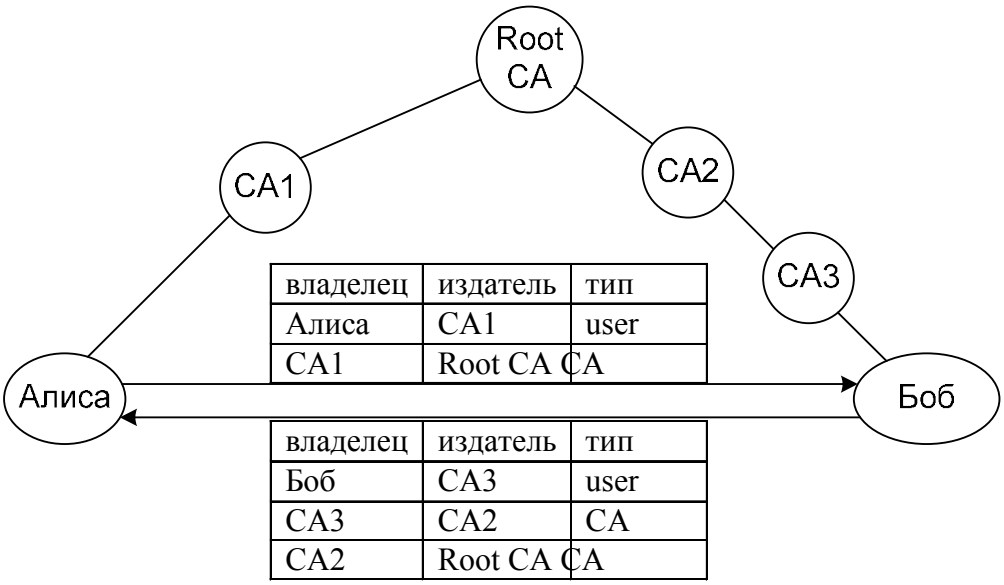


Рис. 41. Обмен открытыми ключами

Боб проверяет ЭЦП на сертификате Алисы с помощью открытого ключа указанного на сертификате CA1 и проверяет ЭЦП на сертификате CA1 с помощью открытого ключа Root CA, полученного из источника, которому можно доверять. Боб проверяет наличие сертификатов Алисы и CA1 в списке аннулированных сертификатов (CRL – certificate revocation list), и если их там нет, то принимает решение доверять ключу, полученному от Алисы.

Здесь надо заметить, что доверять можно тому, что центр сертификации действительно имеет ключ, указанный в сертификате, а не тому, что он не нарушает правил. Алиса обладает ключом, подписанным центром сертификации, но проверил ли центр сертификации ее личность? Действительно ли она является Алисой, а не злоумышленником, представляющимся Алисой? Если получить сертификат можно, без каких либо проверок, то злоумышленник может получить сертификаты на имена Алиса и Боб (может быть, с немного отличающимися анкетными данными). Если злоумышленнику удалось раскрыть ключ центра сертификации, то он может выписать для себя сертификаты на любые имена и с любыми данными и самому стать центром сертификации. Теперь он может провести атаку «человек посередине», передав и Алисе и Бобу свои открытые ключи, подтвержденные сертификатами. Для решения этой

проблемы, необходимо однозначное сопоставление имени – некоторому человеку или организации, что также является проблемой. Если общение осуществляется через почтовый протокол, то сертификат можно привязать к почтовому ящику, если же нет, то это всего лишь еще одно поле анкеты. Боб не сильно удивится, если он знает что у Алисы десяток ящиков и вместо привычного alisa@gmail.com увидит alisa@yahoo.com. Можно сделать вывод, что система передачи сообщений должна основываться на тех же именах, которые являются идентифицирующими в сертификатах. Если это надстройка над почтовым протоколом, то сертификаты привязываются к ящику, если SSL то к домену, если это внутренняя система, то имена и адреса должны быть сопоставлены также однозначно. Однако точность адресации не станет препятствием, если злоумышленник может создавать сертификаты на любые имена.

SSL сертификаты привязываются к доменным именам, что однозначно идентифицирует владельца и решают проблему имен. Однако если создан поддельный сайт с именем и оформлением похожим на другой, то наличие у него SSL сертификата не говорит о том, что сайт подлинный. Но, если, заходя на сайт крупной компании, вы получаете сообщение о невозможности проверки сертификата, то это может быть поводом считать сайт поддельным.

На рис. 41 приведена таблица с передаваемыми от Алисы к Бобу и обратно сертификатами. В столбце «тип», указано, является ли владелец центром сертификации. Если пользовательское ПО не проверяет это поле, то любой пользователь, получивший сертификат может подписать другие сертификаты, выступив в роли центра сертификации (такая проблема была в библиотеках Windows, которые использовались в Internet Explorer [9]).

Срок жизни

Сертификаты имеют ограниченный срок жизни. В сертификате указывается дата начала и окончания действия. Чем меньше срок действия сертификата, тем меньше времени у злоумышленника для взлома ключей, но в тоже время необходимо большее число взаимодействий с центром сертификации для обновления сертификатов. Сами ЦС имеют сертификаты с большими сроками жизни (для SSL – 10–25 лет), так как смена сертификата корневого ЦС связана с необходимостью обновления сертификата всеми участниками системы. Пользовательские сертификаты могут иметь сроки жизни от нескольких часов (сертификат постоянно обновляется, а ключи служат только для выполнения нескольких транзакций) до нескольких лет.

Отзыв ключей

Если ключ Алисы был украден, то она обращается в центр сертификации и просит внести ее сертификат в список аннулированных сертификатов (CRL). Боб, проверяя перед каждым сеансом связи наличие сертификата Алисы в CRL, узнает, что сертификат отозван и запрашивает у нее новый сертификат.

CRL работает для всех транзакций, осуществляемых после внесения сертификата в список. Но, до того, как Алиса заметила кражу ее закрытого ключа, злоумышленник мог выполнить с его помощью множество транзакций (например, переводов денег со счета Алисы) и Алиса не может отказаться от них. Более того, в алгоритмах ЭЦП обычно не присутствуют третьи стороны, которые могут подтвердить, что подпись была осуществлена в указанное в документе время. Злоумышленник сможет и после появления сертификата в CRL подписывать документы задним числом. Такое мошенничество не будет работать, если подписи используются для осуществления транзакций с платежной системой, проверяющей CRL.

По видимости инфраструктура открытых ключей никак не решает проблему подписи электронных документов, а предназначена в основном для выполнения транзакций в реальном времени. Если в системе предполагается возможность подписи документов, то ключи должны быть надежными, чтобы обеспечить защиту от атак хотя бы на несколько лет и секретные ключи никогда не должны разглашаться. Списки отзыва сертификатов и ограниченное время жизни не смогут гарантировать невозможность подписи задним числом. Или наоборот, невозможно подписать договор длительного срока действия ключом с коротким сроком жизни.

Для решения проблемы подписи задним числом используются дополнительные надстройки над форматом ЭЦП и стандартной структурой PKI. В соответствии с RFC 5126 «CMS Advanced Electronic Signatures» (CAdES) для подтверждения времени подписания используется служба точного времени. Документ с подписью в этом случае должен иметь вид:

$$M, T, S_U(M), S_{TSP}(h(M), T),$$

где M – сообщение; T – время получения хеша $h(M)$ службой штампов времени TSP; S_U – подпись пользователя; S_{TSP} – подпись службы TSP.

Проблема PKI также и в том, что сохранением секретности закрытых ключей занимаются не государственные организации или специалисты по безопасности, а простые пользователи (ПК находятся не в выделенных помещениях, подключены к сети и на них могут быть вирусы). Если пользователь не может быть уверен в том, что сможет сохранить свой ключ, то он, и не может согласиться с принципом невозможности отказа от авторства, что сильно ограничивает области применения подписей.

Еще более серьезная проблема – взлом или кража закрытого ключа корневого ЦС. В этом случае перестает существовать практически вся инфраструктура: нельзя доверять дочерним ЦС, так как они могли быть созданы злоумышленниками, нельзя доверять ни одному пользовательскому сертификату. ЦС может только поместить свой сертификат в CRL и тем самым сообщить пользователям, что доверять больше нельзя никому. PKI должна теперь строиться почти с нуля. Должен быть каким-либо образом распространен корневой сертификат и обновлены сертификаты всех дочерних ЦС и всех пользователей.

PGP

PGP (от Pretty Good Privacy вполне хорошая секретность) – проект, включающий сейчас множество направлений, но начинавшийся с защиты электронной почты.

Распространение ключей построено на идее, что каждый пользователь может самостоятельно передавать свои открытые ключи и заверять чужие ключи. Таким образом, в терминологии PKI, каждый пользователь может выступать в роли центра сертификации [8]. Так как «сертификаты» привязываются к почтовому ящику, то проблема имен вроде бы решена, но это не совсем так. Так как отсутствует централизованная сертификация, злоумышленник может сгенерировать себе сертификаты на любые имена и email. Если теперь он может вмешаться в процесс передачи ключей и почты, то может перекодировать все сообщения своими ключами.

PGP защищает от прослушивания канала, но не от модификации сообщений. Если же злоумышленник «опоздал» на процедуру обмена ключами, то PGP защищает и от модификации.

Схему PGP можно считать безопасной в том смысле, что есть более дешевые способы чтения чужой почты, чем перешифровка всех ключей и сообщений.

Стандарт X.509

X.509 – это стандарт описывающий структуру сертификата. Стандарт является достаточно распространенным, и многие разработчики стремятся обеспечить совместимость с ним в своих приложениях. Если приложение должно работать с другими, соблюдающими этот стандарт, то это действительно необходимо. Иначе, для замкнутой системы, всегда можно использовать собственный стандарт. X.509 имеет достаточно сложную структуру, поэтому в [9] его использование считают нецелесообразным для собственных PKI.

Здесь рассмотрим поля, которые обязательно присутствуют в сертификатах (не обязательно X.509) [8].

- Номер версии стандарта, которому соответствует данный сертификат.
- Серийный номер сертификата.
- Идентификатор подписывающего алгоритма ЦС, что дает информацию об алгоритме и его параметрах.
- Имя изготовителя сертификата, то есть название ЦС.
- Срок действия сертификата в форме «не действителен до – не действителен после».
- Имя владельца. Имя должно однозначно идентифицировать владельца в системе.
- Открытый ключ субъекта, включающий в себя название используемого алгоритма, все необходимые параметры и фактическое значение открытого ключа.
- Подпись ЦС, примененная ко всем данным сертификата.

SSL

SSL (Secure Socket Layer – протокол защищенных сокетов) предназначен для обеспечения безопасности данных, передаваемых между web-узлом и пользователем. С помощью SSL могут быть защищены протоколы HTTP (HTTPS), FTP, Telnet и т. д.

Владелец web-узла генерирует ключи. У ЦС запрашивается сертификат, который используется для установления сессии и подтверждения подлинности web-узла.

- Клиент устанавливает связь с сервером через специальный порт, что является сигналом о защищенном сеансе.
- Сервер высылает клиенту сертифицированный открытый ключ.
- Клиент проверяет сертификат и решает, верить ли этому открытому ключу.
- Клиент выбирает случайный секретный ключ.
- Клиент шифрует выбранный секретный ключ с помощью открытого ключа сервера и пересылает результат серверу.
- К этому моменту клиент и сервер обладают общим сеансовым ключом.
- Сервер подтверждает клиенту свою подлинность, отвечая на его запрос с помощью общего с ним сеансового ключа.

Если сертификат создан самостоятельно, без участия ЦС, то с его помощью можно также установить безопасное соединение, защищенное от прослушивания, но нельзя подтвердить подлинность сервера и нельзя защититься от атаки «человек посередине», так как соединение может быть перешифрованно таким же самодельным сертификатом.

Контрольные вопросы

1. В чем преимущество инфраструктуры открытых ключей перед простой публикацией ключа?
2. Какими способами можно защититься от подписи задним числом?

ТЕМА 17. ПРОТОКОЛЫ АУТЕНТИФИКАЦИИ

При рассмотрении протоколов аутентификации мы исходим из положения, что два пользователя или клиент и сервер уже имеют общий секретный ключ, называемый также в данном контексте паролем. Необходимо определить, в каком виде он будет храниться и посылаться, так, чтобы подтвердить подлинность, желательно, двух сторон, и не раскрыть ключа злоумышленнику. Иногда аутентификация выполняется одновременно с обменом ключами.

Аутентификация с помощью хеш-функций

На сервере хранится хеш-значение пароля $h(K)$. В случае если злоумышленник получит доступ к серверу, ему будет сложно узнать пароль по хеш-значению.

1. Пользователь передает пароль K на сервер в открытом виде.
2. На сервере вычисляется хеш $h(K)$ и сравнивается с хранимым значением.

Такой простейший способ, может быть применим, если используется надежный канал связи или он уже надежно зашифрован, например, с помощью протокола SSL.

Другой способ защищает от перехвата во время передачи по каналам связи, но не защищает при хранении.

1. Сервер генерирует случайную строку salt (соль) и передает пользователю.
2. Пользователь вычисляет $h(K||salt)$ и передает на сервер.
3. Сервер сравнивает полученное и вычисленное значение.

Эти способы применяется на большинстве web серверов.

Аутентификация с использованием симметричного шифрования

На сервере хранится пароль K в открытом виде (или зашифрованный так, что всегда можно получить открытый вид).

1. Сервер генерирует случайное число R и отправляет Алисе.
2. Алиса вычисляет $E_K(R)$ и отправляет результат.
3. Сервер вычисляет $D_K(E_K(R))$ и сравнивает с R.

Здесь пароль может быть однозначно определен из передаваемых по открытым каналам связи R и $E_K(R)$. Необходимо решить задачу нахождения ключа по открытому тексту и шифротексту. Кроме того, такая схема беззащитна, если злоумышленник получит доступ к серверу.

Аутентификация с использованием открытых ключей

На сервере хранятся открытые ключи всех пользователей.

1. Сервер генерирует случайное число R и отправляет Алисе.
2. Алиса вычисляет $E_K(R)$ с помощью своего закрытого ключа. Фактически эта схема подписи, а не шифрование.
3. Сервер вычисляет $D_K(E_K(R))$ с помощью открытого ключа Алисы и сравнивает с R.

Если Алиса использует свой закрытый ключ не только для аутентификации, но и для подписей и шифрования, то такая схема позволяет выполнить множество атак, некоторые из которых описаны выше. Особенно уязвим к таким атакам алгоритм RSA. Напомним вариант атаки, для этой схемы.

1. Злоумышленник представляется сервером и отправляет Алисе текст, зашифрованный ее открытым ключом (возможно с преобразованиями, к которым можно позже применить обратные). Либо текст, который он хочет подписать.
2. Алиса подписывает эту строку, считая ее случайной.
3. Злоумышленник получает расшифрованный текст или подпись.

Самый простой способ обезопаситься от таких атак – просто не использовать одну и туже пару ключей для различных целей (для подписи одна пара, для шифрования другая, для аутентификации третья).

Безопасные протоколы аутентификации на основе открытых ключей разрабатываются применительно к конкретным алгоритмам, и используется схема, отличная от схемы подписи. Тем не менее, схемы подписи и схемы идентификации взаимозаменяемы.

В зависимости от условий предпочтение может быть отдано меньшим длинам ключа или меньшему числу шагов.

Например, в [11] приведена схема SCHNORR Клауса Шнорра

Параметрами являются числа p – простое, q – множитель p–1 и a, такое, что $a^q \bmod p = 1$. Закрытым ключом является случайное $s < q$, открытым – $v = a^{-s}$.

1. Алиса выбирает случайное $r < q$, вычисляет $x = a^r$ и отправляет на сервер.
2. Сервер выбирает случайное e и отправляет Алисе. Длина числа e должна быть меньше длины q в 2 раза.
3. Алиса вычисляет $y = r + se$ и отправляет на сервер.
4. Сервер проверяет, что $x = a^y v^e$.

Этот алгоритм может быть преобразован в схему подписи, заменой $e = H(M, x)$. Подпись будет состоять из e и y. А проверка из вычисления $x = a^y v^e$ и $e = H(M, x)$.

Очевидно, что если такая схема аутентификации будет использоваться одновременно с подписями, то злоумышленник сможет подписать любой текст, так как для любого текста сможет сформировать e с учетом выбранного Алисой x и получить в ответ y. Но если длина хеш больше длины e, как определено в схеме, то выполнить такую атаку не получится.

Преимуществом таких схем является то, что ни передаваемое по сети, ни хранимое на сервере значение не позволяют в приемлемые сроки восстановить ключ.

Аутентификация в схемах обмена ключами

При рассмотрении обмена ключами, уже был рассмотрен протокол «точка–точка». Он позволяет проверить подлинность обоих участников на основе проверки правильности подписей.

Рассмотрим алгоритм EKE (Encrypted Key Exchange), позволяющий произвести секретный обмен ключами, не раскрывая почти ничего о пароле [11]. Если обмен ключами произведен удачно и пользователи смогли зашифровать и расшифровать случайные строки, то они использовали одинаковые пароли.

1. Алиса генерирует пару «открытый ключ/закрытый ключ». Она шифрует открытый ключ PK с помощью симметричного алгоритма E, используя P в качестве ключа: $E_P(PK)$. Ключ является открытым только с точки зрения алгоритма, но этот ключ не должен стать известен злоумышленнику. Алиса посылает Бобу

Имя, $E_P(PK)$

2. Боб знает P. Он расшифровывает сообщение, получая PK. Затем он генерирует случайный сеансовый ключ K и шифрует его с помощью ассиметричного алгоритма PE, открытым ключом PK, который он получил от Алисы, а затем используя P в качестве ключа. Он посылает Алисе

$E_P(PE_{PK}(K))$.

3. Алиса расшифровывает сообщение, получая K. Теперь необходимо проверить, что обе стороны имеют одно и то же значение K. Алиса генерирует случайную строку R_A, шифрует ее с помощью K и посылает Бобу

$E_K(R_A)$

4. Боб расшифровывает сообщение, получая R_A. Он генерирует другую случайную строку, R_B, шифрует обе строки ключом K и посылает Алисе результат.

$E_K(R_A, R_B)$.

5. Алиса расшифровывает сообщение, получая R_A и R_B. Если строка R_A, полученная от Боба, – это та самая строка, которую она послала Бобу на этапе 3, она, используя K, шифрует R_B и посылает ее Бобу.

$E_K(R_B)$

6. Боб расшифровывает сообщение, получая R_B. Если строка R_B, полученная от Алисы, – это та самая строка, которую он послал ей на этапе 4, то он может быть уверен в правильности обмена ключами. Теперь обе стороны могут обмениваться информацией, используя K в качестве сеансового ключа.

При использовании этого протокола, злоумышленнику очень сложно проверить, правильно ли он угадал значение P. Для этого ему нужно либо угадать еще и K, либо найти открытый ключ и вычислить K. И только значение K можно проверить из $E_K(R_A)$, $E_K(R_A, R_B)$ и $E_K(R_B)$.

Для того чтобы проверить значение K, для каждого K надо выполнить как минимум два расшифрования симметричного алгоритма. Если K – случайно и большой длины, то эта задача не может быть решена за приемлемое время.

Если найдено K, то для проверки каждого P необходимо выполнить два расшифрования симметричного алгоритма и одно шифрование ассиметричного. До тех пор пока не вскрыто K или PK, злоумышленник не сможет проверить ни одного пароля.

Серверы Ключей

Другой подход к установлению безопасных соединений и проверки пользователей связан с запросом ключей у доверенной третьей стороны (TTP). Если в PKI, ЦС только подтверждает подлинность ключа своей подписью, то в схемах с серверами ключей (key server), TTP непосредственно участвует в генерации ключа.

Если, существует множество пользователей системы. Каждый пользователь разделяет с сервером секретный ключ, известный только ему и серверу. Если пользователь A хочет установить связь с пользователем B, то необходимо при помощи сервера сгенерировать ключ K_{AB}, который будет использоваться для установления соединения между A и B. Наиболее распространенным протоколом этого типа является KERBEROS.

KERBEROS

В протоколе KERBEROS принимают участие четыре стороны.

AS – authentication server – сервер аутентификации, также называемый KDC – key distribution server – сервер распределения ключей – хранит базы клиентов и их ключи.

TGS – ticket-granting server – сервер выдачи мандатов – может хранить права доступа клиентов к службам серверов.

Client – клиент, желающий получить доступ к серверу.

Server – сервер, непосредственно предоставляющий интересующий клиента сервис.

Определим используемые обозначения:

c – клиент;

s – сервер;

n – неповторяющееся случайное число (nonce);

a – сетевой адрес;

v – срок действия мандата (начало-окончание);

t – метка времени;

K_x – секретный ключ x;

$K_{x,y}$ – сеансовый ключ для x и y;

{M}K – M шифрованное ключом K;

$T_{x,y}$ – мандат выданный x на использование y;

$A_{x,y}$ – удостоверение x для y;

Информация о клиенте передается с помощью мандатов и удостоверений. Мандат клиенту выдается сервером аутентификации (AS) для обращения к серверу выдачи мандатов (TGS) или TGS на обращение серверу. Мандат имеет следующий формат:

$T_{c,s} = \{c, a, v, K_{c,s}\}$.

Мандат зашифрован ключом получателя и клиент не может влиять на его содержимое, а может только передать его. Клиент может использовать мандат на доступ к серверу много раз, но с одного и того же адреса, и пока не истечет срок действия. Изменить мандат клиент не может, так как не знает ключа сервера. Мандат передается в виде $\{T_{c,s}\}K_s$.

Удостоверение клиент может генерировать самостоятельно. Удостоверение имеет формат $A_{c,s} = \{c, t, K2_{c,s}\}$ и передается в виде $\{A_{c,s}\}K_{c,s}$.

K2 – необязательное поле – дополнительный сеансовый ключ.

Протокол KERBEROS можно описать следующей последовательностью шагов (рис. 42) [11,34]:

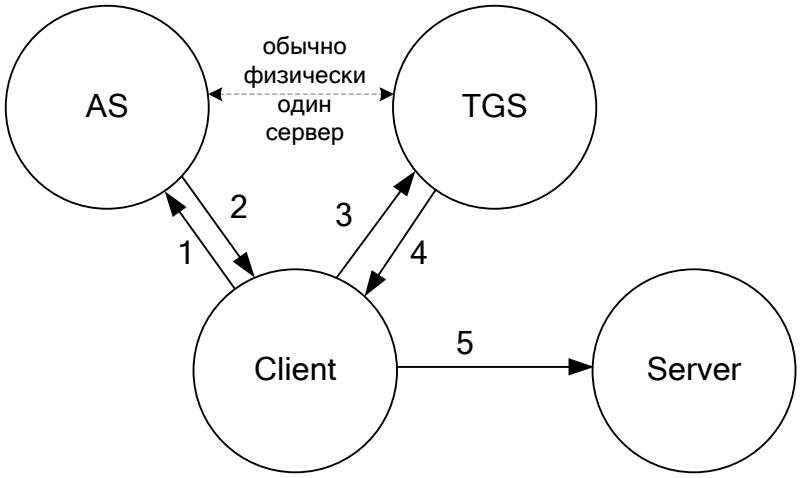


Рис. 42. Схема протокола KERBEROS

1. Запрос мандата на выделение мандата: клиент запрашивает у AS доступ к TGS. Сообщает свое имя, имя TGS и случайное n в открытом виде.

$c, tgs, n.$

2. Мандат выделения мандата: AS генерирует сеансовый ключ, который будет использоваться для обмена данными между клиентом и TGS. AS шифрует этот сеансовый ключ и полученное от клиента n секретным ключом клиента. Затем он создает для клиента TGT (Ticket Granting Ticket – мандат на выделение мандата), доказывающий подлинность клиента TGS, и шифрует его секретным ключом TGS. AS посылает эти два зашифрованных сообщения клиенту.

$\{K_{c,tgs}, n\}K_c, \{T_{c,tgs}\}K_{tgs}.$

3. Запрос мандата сервера: клиент расшифровывает свой сеансовый ключ $K_{c,tgs}$, проверяет правильность расшифровки с помощью n и шифрует с помощью $K_{c,tgs}$ удостоверение для TGS, пересылает мандат TGT, полученный от AS, имя сервера и новое случайное n.

$\{A_{c,tgs}\}K_{c,tgs}, \{T_{c,tgs}\}K_{tgs}, s, n.$

4. Мандат сервера: TGS расшифровывает мандат TGT и проверяет адрес клиента и срок жизни. С помощью содержащегося в TGT ключа $K_{c,tgs}$ расшифровывает удостоверение и сверяет метку времени с текущим временем и запоминает ее. Если та же метка времени в удостоверении будет использована повторно, то TGS не будет обрабатывать такой запрос. Если клиенту может быть предоставлен запрашиваемый сервис, TGS генерирует сеансовый ключ для клиента и сервера и мандат для сервера, который шифрует секретным ключом сервера.

$\{K_{c,s}, n\}K_{c,tgs}, \{T_{c,s}\}K_s.$

5. Запрос сервиса: клиент расшифровывает сеансовый ключ для сервера с помощью сеансового ключа для TGS, проверяет правильность расшифровки с помощью n. Создает удостоверение для сервера и пересылает мандат, полученный от TGS.

$\{A_{c,s}\}K_{c,s}, \{T_{c,s}\}K_s.$

6. Сервер расшифровывает и проверяет мандат и удостоверение и если все верно, предоставляет запрошенный сервис.

Контрольные вопросы

1. Почему в схеме SCHNORR нельзя поменять местами шаги 1 и 2? Как, зная e и не зная s, можно подобрать такое r, что x и y будут проходить проверку?

2. В чем преимущества и недостатки схем с использованием серверов ключей, перед инфраструктурой открытых ключей?

ТЕМА 18. СХЕМЫ РАЗДЕЛЕНИЯ СЕКРЕТА

Общие сведения

Если существует некоторая система или информация (далее называемая секрет), к которой каждый человек из некоторой группы должен иметь доступ, то такая задача решается просто: каждому выдается ключ.

Если необходимо защитить систему так, чтобы n человек могли получить доступ к секрету только совместно, то необходимо, чтобы доступ предоставлялся, при одновременном использовании нескольких ключей. Например, системы управления стратегическим вооружением могут требовать одновременного поворота ключей, имеющихся у разных людей. Криптографические ключи можно разбить на нужное количество частей, так, чтобы их конкатенация, или их сумма давала правильный результат. Но в случае, если используется конкатенация, $K = k_1 \,||\, k_2 \,||\, ... \,||\, k_n$, то m человек, собравшись вместе, могут получить m/n длины результирующего ключа. Перебор оставшейся части становится проще. Если используется суммирование, $K = k_1 \oplus k_2 \oplus ... \oplus k_n$, то при любом $m < n$, m человек не могут получить никакой информации о разделяемом секрете.

Теперь, если один из n не хочет делиться своей частью секрета или не оказался на месте, или умер, то уже никто не может получить доступа к секрету. Чтобы избежать этого используются пороговые схемы (m, n), характеризующиеся следующими свойствами:

- Для получения доступа к секрету необходимо получение как минимум m из n частей секрета.
- Любые m частей позволяют получить один и тот же секрет.
- Менее m частей секрета не дают никакой информации о секрете.

Части секрета также называют тенями [11].

Рассмотрим два варианта схем разделения секрета. Обе идеи основываются, в конечном счете, на том, что система линейных уравнений имеет 0, 1 или бесконечное множество решений. 0 решений будет получено, если соберется более m человек и из них хотя бы один – мошенник, который либо не знает части секрета, либо пытается запутать остальных, предоставив неверное значение. Одно решение будет получено, если соберется m человек или более m честных. Если из m – хотя бы один будет нечестным, то будет получено одно решение, но оно будет неверным. Бесконечное множество решений будет получено, если соберется менее m человек.

Схема интерполяционных многочленов Лагранжа

Основной идеей схемы, предложенной Ади Шамиром, является то, что многочлен степени m–1 может быть однозначно определен m точками. Это выполняется как для поля действительных чисел, так и для конечного поля.

Если необходимо построить пороговую схему (3, n), то для выдачи теней, генерируется квадратный трехчлен вида

$$y = ax^2 + bx + M \ mod \ p,$$

если (6, n)

$$y = ax^5 + bx^4 + cx^3 + dx^2 + fx + M \ mod \ p$$

где M – разделяемый секрет; a и b – случайные числа; p – простое и больше каждого коэффициента. Тенями являются точки полученной кривой. В качестве секрета нельзя использовать все три коэффициента в виде конкатенации, так как решение имеющихся уравнений позволяет найти соотношение между коэффициентами, но использование их суммы не ослабит систему.

Пусть необходимо построить схему (3, 6), разделяемый секрет $M =$ 17 и для него сгенерированно уравнение $y = 3x^2 + 8x + 17 \ mod$ 19.

Выберем 6 точек кривой (рис. 43). Можно брать любые точки, кроме точки (0, M), так как в ней уже определено значение ключа.

$k_1 = (5,18)$, $k_2 = (11,12)$, $k_3 = (16,1)$, $k_4 = (2,7)$, $k_5 = (14,14)$, $k_6 = (15,14)$.

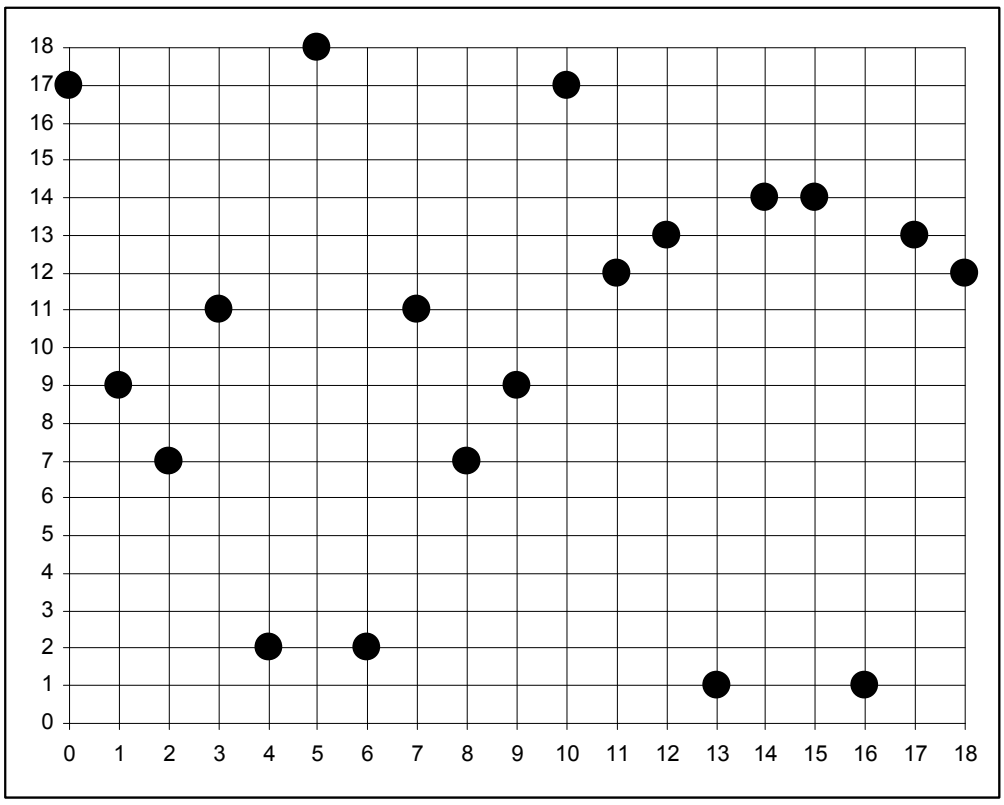


Рис. 43. График функции $y = 3x^2 + 8x + 17 \ mod$ 19

Чтобы восстановить M по 3-м теням, например по k_1, k_5, k_6, необходимо решить систему уравнений:

$$\begin{cases} a\cdot 5^2 + b\cdot 5 + M = 18 \ (mod\ 19) \\ a\cdot 14^2 + b\cdot 14 + M = 14 \ (mod\ 19) \\ a\cdot 15^2 + b\cdot 15 + M = 14 \ (mod\ 19) \end{cases} \rightarrow \begin{cases} 6a + 5b + M = 18 \ (mod\ 19) \\ 6a + 14b + M = 14 \ (mod\ 19) \\ 16a + 15b + M = 14 \ (mod\ 19) \end{cases} \rightarrow$$

$$\begin{cases} 9b = 15 \ (mod\ 19) \\ 10a + b = 0 \ (mod\ 19) \\ 16a + 15b + M = 14 \ (mod\ 19) \end{cases} \rightarrow \begin{cases} b = 15\cdot 9^{17} = 8 \ (mod\ 19) \\ a = (19 - 8)\cdot 10^{17} = 3 \ (mod\ 19) \\ M = 14 + 3a + 4b = 17 \ (mod\ 19) \end{cases}$$

Секрет $M = 17$ восстановлен по трем точкам.

Векторная схема

Схема, предложенная Джорджем Блекли, использует для решения задачи плоскости в m-мерном пространстве. m плоскостей в m-мерном пространстве обычно пересекаются одной точке. Например, две плоскости в трехмерном пространстве пересекаются по прямой, три плоскости пересекаются в одной точке, четыре плоскости либо пересекаются в одной точке, либо пересекаются по три в четырех точках.

Секретом в этой схеме является точка (одна координата точки), а тенями – плоскости.

Если секретом является точка $Q(x_0, y_0, z_0)$, то для каждого участника можно сгенерировать уравнения плоскостей из соотношения $d = ax_0 + by_0 + cz_0 \ mod\ p$, подставляя вместо a, b и c случайные числа меньшие p. Тенями являются коэффициенты a, b, c и d.

Если для схемы (3, n) собрано 3 тени, то пересечение плоскостей можно получить, решив систему линейных уравнений:

$$\begin{cases} a_1x + b_1y + c_1z = d_1 \ (mod\ p) \\ a_2x + b_2y + c_2z = d_2 \ (mod\ p) \\ a_3x + b_3y + c_3z = d_3 \ (mod\ p) \end{cases}.$$

Мошенники

Приведенные пороговые схемы не защищены от мошенничества. Мошенник может намеренно ввести неправильную тень, так как не знает верной или не хочет раскрытия секрета. Если при этом собранно m теней, то узнать, кто ввел неверную затруднительно. Если собрано $m+1$ теней, то может быть получено $m+1$ различных решений (число сочетаний по m из $m+1$). Если в секрет добавлять его контрольную сумму, то узнать верный секрет и выявить мошенника не составит труда. Кроме того, существуют специальные схемы с выявлением мошенников.

Контрольные вопросы

1. Что такое пороговая схема?
2. Как можно защититься от мошенников при применении пороговых схем?

ТЕМА 19. СИСТЕМЫ ЭЛЕКТРОННОГО ГОЛОСОВАНИЯ

Постановка задачи

Обеспечение честных выборов является сложной задачей с тех пор как они перестали умещаться в рамки одной площади. Государство хочет защититься от вброса бюллетеней, избиратели от нечестного подсчета голосов и разглашения информации о выборе. Похоже, что выборы на основе бумажных бюллетеней уже не позволят решить эти проблемы. Возможно, они будут когда-нибудь решены с помощью систем электронного голосования.

Сформулируем требования, предъявляемые к системам голосования [11].

1. Голосовать могут только те, кто имеет на это право.
2. Каждый может голосовать не более одного раза.
3. Никто не может узнать, за кого проголосовал конкретный избиратель.
4. Никто не может проголосовать вместо другого. (Это оказывается самым тяжелым требованием.)
5. Никто не может тайно изменить чей-то голос.
6. Каждый голосующий может проверить, что его голос учитывался при подведении итогов голосования. В системах голосования используются некоторые не рассмотренные ранее алгоритмы и протоколы.

Слепые подписи

Слепые подписи позволяют подписать документ, не зная его содержимого. Такие подписи могут применяться, если нотариусу все равно, что написано в подписываемом им документе, и он удостоверяет только, например, существование документа на момент подписания. Для схем голосования важно, на чем ставить подпись, но можно сократить риск подписи неправильно сформированных документов [11].

Схема таких подписей была рассмотрена ранее, но в качестве атак (когда один документ выдается за другой).

1. Алиса берет сообщение и умножает его на случайное число. Это случайное число называется маскирующим множителем. Алиса посылает замаскированный документ Бобу.

2. Боб подписывает замаскированный документ.

3. Алиса удаляет маскирующий множитель, получая оригинальный документ, подписанный Бобом.

На примере алгоритма RSA схема может выглядеть:

1. Алиса вычисляет Mx^e, где M – сообщение; x – случайный множитель; e – открытый ключ Боба.

2. Боб вычисляет $(Mx^e)^d = M^d x$.

3. Алиса вычисляет $M^d x \cdot x^{-1} = M^d$, являющееся подписью сообщения M.

Свойство таких подписей не столько в том, что Боб не знает, что он подписывает, но в том, что, получив ранее подписанное им сообщение, он не может узнать, кому он его подписывал.

Если Боб должен быть уверен, что не подписывает ничего противоречащего протоколу, можно использовать следующие схемы, снижающие риск мошенничества.

Алиса формирует n сообщений, незначительно отличающихся друг от друга информацией, которую необходимо скрыть (например, идентификационным номером). Алиса маскирует эти сообщения и передает Бобу.

1. Боб просит раскрыть маски n–1, выбранных им сообщений.

2. Алиса передает запрашиваемые маски.

3. Боб вычисляет сообщения и проверяет, что сообщения имеют правильный формат.

4. Боб подписывает оставшееся одно сообщение.

5. Алиса снимает маску с подписанного сообщения.

Таким образом, риск подписи сообщений, не отвечающих требуемому формату, снижается до $1/n$.

Схема голосования на основе слепых подписей

1. Каждый избиратель создает 10 наборов сообщений, каждый набор содержит правильный бюллетень для каждого возможного результата (например, если бюллетенем является один из трех вариантов, то каждый набор состоит из трех бюллетеней). Каждое сообщение содержит также случайным образом созданный идентификационный номер, достаточно большой, чтобы избежать путаницы с другими избирателями.

2. Каждый избиратель лично маскирует все сообщения и посылает их в ЦИК.

3. ЦИК по своей базе данных проверяет, что пользователь не присылал раньше для подписания свои бюллетени. ЦИК запрашивает у избирателя 9 из 10 маскирующих множителей.

4. ЦИК открывает 9 наборов, проверяя, что они правильно сформированы. Затем она индивидуально подписывает каждое сообщение набора

и посылает их обратно избирателю, сохраняя имя избирателя в своей базе данных, чтобы предотвратить повторную подачу бюллетеней.

5. Избиратель снимает маскировку с сообщений и получает набор бюллетеней, подписанных ЦИК.

6. Каждый избиратель выбирает один из бюллетеней и шифрует его открытым ключом ЦИК. Избиратель отправляет свой бюллетень по анонимному каналу.

7. ЦИК расшифровывает бюллетени, проверяет подписи, проверяет по базе данных уникальность идентификационного номера, сохраняет последовательный номер и подводит итоги. Она опубликовывает результаты голосования вместе с каждым идентификационным номером и соответствующим бюллетенем.

8. Каждый избиратель может проверить, что его голос учтен.

Первые пять пунктов обеспечивают защиту ЦИК от неверно сформированных бюллетеней и от попыток подписать несколько бюллетеней в одном сообщении. Но если такая атака вычислительно невозможна для используемого алгоритма, а неверные бюллетени просто не учитываются (ПО не должно позволять неправильно сформировать бюллетень, честному пользователю), то можно использовать более простую версию.

1. Избиратель формирует один бюллетень, маскирует его и отправляет ЦИК.

2. ЦИК подписывает замаскированный бюллетень и отправляет избирателю.

3. Избиратель снимает маскировку и получает подписанный бюллетень.

4. Шаги 6–8 предыдущей версии.

Этот протокол защищает ЦИК от неправомерных действий избирателей. Каждый избиратель может подписать только один бюллетень и ЦИК не может узнать, что подписал конкретный избиратель. Каждый избиратель может проверить, что его голос учтен, но не может узнать, кто отправил другие бюллетени. ЦИК может подписать себе любое количество бюллетеней и учесть их при подсчете голосов. Несколько снизить возможности для такого мошенничества, можно опубликовав списки всех, кто подписывал свои маскированные бюллетени. Тогда, каждый не голосовавший сможет проверить отсутствие себя в этом списке. Каждый сможет проверить, не появилось ли у него неизвестных ранее соседей, новых улиц в городе или новых населенных пунктов. В этом случае вброс бюллетеней будет ограничен количеством человек, подписавших бюллетени, но не голосовавших.

Раскрытие секретов «все или ничего»

Протокол раскрытие секретов «все или ничего» (all-or-nothing disclosure of secrets, ANDOS), используется для того, чтобы передать одну из множества секретных строк, так, чтобы получатель мог получить только один секрет, а отправитель не знал, который именно.

Пусть Алиса обладает множеством секретов $S_1, S_2, ..., S_k$, Боб должен получить один секрет на его выбор S_b, протокол не должен позволить Бобу получить все секреты.

Такой протокол можно реализовать на основе криптосистемы RSA.

1. Алиса шифрует все секреты своим открытым ключом RSA: $C_i = S_i^e \ mod \ n$. И предоставляет к ним доступ Бобу.

2. Боб вычисляет $C' = C_b r^e \ mod \ n$, где r – случайное, и отправляет Алисе.

3. Алиса вычисляет $P' = C'^d \ mod \ n$ и передает Бобу.

4. Боб вычисляет $S_b = P' r^{-1} \ mod \ n$.

Таким образом, Боб получает секрет с выбранным им номером, и не может узнать ничего о других секретах. Алиса не может узнать ничего о номере секрета, который получил Боб.

Схема голосования на основе ANDOS

1. ЦИК публикует список всех правомочных избирателей.
2. В течение определенного срока каждый избиратель сообщает в ЦИК, собирается ли он голосовать.
3. ЦИК публикует список избирателей, участвующих в выборах .
4. Каждый избиратель получает идентификационный номер, I, с помощью протокола ANDOS.
5. Каждый избиратель генерирует пару открытый ключ/закрытый ключ: k, d. Если v – это бюллетень, то избиратель создает и посылает в ЦИК следующее сообщение: $I, E_k(I,v)$. Это сообщение должно быть послано анонимно.
6. ЦИК подтверждает получение бюллетеня, публикуя: $E_k(I,v)$.
7. Каждый избиратель посылает ЦИК: I, d.
8. ЦИК расшифровывает бюллетени. В конце выборов она публикует их результаты и для каждого варианта выбора – список соответствующих значений $E_k(I,v)$.
9. Если избиратель обнаруживает, что его бюллетень подсчитан неправильно, он протестует, посылая ЦИК: I, $E_k(I,v)$, d.

На шаге 4 – количество идентификационных номеров должно быть не менее n^2, где n – количество избирателей. Длина номера должна быть недостижима для поиска совпадений с опубликованными шифротекстами. В этом протоколе ЦИК не может изменить голоса избирателей, а вброс бюллетеней ограничен, так как список желающих принять участие в голосовании опубликован заранее. Превышение количества бюллетеней над количеством избирателей говорит о явной фальсификации.

Контрольные вопросы

1. Какие проблемы не решают приведенные схемы голосования?
2. Какие сведения должны быть опубликованы, чтобы максимально защитить избирателей от мошенничества со стороны ЦИК?

ТЕМА 20. ПРОГРАММНЫЕ И АППАРАТНЫЕ РЕАЛИЗАЦИИ

Криптографические библиотеки

В связи с ростом потребности в криптографической защите, среди пользователей, увеличивается количество программ, обеспечивающих защищенное хранение файлов или безопасное общение. Упрощается и разработка таких программ. Для различных языков программирования появляются библиотеки, реализующие большинство, необходимых в криптографии вычислений или алгоритмы шифрования целиком. Такие библиотеки могут распространяться как скомпилированными (например, в виде dll), так и в исходных кодах. Наибольший интерес представляют, конечно, вторые, так как всегда можно проверить реализованные в них алгоритмы на отсутствие закладок.

Open SSL

Библиотека OpenSSL (openssl.org) распространяется в виде исполняемых файлов и модулей, которые могут использоваться другими приложениями. Является бесплатной и доступна в исходных кодах. OpenSSL в основном используется совместно с web-сервером apache с модулем mod_ssl, для установления SSL соединений. Для языков C/C++ есть заголовочные файлы. Для php и perl – подключаемые модули. Из большинства языков можно обращаться к dll-библиотекам.

Библиотека OpenSSL позволяет следующее.

- Создавать и управлять открытыми и закрытыми ключами.
- Выполнять криптографические операции.

• Создавать сертификаты X.509, запросы подписей сертификатов, и запросы на добавление в список аннулированных.

• Шифровать соединения по протоколам SSL/TLS.

• Вычислять значения хеш-функций.

• Выполнять симметричное и ассиметричное шифрование и расшифрование.

• Выполнять шифрование и подпись почтовых сообщений по протоколу S/MIME.

Bouncy Castle

Bouncy Castle (bouncycastle.org) – это бесплатные криптографические библиотеки для языков java и C#. Библиотека построена в соответствии с JCA (Java Cryptography Architecture). Распространяются в виде классов и исходных кодов.

В библиотеке реализовано большинство известных алгоритмов шифрования (симметричного и ассиметричного) и цифровой подписи, в том числе алгоритмы ГОСТ. Могут использоваться и отдельные математические примитивы, например, можно сгенерировать параметры эллиптических кривых и далее использовать их в собственных реализациях. С помощью библиотеки может быть реализовано управление сертификатами X.509. Возможно шифрование в соответствие со стандартами OpenPGP.

Свободное криптографическое ПО

Обычно, использование программ с закрытым исходным кодом считается потенциально опасным, так как невозможно (сложнее) проверить отсутствие в программе уязвимостей или специально оставленных закладок или лазеек (backdoors). При использовании программ с открытым исходным кодом, такие проверки выполнить, возможно. Недостатком свободных систем обычно являются ограниченные возможности или слабый интерфейс. Часто реализуются только отдельные функции, не являющиеся единой системой.

GnuPG

GnuPG (GNU Privacy Guard, Страж приватности GNU) – это свободный некоммерческий аналог PGP. GnuPG основан на IETF-стандарте OpenPGP, в котором определены форматы сообщений, алгоритмы и параметры, используемые в PGP. Таким образом, GnuPG и PGP являются совместимыми. GnuPG распространяется по лицензии GPL, доступна в исходных кодах и бесплатна для использования в коммерческих и некоммерческих целях. PGP, являясь коммерческим продуктом, при этом

также доступна в исходных кодах, чтобы любой желающий мог проверить отсутствие в ней уязвимостей. GnuPG используется для выполнения следующих задач.

- Шифрования/Расшифрования.
- Подписывания/Проверки Подписи.
- Вычисления хеш-значения.
- Управления OpenPGP ключами.

GnuPG с помощью дополнительных модулей может быть интегрирована с различными почтовыми клиентами или может обеспечивать шифрование файлов.

TrueCrypt, DiskCryptor, FreeOTFE

Это программы, предназначенные для создания файловых контейнеров или шифрования разделов жесткого диска. Одной из первых программ этого класса была PGPdisk. В настоящее время существует множество коммерческих и свободных программ, позволяющих создавать шифрованные файловые контейнеры.

TrueCrypt (truecrypt.org) позволяет создать файл-контейнер, зашифрованный алгоритмами AES, Serpent, Twofish и их различными комбинациями, или зашифровать раздел жесткого диска. Файл-контейнер может быть подключен в системе как диск, соответственно храниться на нем могут не только документы, но и устанавливаться программы. Файл-контейнер может свободно копироваться и монтироваться с флеш-дисков. Ограничением является то, что для монтирования необходимы соответствующие права в системе. Создание виртуальных дисков может быть запрещено.

TrueCrypt-контейнер не содержит никаких сигнатур, с помощью которых можно было бы узнать что это именно файл программы TrueCrypt.

Возможно создание скрытых томов: файл-контейнер содержит в себе в этом случае два диска. При этом если известно, что файл является TrueCrypt контейнером, то о наличии второго диска узнать невозможно.

Также в TrueCrypt появилась возможность шифровать системные разделы. В этом случае устанавливается TrueCrypt загрузчик, который требует ввода пароля до начала загрузки операционной системы. При работе операционной системы все обращения к диску перехватываются, и выполняется расшифрование.

DiskCryptor (freed0m.org) имеет форматы совместимые с TrueCrypt и является его заменой с более свободной лицензией, допускающей свободную модификацию. Также возможна интеграция DiskCryptor со стандартными загрузчиками Linux (GRUB, LILO) при шифровании системных разделов.

FreeOTFE – Free On The Fly Encryption (freeotfe.org) также выполняет перечисленные действия, кроме возможности шифрования системных разделов.

Программные и аппаратные средства, лицензированные для использования в государственных учреждениях России

В России для обеспечения конфиденциальности информации, особенно в государственных учреждениях, могут использоваться только лицензированные в ФСБ программные и аппаратные средства. Их разработка может осуществляться только лицензированными компаниями. Уровень доверия к этим продуктам может определяться уровнем доверия к компаниям и к ФСБ, так как их исходный код не может быть проверен конечными пользователями.

КриптоПро

Компания КриптоПро (cryptopro.ru) с 2000 г. занимается разработкой средств криптографической защиты информации. Разработкой и внедрением криптографических средств, реализующих российские криптографические алгоритмы в соответствии со стандартом Microsoft Cryptographic Service Provider. Интеграцией стандартизированных национальных криптографических алгоритмов с продуктами Microsoft. Разработкой криптографических средств, поддерживающих Microsoft Public Key Infrastructure. Внедрением PKI.

Ниже приведены программные продукты компании.

- КриптоПро CSP – расширяет Windows Crypto API, обеспечивая возможность приложениям, работающим с Windows Crypto API использовать российские алгоритмы.
- КриптоПро JCP – реализует российские криптографические стандарты, в соответствии со спецификацией JCA (Java Cryptography Architecture).
- КриптоПро Sharpei – программный продукт, позволяющий использовать средство криптографической защиты информации. КриптоПро CSP на платформе Microsoft .Net Framework.
- КриптоПро УЦ – реализует функции удостоверяющего центра (центра сертификации).
- КриптоПро TSP – служба штампов времени. Штамп времени (timestamp) – это подписанный ЭЦП документ, которым Служба штампов времени удостоверяет, что в указанный момент времени ей было предоставлено значение хэш-функции от другого документа. Само значение хэш-функции также указывается в штампе.
- КриптоПро OCSP – служба проверки статуса сертификата в соответствии с RFC 2560 «Internet X.509 Public Key Infrastructure Online Certificate Status Protocol – OCSP».
- КриптоПро Revocation Provider – служба отзыва сетртификатов.
- КриптоПро ЭЦП – реализация ЭЦП документов. Для ЭЦП используется усовершенствованный стандарт RFC 5126 «CMS Advanced

Electronic Signatures (CAdES)», позволяющий решить проблемы базовой инфраструктуры открытых ключей. Для обеспечения юридической значимости, в подпись добавляется цепочка запросов и ответов OCSP, подтверждающих действительность ключа на момент подписания, на которые ставится метка времени. Такой формат подписи подтверждает то, что в момент подписания ключ не находился в CRL.

• КриптоПро TLS – реализует протокол TLS (SSL) на базе алгоритмов симметричного шифрования ГОСТ 28147-89, хеширования ГОСТ Р 34.11-94 и ЭЦП ГОСТ Р 34.10-2001 или ГОСТ Р 34.10-94.

• КриптоПро Winlogon – обеспечивает аутентификацию в Windows с использованием смарт-карт и USB-токенов. Для аутентификации в домене, используются открытые ключи и ЭЦП по ГОСТ Р 34.10-2001.

Программные продукты КриптоПро являются основным средством построения инфраструктуры открытых ключей в России. Из продуктов других компаний, предназначенных для построения инфраструктуры открытых ключей, можно отметить VCERT PKI.

Устройства криптографической защиты данных серии «Криптон»

Устройства «Криптон» гарантируют защиту информации, обрабатываемой на персональном компьютере и передаваемой по открытым каналам связи, выполнены в виде плат расширения ISA и PCI.

Ключи шифрования в устройство «Криптон» загружаются со смарт-карт и идентификаторов Touch Memory напрямую, минуя оперативную память и системную шину компьютера, что исключает возможность перехвата ключей, даже с зараженной системы.

Система Криптон позволяет:

• шифровать файлы и группы файлов, обеспечивая их конфиденциальность;

• осуществлять электронную цифровую подпись файлов, проверяя их целостность и авторство;

• создавать прозрачно шифруемые логические диски, максимально облегчая и упрощая работу пользователя с системой;

• формировать криптографически защищенные виртуальные сети, шифровать IP-трафик;

• создавать системы защиты информации от несанкционированного доступа и разграничения доступа к компьютеру.

Преимуществами аппаратных реализаций СКЗИ является возможность организовать надежную криптографическую защиту, не снижая производительности компьютера, так как все операции шифрования выполняются на аппаратном уровне. Шифрование дисков может выполняться абсолютно прозрачно для установленной операционной системы. А используемые ключи не могут быть перехвачены ни операционной

системой, ни вирусами. В программных реализациях ключи (пароли) могут быть перехвачены, при вводе с помощью сканеров клавиатуры (каждая клавиша дает различные ЭМ излучения) или программно (key-logerами или из буфера обмена) или при шифровании из оперативной памяти, если к ней может быть получен доступ.

Программные реализации с вводом ключей с смарт-карт решают только проблему перехвата ввода.

Контрольные вопросы

1. В чем заключаются преимущества и недостатки аппаратных реализаций криптосистем?
2. Какого рода лазеек следует опасаться в криптографическом программном и аппаратном обеспечении.

ТЕМА 21. КВАНТОВАЯ КРИПТОГРАФИЯ

Общие сведения

Квантовая криптография – метод защиты коммуникаций, основанный на определенных явлениях квантовой физики. В отличие от традиционной криптографии, которая использует математические методы, чтобы обеспечить секретность информации, квантовая криптография работает с физикой передачи информации. Процесс отправки и приема информации выполняется физическими средствами, при помощи фотонов в линиях волоконно-оптической связи, естественной среде или вакууме.

Технология квантовой криптографии опирается на свойства квантовых систем.

- Невозможно произвести измерение квантовой системы, не нарушив ее.
- Невозможно определить одновременно координату и состояние частицы со сколь угодно высокой точностью.
- Невозможно одновременно проверить поляризацию фотона в вертикально-горизонтальном и в диагональных направлениях.
- Невозможно дублировать квантовое состояние, пока оно не измерено.

Понятно, что из-за ограниченности возможностей по измерению использовать квантовые способы передачи данных в целом невыгодно, однако, используя квантовые явления, можно спроектировать и создать такую систему связи, которая всегда может обнаруживать подслушивание. Это обеспечивается тем, что попытка измерения взаимосвязанных параметров в квантовой системе вносит в нее нарушения, разрушая исходные сигналы, а значит, по уровню шума в канале пользователи могут распознать степень активности перехватчика.

История

Идея использования феномена квантовой корреляции для создания криптографических систем возникла в работах Стивена Уизнера в 1970-х гг. Однако его статья на эту тему была напечатана лишь в 1983 г. В 1989 г. Беннет и Брассар в Исследовательском центре IBM построили первую, работающую квантовую криптографическую систему. Она состояла из квантового канала, содержащего передатчик на одном конце и приёмник на другом, размещённые на оптической скамье длиной около метра в светонепроницаемом полутораметровом кожухе размером 0,5×0,5 м. Собственно квантовый канал представлял собой свободный воздушный канал длиной около 32 см. Макет управлялся от персонального компьютера, который содержал программное представление двух пользователей и злоумышленника. Передача сообщения посредством потока фотонов через воздушную среду на расстояние 32 см с компьютера на компьютер завершилась успешно. Основная проблема при увеличении расстояния между приемником и передатчиком – сохранение поляризации фотонов.

Протокол передачи ключа BB84

В 1984 г. Беннет и Брассард опубликовали статью, в которой описывался протокол квантового распространения ключа BB84. Носителями информации в протоколе BB84 являются фотоны, поляризованные под углами 0, 45, 90, 135°. В соответствии с законами квантовой физики, с помощью измерения можно различить лишь два ортогональных состояния: если известно, что фотон поляризован либо вертикально, либо горизонтально, то путем измерения, можно установить – как именно; то же самое можно утверждать относительно поляризации под углами 45 и 135°. Однако с достоверностью отличить вертикально поляризованный фотон от фотона, поляризованного под углом 45°, невозможно. При попытке измерения фотона, поляризованного под углом 45°, с помощью прямоугольного поляризатора с одинаковой вероятностью могут быть получены результаты 0 и 1. Эти особенности поведения квантовых объектов легли в основу протокола квантового распространения ключа [13].

Отправитель кодирует отправляемые данные, задавая определенные квантовые состояния, получатель регистрирует эти состояния. Затем получатель и отправитель совместно обсуждают результаты наблюдений по открытому каналу. Открытый канал связи не обязан быть конфиденциальным, но должен быть аутентифицированным. Чтобы обменяться ключом, Алиса и Боб выполняют следующие шаги:

1. Алиса посылает Бобу бит A_i, задавая определенные квантовые состояния – поляризацию фотонов в 0, 45, 90, 135°.

2. Боб располагает двумя анализаторами: один распознает вертикально-горизонтальную поляризацию, другой – диагональную. Для каждого фотона Боб случайно выбирает один из анализаторов и записывает тип анализатора и результат измерений B_i. Измерения могут быть выполнены в неправильном базисе либо фотоны могут быть просто не получены приемником. В полученном, «сыром», ключе$B_i = A_i$ с вероятностью P = 75 %. То есть он содержит ~25 % ошибок, для полученных фотонов.

3. По открытому каналу связи Боб сообщает Алисе, какие анализаторы использовались для полученных битов, но не сообщает, какие результаты были получены.

4. Алиса по открытому каналу связи сообщает Бобу, какие анализаторы он выбрал правильно.

5. Те фотоны, для которых Боб неверно выбрал анализатор, отбрасываются.

6. Для обнаружения перехвата Алиса и Боб выбирают случайный участок ключа и сравнивают его по открытому каналу связи. Если процент ошибок велик, то он может быть отнесен на счет злоумышленника, и процедура повторяется сначала. В качестве источника света может использоваться светоизлучающий диод или лазер. В качестве проводника используют либо пространство, либо оптические кабели.

Пример шифрования по протоколу BB84:

Будем обозначать:

- прямоугольный анализатор – «+»;
- диагональный анализатор – «×»;
- вертикальная поляризация – «|» кодирует 0;
- горизонтальная поляризация – «—» кодирует 1;
- поляризация под углом 45° – «⁄» кодирует 0;
- поляризация под углом 135° – «\» кодирует 1;

0	0	1	1	1	0	1	0	0	1	1	1	0	1	0	1	0	1	1	0
1	|	—	\		\	|	—	|		⁄	—	\		⁄	—	⁄	\		|	—	\		⁄
2	+	×	+	×	+	×	+	+	×	×	×	+	×	×	+	×	+	×	×
3	0	?	1	1	0	?	0	1	0	?	1	0	?	0	0	?	1	?	0
4	+		+	×	+		+	+	×		×	+		×	+		+		×
5	√		–	√	√		√	–	–		√	–		√	–		√		√
6				1			0							0					
7				√			√							√					
8	0				0						1						1		0

Рис. 44. Пример передачи ключа по протоколу BB84: 0 – шифруемый Алисой бит; 1 – положение поляризатора Алисы; 2 – анализатор Боба; 3 – полученные Бобом биты (? – потерянный фотон); 4 – передаваемая Алисе последовательность анализаторов; 5 – ответ Алисы с верными положениями анализаторов; 6 – биты, передаваемые Алисе для проверки; 7 – результат проверки; 8 – полученная ключевая последовательность

Если бы злоумышленник производил перехват информации при помощи оборудования, подобного оборудованию Боба, то примерно в 50 % случаев он выберет неверный анализатор, не сможет определить состояние полученного им фотона, и отправит фотон Бобу в состоянии, выбранном наугад. Таким образом, примерно в 25 % случаев результаты измерений Боба могут отличаться от результатов Алисы. Это довольно заметно и быстро обнаруживаемо на стадии проверки.

Так как возможна не только потеря битов, но и неправильное измерение с верным анализатором, то необходимо использовать также и коды с исправлением ошибок. Либо после обмена проверять ключ, с помощью одного из протоколов авторизации. И если ключ неверен, то передача должна быть повторена сначала.

Протокол B92

Протокол B92 (Bennett, 1992) также может использоваться для распределения ключей. В отличие от BB4, где получатель может при получении с вероятностью 0,75 получить состояние каждого фотона, в этом протоколе получатель с вероятностью близкой к 1 может получить состояние 25 % фотонов.

Для представления нулей и единиц в этом протоколе используются фотоны, поляризованные в двух различных направлениях. Угол между направлениями поляризации этих фильтров равен 45°. Например, 0 и 45°.

Получатель использует фильтры с углами 90 и 135°для приема фотонов. Если различие в поляризации фотона и фильтра составляет 90°, фотон не проходит через фильтр. При различии в поляризации, составляющем 45° вероятность прохождения фотона через фильтр составляет 0,5.

Пусть поляризация излученного фотона с углом 0° соответствует биту 0, с углом 45° – биту 1. Прохождение фотоном фильтра с углом 90° соответствует биту 1, непрохождение условно обозначаем битом 0. Прохождение фильтра с углом 135° соответствует биту 0, непрохождение условно обозначаем битом 1.

Алиса передает информацию через два фильтра с ориентацией на 0 и 45°, представляющие нули и единицы. Направление поляризации выбирается случайно. Боб может применять фильтры с ориентацией 90 и 135°.

1. Алиса посылает Бобу последовательность случайно сориентированных фотонов.

2. Для определения поляризации Боб пропускает фотоны, через один из двух фильтров. Допустим, что через один из фильтров (например, 135°) фотон не проходит. Боб не знает, что послано ему: 1, соответствующая фотону, который не проходит, или 0, соответствующий фотону, который не проходит с вероятностью 0.5. Если же фотон проходит через

фильтр, Боб уверен, что принят фотон, соответствующий 0. Если положение поляризатора позволяет однозначно определить поляризацию фотона, очередной бит ключа кодируется 0 или 1 в соответствии с примененным фильтром.

3. Боб по открытому каналу связи сообщает Алисе номера битов, которые он принял удачно.

Определение бита отправителя на шаге 3, может быть проиллюстрировано табл. 15 и 16.

Таблица 15

Вероятности правильного определения

α_A	α_B	b_A	P	b_B
0°	90°	0	1	0
45°	90°	1	0,5	0 или 1
0°	135°	0	0,5	0 или 1
45°	135°	1	1	1

Таблица 16

Знание получателя о бите отправителя

b_B	α_B	Боб знает о b_A
0	90°	0 (P = 2/3) или 1 (P = 1/3)
1	90°	1
0	135°	0
1	135°	0 (P = 1/3) или 1 (P = 2/3)

Если получен бит 1, при ориентации фильтра в 90°, то можно сказать что был отправлен бит 1, если получен бит 0, при ориентации фильтра в 135°, то можно сказать что был отправлен бит 0. Иначе определенно узнать значение бита отправителя нельзя.

Проблемы практических реализаций

При создании практических криптосистем, основанных на квантовом распространении ключа, приходится сталкиваться со следующими проблемами.

• Низкая скорость передачи – скорость передачи по каналам большой длины ~кбит, по коротким каналам ~ 10–100 кбит.

• Небольшие расстояния – до 100 км со скоростями порядка кбит.

• Интенсивность импульсов квантов – обычно фотоны излучаются пучком, что позволяет злоумышленнику отделить часть фотонов и проанализировать их состояние. Излучение одиночных фотонов заданной поляризации возможно только с некоторой вероятностью.

Контрольные вопросы

1. За счет чего достигается невозможность перехвата сообщений?

2. Возможна ли в приведенных протоколах атака «человек посередине»?

3. Какие условия должны быть выполнены, чтобы злоумышленник мог свободно перехватывать сообщения. Каков процент правильных бит будет для приведенных протоколов?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ГОСТ 28147-89. Системы обработки информации. защита криптографическая. Алгоритм криптографического преобразования

2. ГОСТ Р 34.11-94. Информационная технология. Криптографическая защита информации. Функция хэширования.

3. ГОСТ Р 34.10-2001 Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи

4. Rivest, R. Request for Comments: 1320. The MD4 Message-Digest Algorithm [электронный ресурс] / R. Rivest. – MIT Laboratory for Computer Science, 1992. – www.ietf.org

5. Krawczyk, H. Request for Comments: 2104. HMAC: Keyed-Hashing for Message Authentication [электронный ресурс] / H. Krawczyk, M. Bellare, R. Canetti. – Network Working Group, 1997. – www.ietf.org

6. Krovetz, T. Request for Comments: 4418. UMAC: Message Authentication Code using Universal Hashing [электронный ресурс] / T. Krovetz. – Network Working Group, 2006. – www.ietf.org

7. Federal Information Processing Standards Publication 197. Announcing the Advanced encryption standard (AES) [электронный ресурс] / NIST, 2006 – 51 с. – www.csrc.nist.gov.

8. Смарт, Н. Криптография / Н. Смарт. – М. : Техносфера, 2005. – 528 с.

9. Фергюсон, Н. Практическая Криптография / Н. Фергюсон, Б. Шнайер. – М. : Издательский дом «Вильямс», 2005. – 416 с.

10. Шнайер, Б. Секреты и ложь. Безопасность данных в цифровом мире / Б. Шнайер. – СПб. : Питер, 2003. – 370 с.

11. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. – М. : ТРИУМФ, 2002. – 816 с.

12. Mao, Wenbo. Modern Cryptography: Theory and Practice / Wenbo Mao. – Prentice Hall, 2004. – 755 с.

13. Брассар, Ж. Современная криптология / Ж. Брассар – М. : ПОЛИМЕД, 1999. – 176 с.

14. Анисимов, В. В. Криптография : метод. указания / В. В. Анисимов. – Хабаровск : Изд-во ДВГУПС, 2004. – 32 с.

15. Криптографическая защита информации : учеб. пособие / А. В. Яковлев [и др.]. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2006. – 140 с.

16. Баричев, С. Г. Основы современной криптографии : учеб. пособие / С. Г. Баричев, Р. Е. Серов. – М. : Горячая Линия – Телеком, 2002. – 152 с.

17. Коробейников, А. Г. Математические основы криптологии : учеб. пособие / Ю. А. Гатчин, А. Г. Коробейников. – СПб. : СПбГУ ИТМО, 2004. – 106 с.

18. Василенко, О. Н. Теоретико-числовые алгоритмы в криптографии / О. Н. Василенко. – М. : МЦНМО, 2003. – 328 с.

19. Алгоритмические основы эллиптической криптографии / А. А. Болотов [и др.]. – М. : Изд-во РГСУ, 2004. – 499 с.

20. Колбиц, Н. Курс теории чисел и криптографии / Н. Колбиц. – М. : ТВП, 2001. – 254 с.

21. Петров, А. А. Компьютерная безопасность. Криптографические методы защиты / А. А. Петров. – М. : ДМК, 2000. – 448 с.

22. Погорелов, Б. А. Словарь криптографических терминов / Б. А. Погорелов, В. Н Сачков. – М. : МЦНМО, 2006. – 88 с.

23. Введение в криптографию / под ред. В. В. Ященко. – СПб. : Питер, 2001. – 288 с.

24. Чернова, Н. И. Теория вероятностей : учеб. пособие / Н. И. Чернова. – Новосибирск : Новосиб. гос. ун-т, 2007. – 160 с.

25. Коутинхо, С. Теория чисел. Алгоритм RSA / С. Коутинхо. – М. : Постмаркет, 2001. – 328 с.

26. Масленников, М. Е. Криптография и свобода [электронный ресурс] / М. Е. Масленников. – mikhailmasl.livejournal.com

27. Лапонина, О. Р. Криптографические основы безопасности [электронный ресурс] / О. Р. Лапонина. – www.intuit.ru

28. Шефановский, Д. Б. ГОСТ Р 34.11-94. Функция хэширования. Краткий анализ [электронный ресурс] / Д. Б. Шефановский. – М. : Информзащита, 2001. – 9 с.

29. Оценки времени исполнения. Символ O() [электронный ресурс]. – www.algolist.ru

30. Зенин, О. Режимы шифрования [электронный ресурс] / О. Зенин – 2003. – citforum.ru

31. Винокуров, А. Как устроен блочный шифр? [электронный ресурс] / А. Винокуров, 1995. – enlight.ru/crypto/articles/vinokurov/__blcyph.htm

32. Anderson, R. J. On Fibonacci Keystream Generators [электронный ресурс] / R. J. Anderson. – www.cl.cam.ac.uk/~rja14/

33. Schoof, R. Counting points on elliptic curves over finite fields [электронный ресурс] / R. Schoof // Journal de Theorie des Nombres de Bordeaux, 1995. – www.numdam.org/item?id=JTNB_1995__7_1_219_0.

34. Kohl, J. T. The Evolution of the Kerberos Authentication Service [электронный ресурс] / J. T. Kohl, B. C. Neuman, T. Y. Ts'o. – IEEE Computer Society Press, Distributed Open Systems, 1994. – www.isi.edu/div7/publication_files/rs-94-412.pdf

Отпечатано в мини-типографии
ГАОУ СПО РБ «БРМТИТ»
2012 год. Тираж 20 экз.
671700, Республика Бурятия,
г. Северобайкальск, пр. 60 лет СССР, д. 40 А,
тел. факс (830130) 2 – 08 – 51