Прикладная криптография

2-е издание

Протоколы, алгоритмы и исходные тексты на языке С

Брюс Шнайер

Предисловие
Уитфилд Диффи

История литературы по криптографии довольно любопытна . Секретность, конечно же, всегда играла важную роль, но до Первой мировой войны о важных разработках время от времени сообщалось в печати и криптогр афия развивалась также, как и другие специализированные дисциплины. В 1918 году в виде научного отчета ч астной Лаборатории Ривербэнк вышла в свет монография Вильяма Ф. Фридмана *Показатель совпадений и его применения в криптографии* (*Index of Coincidence and Its Applications in Cryptography*) [577], одна из определяющих работ 20-го столетия. И это несмотря на военный заказ, по которому была сделана эта работа. В том же году Эдвард Х. Хеберн из Окленда, Калифорния, получил первый патент [710] на роторную машину, устройство, на котором основывалась военная криптография в течение почти 50 лет.

После Первой мировой войны, однако, все изменилось. Организации армии и флота Соединенных Штатов, полностью засекретив свои работы, добились фундаментальных успехов в криптографии. В течение 30 -х и 40-х годов в открытой литературе по данному предмету появлялись только отдельные основные работы и моногр афии, но чем дальше, тем меньше они соответствовали реальному положению дел. К концу войны переход по лностью завершился. Открытая литература умерла за исключением одного заметного исключения, работы Клода Шэннона "The Communication Theory of Secrecy systems" (*Теория связи между секретными системами*), напечатанной в 1949 году в *Bell System Technical Journal* [1432]. Эта статья, как и работа Фридмана в 1918 году, явилась результатом исследований Шэннона во время войны. После окончания Второй мировой войны она была рассекречена, возможно по ошибке.

С 1949 по 1967 литература по криптографии была бессодержательной. В 1967 году она пополнилась работой другого типа, историей Дэвида Кана *Дешифровщики* (*The Codebreakers*) [794]. В этой книге не было новых идей, но она содержала достаточно полную историю предмета, включая упоминание о некоторых вещах, все еще засекреченных правительством. Значение *Дешифровщиков* заключалось не только в значительном охвате предмета, книга имела заметный коммерческий успех и познакомила с криптографией тысячи людей, раньше и не задумывавшихся о ее существовании. Тоненьким ручейком начали появляться новые работы по криптогр афии.

Почти в то же время Хорста Фейстела, ранее работавшего над прибором "свой/чужой" для ВВС, на всю дальнейшую жизнь охватила страсть к криптографии, и он перешел в Уотсоновскую Лабораторию фирмы IBM, расположенную в Йорктаун Хайтс, Нью-Йорк. Там он начал разработку того, что затем стало стандартом DES (U.S. Data Encryption Standard, Стандарт шифрования данных Соединенных Штатов). В начале 70-х годов IBM опубликовала ряд технических отчетов по криптографии, выполненных Фейстелом и его коллегами [1482, 1484, 552].

Таково было положение, когда в конце 1972 года я начал работать в этой области. Литература по крипт ографии обильной не была, но в ней можно было найти ряд сверкающих самородков.

В криптографической науке есть особенность, отсутствующая в обычных академических дисциплинах: нео бходимость взаимодействия криптографии и криптоанализа. Причиной этого является отсутствие требований к передаче реальной информации, следовательно, нетрудно предложить систему, которая кажется непогрешимой. Многие академические разработки настолько сложны, что будущий криптоаналитик не знает с чего начать. О бнаружить дыры в этих проектах намного сложнее, чем разработать их. В результате невозможно соревнование, являющееся одним из сильнейших мотивов в академических исследований.

Когда Мартин Хеллман и я в 1975 году предложили криптографию с открытыми ключами [496], одним из косвенных аспектов нашего предложения было появление проблемы, решение которой не кажется простым. Теперь честолюбивый проектировщик мог создать что-то - вполне разумную криптосистему, решающую более обширные задачи, чем простое превращение значимого текста в чепуху. В результате значительно возросло число людей, занимающихся криптографией, число проводимых встреч и число опубликованных книг и статей.

В речи по поводу присуждения мне совместно с Мартином Хеллманом премии Дональда Е. Финка (присуждаемой за лучшую пояснительную статью в журнале IEEE) я сказал, что, написав "Privacy and Authentication" ("Секретность и удостоверение подлинности"), я получил опыт, который необычен даже для выдающи хся ученых, получивших премии IEEE. Я написал статью, которую я хотел бы изучить, когда я впервые серьезно заинтересовался криптографией, и которую не смог найти. Если бы я сегодня отправился в Стэнфордскую би блиотеку и собрал бы современные работы по криптографии, я, возможно, получил бы представление о предмете гораздо раньше. Но осенью 1972 года были доступны только несколько классических работ и ряд туманных технических отчетов.

У сегодняшнего исследователя нет такой проблемы. Сегодня основная сложность состоит в выборе, с чего

начать среди тысяч статей и десятков книг. А сегодняшние программисты и инженеры, которые просто хотят использовать криптографию? К каким источникам им обращаться? До сих пор необходимо было проводить долгие часы, выискивая научную литературу и изучая ее, прежде чем удавалось начать разработку криптогр афических приложений, так гладко описанных в популярных статьях.

Именно этот промежуток и призвана заполнить *Прикладная криптография* Брюса Шнайера. Начав с целей засекречивания передачи данных и элементарных примеров программ для достижения этих целей, Шнайер ра зворачивает перед нами панораму результатов 20 лет открытых исследований. Содержание книги полностью определяется ее названием, вы найдете в ней описание различных приложений, от засекречивания телефонного разговора до электронных денег и криптографического обеспечения выборов.

Не удовлетворенный простым изложением алгоритмов и описанием кода, Шнайер включил в книгу обсу ждение различных мировых организаций, связанных с разработкой и применением криптографических средств, от Международной ассоциации криптологических исследований до NSA (National Security Agency, Агентство национальной безопасности).

Когда на рубеже 70-х и 80-х годов возрос общественный интерес к криптографии, NSA, официальный крип тографичекий орган США, предприняло ряд попыток подавить этот интерес. Первой такой попыткой было письмо старого сотрудника NSA, по видимому действовавшего по своему усмотрению. Письмо было послано в IEEE и предупреждало, что публикация материалов по криптографии является нарушением Правил междун ародной продажи оружия (International Traffic in Arms Regulations, ITAR). Эта точка зрения, как оказалось, не поддерживаемая самими правилами, в явном виде содержащими льготы для публикуемых материалов, создала неожиданную рекламу использованию криптографии и Семинару по теории информации 1977 года.

Более серьезная попытка была предпринята в 1980 году, когда NSA финансировало изучение вопроса Ам ериканским советом по образованию с целью убедить Конгресс узаконить контроль над публикациями в области криптографии. Результаты, оказавшиеся далекими от ожиданий NSA, привели к программе добровольного р ецензирования работ по криптографии. От исследователей потребовали перед публикацией запрашивать мнение NSA, не принесет ли раскрытие результатов исследований вред национальным интересам.

К середине 80-х годов основным объектом внимания стала не теория, а практика криптографии. Сущес т-вующие законы дают NSA право с помощью Госдепартамента регулировать экспорт криптографического об орудования. Так как бизнес все больше и больше принимает международный характер и американская часть м ирового рынка уменьшается, возрастает желание использовать единый продукт и для внутреннего, и для внешн его рынка. Такие продукты являются субъектами контроля над экспортом, и поэтому NSA получило возможность контролировать не только экспортируемые криптографические продукты, но и продаваемые в Соединенных Штатах.

В то время, когда писались эти строки, возникло новое препятствие для общественного использования кри птографии. Правительство дополнило широко опубликованный и используемый алгоритм DES засекреченным алгоритмом, реализованным в микросхемах памяти, независящей от времени. Эти микросхемы будут содержать кодифицированный механизм правительственного контроля. Отрицательные аспекты такой програ м-мы-троянского коня простираются от потенциально губительного раскрытия тайны лич ности до высокой сто имости аппаратной модернизации продуктов, ранее реализованных программно. Таким образом, предлагаемое нововведение не вызвало энтузиазма и подверглось широкой критике, особенно со стороны независимых кри птографов. Ряд людей, однако, видят свое будущее в программировании, а не в политике и удваивают свои ус илия, стремясь представить миру мощные средства криптографии.

Значительное отступление от возможности того, что закон о контроле над экспортом отменит Первую п оправку[1], казалось было сделан в 1980 году, когда в опубликованные в *Federal Register* исправления ITAR вошло следующее положение: "...положение было добавлено с целью показать, что регулирование экспорта технич еских данных не приведет к конфликту с правами личности, определяемыми Первой поправкой". Но то, что ко нфликт между Первой поправкой и законами о контроле над экспортом не разрешен окончательно, должно быть очевидно из заявлений, сделанных на конференции, проводимой RSA Data Security. Представитель NSA из отдела контроля над экспортом выразил мнение, что люди, публикующие криптографические программы, нах одятся " в серой зоне" по отношению к закону. Если это так, то именно эту "серую зону" немного осветило первое издание этой книги. Экспорт приложений для этой книги был разрешен с подтверждением того, что опублик ованные материалы не попадают под юрисдикцию Совета по контролю над вооружением. Однако, экспортир овать опубликованные программы на диске было запрещено.

Изменение стратегии NSA от попыток контролировать криптографические исследования к усилению регул ирования в области разработки и развертывания криптографических продуктов по видимому обусловлено осо знанием того, что все величайшие криптографические работы не защитили ни одного бита информации. Будучи

---

[1] К конституции США

поставлен в шкаф, этот том не сделает ничего нового по сравнению с предшествующими книгами и работами, но использование его содержания на рабочей станции, где пишется криптографический код, может привести к иному результату.

Уитфилд Диффи

Маунтэйн Вью, Калифорния.

Введение

Криптография бывает двух типов: криптография, которая помешает читать ваши файлы вашей младшей с е-стре, и криптография, которая помешает читать ваши файлы дядям из правительства. Эта книга о втором типе криптографии.

Если я беру письмо, кладу его в сейф где-нибудь в Нью-Йорке, затем велю Вам прочитать это письмо, то это не безопасность. Это непонятно что. С другой стороны, если я беру письмо и кладу его в сейф, затем передаю этот сейф Вам вместе с детальным описанием, передаю также сотню подобных сейфов с их комбинациями, чт обы Вы и лучшие "медвежатники" мира могли изучить систему замков, а вы все равно не сможете открыть сейф и прочитать письмо - вот это и есть безопасность.

В течение многих лет этот тип криптографии использовался исключительно в военных целях. Агентство н ациональной безопасности Соединенных Штатов Америки (National Security Agency, NSA) и его аналоги в быв-шем Советском Союзе, Англии, Франции, Израиле и прочих странах тратили миллиарды долларов на очень серьезную игру в обеспечение безопасности собственных линий связи, одновременно пытаясь взломать все о с-тальные. Отдельные личности, обладающие значительно меньшими средствами и опытом, были беспомощны защитить свои секреты от правительств.

В течение последних 20 лет значительно вырос объем открытых академических исследований. Пока обычные граждане использовали классическую криптографию, со времен Второй мировой войны компьютерная крипт о-графия во всем мире применялась исключительно в военной области. Сегодня искусство компьютерной крипт о-графии вырвалось из стен военных ведомств. Непрофессионалы получили возможность средства, позволяющие им обезопасить себя от могущественнейших противников, средства, обеспечивающие защиту от военных в е-домств.

А нужна ли обычному человеку такая криптография? Да. Люди могут планировать политическую кампанию, обсуждать налоги, вести незаконные действия. Они могут разрабатывать новые изделия, обсуждать рыночную политику или планировать захват конкурирующей фирмы. Они могут жить в стране, которая не соблюдает з а-прета на вторжение в личную жизнь своих граждан. Они могут делать что-либо, что не кажется им незаконным, хотя таковым и является. По многим причинам данные и линии связи должны быть личными, тайными и з а-крытыми от постороннего доступа.

Эта книга выходит в свет в беспокойное время. В 1994 году администрация Клинтона приняла Стандарт у с-ловного шифрования (Escrowed Encryption Standard), включая микросхему Clipper и плату Fortezza, и превра-тило Билль о Цифровой телефонии в закон. Эти инициативы пытаются увеличить возможности правительства проводить электронный контроль.

Вступают в силу некоторые опаснейшие домыслы Оруэлла: правительство получает право прослушивать личные переговоры, а с человеком, пытающимся скрыть свои секреты от правительства, может что-нибудь сл у-читься. Законодательство всегда разрешало слежку по решению суда, но впервые люди сами должны предпр и-нимать какие-то шаги, чтобы *сделаться доступными* для слежки. Эти инициативы не просто предложения пр а-вительства в некой туманной сфере, это упреждающая и односторонняя попытка присвоить прежде принадл е-жащие людям права.

Законопроекты о микросхеме Clipper и Цифровой телефонии не способствуют сохранению тайны, но беспо ч-венно заставляют людей считать, что правительство уважает их тайны. Те же самые власти, которые незаконно записывали телефоны Мартина Лютера Кинга, могут легко прослушать телефон, защищенный микросхемой Clipper. В недавнем прошлом полицейские власти на местах были привлечены к гражданской или уголовной ответственности за незаконное прослушивание во многих судах - в Мэриленде, Коннектикуте, Вермонте, Джорджии, Миссури и Неваде. Идея развернуть технологию, которая может привести к появлению полице й-ского государства - это плохая идея.

Дело в том, что недостаточно защитить себя законами, нам нужно защитить себя математикой. Шифров а-ние имеет слишком большое значение, чтобы оставить ее использование только правительствам .

Эта книга снабдит Вас инструментарием, позволяющим защитить ваши тайны. Передача криптографич е-ских продуктов может быть объявлена незаконной, передача информации - никогда .

Как читать эту книгу

Я написал *Прикладную криптографию* как живое введение в криптографию и как всеобъемлющий спр а-вочник. Я пытался сочетать читаемость текста с жертвенной точностью, но эта книга писалась не как мат е-матическая работа. Хотя я не искажал информацию умышленно, торопясь, я опускал теорию . Для интере-сующихся теоретическими выкладками приведены обширные ссылки на академическую литературу .

Глава 1 представляет собой введение в криптографию, описывает множество терминов, в ней кратко ра с-

сматривается докомпьютерная криптография.

Главы со 2 по 6 (Часть I) описывают криптографические протоколы - что люди могут сделать с помощью криптографии - от простых (передача шифрованных сообщений от одного человека другому) до сложных (щелканье монетой по телефону) и тайных (секретное и анонимное обращение электронных денег). Некоторые из этих протоколов очевидны, другие - удивительны. Множество людей и не представляет многие из проблем, которые может решить криптография.

Главы с 7 по 10 (Часть II) содержат обсуждение методов криптографии. Все эти четыре главы важны для самых распространенных применений криптографии. В главах 7 и 8 рассказывается о ключах: какова должна быть длина безопасного ключа, как генерировать, хранить и распределять ключи, и т.д. Управление ключами представляет собой труднейшую часть криптографии и часто является ахиллесовой пятой систем, безопасных во всем остальном. В главе 9 рассматриваются различные способы использования криптографических алгоритмов, а глава 10 описывает особенности и цели использования этих алгоритмов - как их выбирать, реализовывать и применять.

Главы с 11 по 23 (Часть III) описывают эти алгоритмы. Глава 11 представляет собой математическую базу и является обязательной только, если вы интересуетесь алгоритмами с открытыми ключами. Если вы собираетесь использовать DES (или что-то похожее), ее можно пропустить. В главе 12 обсуждается алгоритм DES, его история, безопасность и разновидности. В главах 13, 14 и 15 рассказывается о других блочных алгоритмах. Если вам нужно что-то более надежное чем DES, сразу переходите к разделам о IDEA и тройном DES. При желании узнать о группе алгоритмов, некоторые из которых могут быть безопаснее DES, прочитайте всю главу. В главах 16 и 17 обсуждаются потоковые алгоритмы. В главе 18 подробно рассматриваются однонаправленные хэш-функции, среди которых самыми являются MDS и SHA, хотя я останавливаюсь и на многих других. В главе 19 рассматриваются алгоритмы шифрования с открытым ключом, а в главе 20 - алгоритмы цифровой подписи с открытым ключом. В главе 21 обсуждаются алгоритмы идентификации с открытым ключом, а в главе 22 - алгоритмы обмена с открытым ключом. Самыми важными являются алгоритмы RSA, DSA, Фиат-Шамира (Fiat-Shamir) и Диффи-Хелмана (Diffie-Hellman). Глава 23 содержит ряд эзотерических алгоритмов и протоколов с открытым ключом, математика в этой главе достаточно сложна, так что пристегните ремни.

Главы 24 и 25 (Часть IV) переносят вас в реальный мир криптографии. В главе 24 обсуждаются некоторые современные применения алгоритмов и протоколов, в то время как глава 25 касается некоторых политических аспектов криптографии. Несомненно, эти главы не являются всеохватывающими.

В книгу также включены исходные коды 10 алгоритмов, рассмотренных в Части III. Я не смог включить весь код, который хотел, из-за его большого объема, кроме того, криптографические коды в любом случае нельзя экспортировать. (Любопытно, что Госдепартамент разрешил экспортировать первое издание этой книги с исходным кодом, но не разрешил экспортировать компьютерный диск с теми же исходными кодами. Смотри рисунок.) Соответствующий набор дисков с исходным кодом содержит существенно больше исходных кодов, чем я смог включить в эту книгу, возможно, это самая большая подборка криптографических исходных кодов, появившаяся за пределами военных ведомств. Сейчас я могу переслать эти диски с исходным кодом только гражданам США и Канады, живущим в этих странах, но, возможно, когда-нибудь все изменится. Если вы собираетесь использовать или попробовать эти алгоритмы, добудьте диск. Подробности на последней странице книги..

К недостаткам этой книги относится то, что из-за ее энциклопедической природы пострадала читаемость книги. Я хотел написать единый справочник для тех, кто мог встретиться с каким-либо алгоритмом в академической литературе или при использовании какого-то продукта, и заранее извиняюсь перед теми, кто разыскивает учебное пособие. Впервые все множество сделанного в криптографии собрано под одной обложкой. Несмотря на это, соображения объема заставили меня оставить многое за пределами этой книги, я включил те темы, которые мне показались важными, практическими или интересными. Если я не мог полностью охватить тему, я приводил ссылки на соответствующие работы и статьи.

Я сделал все, что мог, пытаясь выловить и исправить все ошибки в книге, но многие люди уверяли меня, что это все равно невозможно. Конечно, во втором издании ошибок меньше, чем в первом. Перечень ошибок можно получить у меня, он также периодически рассылается в телеконференции Usenet sci.crypt. Если кто-нибудь из читателей обнаружит ошибку, пожалуйста, пусть сообщит мне об этом. Каждому, кто первый обнаружит данную ошибку в книге, я бесплатно пошлю диск с исходным кодом.

Благодарности

Перечень людей, приложивших руку к созданию этой книги, может показаться бесконечным, но все они достойны упоминания. Мне хотелось бы поблагодарить Дона Альвареса (Don Alvarez), Росса Андерсона (Ross Anderson), Дэйва Бейленсона (Dave Balenson), Карла Бармса (Karl Barms), Стива Белловина (Steve Bellovin), Дэна Бернстайна (Dan Bernstein), Эли Байем (Ell Biham), Джоан Бояр (Joan Boyar), Карен Купер (Karen Cooper), Вита Диффи (Whit Diffie), Джоан Фейгенбаум (Joan Feigenbaum), Фила Кана (Phil Karn), Нила Коблица (Neal Koblitz), Ксуейа Лай (Xuejia Lai), Тома Леранта (Tom Leranth), Майка Марковица (Mike Markowitz), Ральфа Меркла (Ralph Merkle), Билла Паттена (Bill Patten), Питера Пирсона (Peter Pearson), Чарльза Пфлегера (Charles

Pfleeger), Кена Пиццини (Ken Pizzini), Барта Пренела (Bart Preneel), Марка Риордана (Mark Riordan), Йоахима Шурмана (Joachim Schurman) и Марка Шварца (Marc Schwartz) за чтение и редактирование всего первого издания или его частей; Марка Воклера (Marc Vauclair) за перевод первого издания на французский; Эйба Абрахама (Abe Abraham), Росса Андерсона (Ross Anderson), Дэйва Бенисара (Dave Banisar), Стива Белловина (Steve Bellovin), Эли Байем (Ell Biham), Мэтта Бишопа (Matt Bishop), Мэтта Блэйза (Matt Blaze), Гэри Картера (Gary Carter), Жана Комениша (Jan Comenisch), Клода Крепо (Claude Crepeau), Джоан Дэймон (Joan Daemon), Хорхе Давила (Jorge Davila), Эда Доусона (Ed Dawson), Вита Диффи (Whit Diffie), Карла Эллисона (Carl Ellison), Джоан Фейгенбаум (Joan Feigenbaum), Нильса Фергюсона (Niels Ferguson), Матта Франклина (Matt Franklin), Розарио Сеннаро (Rosario Cennaro), Дитера Колмана (Dieter Collmann), Марка Горески (Mark Goresky), Ричарда Грэйвмана (Richard Graveman), Стюарта Хабера (Stuart Haber), Джингмана Хе (Jingman He), Боба Хэйга (Bob Hague), Кеннета Айверсона (Kenneth Iversen), Маркуса Джекобсона (Markus Jakobsson), Берта Калиски (Burt Kaliski), Фила Кана (Phil Karn), Джона Келси (John Kelsey), Джона Кеннеди (John Kennedy), Ларса Кнудсена (Lars Knudsen), Пола Кочера (Paul Kocher), Джона Лэдвига (John Ladwig), Ксуейа Лай (Xuejia Lai), Аджена Ленстры (Arjen Lenstra), Пола Лейланда (Paul Leyland), Майка Марковица (Mike Markowitz), Джима Мэсси (Jim Massey), Брюса МакНейра (Bruce McNair), Вильяма Хью Мюррея (William Hugh Murray), Роджера Нидхэма (Roger Needham), Клифа Неймана (Clif Neuman), Кейсу Найберг (Kaisa Nyberg), Люка О'Коннора (Luke O'Connor), Питера Пирсона (Peter Pearson), Рене Перальта (Rene Peralta), Барта Пренела (Bart Preneel), Израиля Радай (Yisrael Radai), Мэтта Робшоу (Matt Robshaw), Майкла Роу (Michael Roe), Фила Рогуэя (Phil Rogaway), Эви Рубина (Avi Rubin), Пола Рубина (Paul Rubin), Селвина Рассела (Selwyn Russell), Казуе Сако (Kazue Sako), Махмуда Салмасизадеха (Mahmoud Salmasizadeh), Маркуса Стадлера (Markus Stadler), Дмитрия Титова (Dmitry Titov), Джимми Аптона (Jimmy Upton), Марка Воклера (Marc Vauclair), Сержа Воденея (Serge Vaudenay), Гидеона Ювала (Gideon Yuval), Глена Зорна (Glen Zorn) и многих безымянных правительственных служащих за чтение и редактирование всего второго издания или его частей; Лори Брауна (Lawrie Brown), Лизу Кэндл (Leisa Candle), Джоан Дэймон (Joan Daemon), Питера Гутмана (Peter Gutmann), Алана Инсли (Alan Insley), Криса Джонстона (Chris Johnston), Джона Келси (John Kelsey), Ксуейа Лай (Xuejia Lai), Билла Лейнингера (Bill Leininger), Майка Марковица (Mike Markowitz), Ричарда Аутбриджа (Richard Outerbridge), Питера Пирсона (Peter Pearson), Кена Пиццини (Ken Pizzini), Кэлма Пламба (Calm Plumb), RSA Data Security, Inc., Майкла Роу (Michael Roe), Майкла Вуда (Michael Wood) и Фила Циммермана (Phil Zimmermann) за предоставленные исходные коды; Пола МакНерланда (Paul MacNerland) за создание рисунков к первому издания; Карен Купер (Karen Cooper) за редактирование второго издания; Бота Фридмана (Both Friedman) за сверку второго издания; Кэрол Кеннеди (Кэрол Kennedy) за работу над предметным указателем для второго издания; читателей sci.crypt и почтового списка Cypherpunks за комментирование идей, ответы на вопросы и поиск ошибок первого издания; Рэнди Сюсс (Randy Seuss) за предоставление доступа к Internet; Джеффа Дантермана (Jeff Duntemann) и Джона Эриксона (Jon Erickson) за то, что помогли мне начать; семью Insley (в произвольном порядке) за стимуляцию, воодушевление, поддержку, беседы, дружбу и обеды; и AT&T Bell Labs, зажегшей меня и сделавшей возможным все это. Все эти люди помогли создать гораздо лучшую книгу, чем я бы смог создать в одиночку.

Брюс Шнайер
Оак Парк, Иллинойс
schneier@counterpane.com

Об авторе

БРЮС ШНАЙЕР - президент Counterpane Systems, Оак Парк, Иллинойс, фирма-консультант, специализирующаяся в криптографии и компьютерной безопасности. Брюс также написал *E-Mail Security*, John Wiley & Sons, 1995, (*Безопасность электронной почты*) и *Protect Your Macintosh*, Peachpit Press, 1994, (*Защити свой Макинтош*). Он является автором дюжин статей по криптографии в основных журналах. Он также соредактор *Dr. Dobb's Journal* (*Журнал доктора Добба*), где он редактирует колонку "Аллея алгоритмов", и соредактор *Computer and Communications Security Reviews* (Обзор безопасности компьютеров и линий связи). Брюс входит в совет директоров Международной Ассоциации Криптологических Исследований (International Association for Cryptologic Research), является членом Консультационного совета Центра Секретности Электронной Информации (Electronic Privacy Information Center) и входит в комитет программы Семинара по Новым парадигмам Безопасности (New Security Paradigms Workshop). К тому же, он находит время для частых лекций по криптографии, компьютерной безопасности и секретности.

Глава 1
Основные понятия

1.1 Терминология

Отправитель и получатель

Предположим, что отправитель хочет послать сообщение получателю. Более того, этот отправитель хочет послать свое сообщение безопасно: он хочет быть уверен, что перехвативший это сообщение не сможет его пр очесть.

Сообщения и шифрование

Само сообщение называется **открытым текстом** (иногда используется термин клер). Изменение вида сообщения так, чтобы спрятать его суть называется **шифрованием**. Шифрованное сообщение называется **шифротекстом**. Процесс преобразования шифротекста в открытый текст называется **дешифрированием**. Эта последовательность показана на 0th.

(Если вы хотите следовать стандарту ISO 7498-2, то в английских текстах используйте термины "enchipher" вместо " encrypt" ("зашифровывать") и "dechipher" вместо " decrypt" ("дешифровывать")).

Искусство и наука безопасных сообщений, называемая **криптографией**, воплощается в жизнь **криптографами**. **Криптоаналитиками** называются те, кто постоянно используют **криптоанализ**, искусство и науку взламывать шифротекст, то есть, раскрывать, что находится под маской. Отрасль математики, охватывающая криптографию и криптоанализ, называется криптологией, а люди, которые ей занимаются, - **криптологами**. Современным криптологам приходится неплохо знать математику.

Рис. 1-1. Шифрование и дешифрирование

Обозначим открытый текст как *M* (от *message*, сообщение), или *P* (от *plaintext*, открытый текст). Это может быть поток битов, текстовый файл, битовое изображение, оцифрованный звук, цифровое видеоизображение… да что угодно. Для компьютера *M* - это просто двоичные данные. (Во всех следующих главах этой книги рассматриваются только двоичные данные и компьютерная криптография.) Открытый текст может быть создан для хранения или передачи. В любом случае, *M* - это сообщение, которое должно быть зашифровано.

Обозначим шифротекст как *C* (от *ciphertext*). Это тоже двоичные данные, иногда того же размера, что и *M*, иногда больше. (Если шифрование сопровождается сжатием, *C* может быть меньше чем *M*. Однако, само шифрование не обеспечивает сжатие информации.) Функция шифрования *E* действует на *M*, создавая *C*. Или, в математической записи:

$E(M) = C$

В обратном процессе функция дешифрирования *D* действует на *C*, восстанавливая *M:*

$D(C) = M$

Поскольку смыслом шифрования и последующего дешифрирования сообщения является восстановление первоначального открытого текста, должно выполняться следующее равенство:

$D(E(M)) = M$

Проверка подлинности, целостность и неотрицание авторства

Кроме обеспечения конфиденциальности криптография часто используется для других функций:

— **Проверка подлинности**. Получатель сообщения может проверить его источник, злоумышленник не сможет замаскироваться под кого-либо.

— **Целостность. Получатель сообщения может проверить, не было ли сообщение изменено в процессе доставки, злоумышленник не сможет подменить правильное сообщение ложным.**

— **Неотрицание авторства. Отправитель не сможет ложно отрицать отправку сообщения.**

Существуют жизненно важные требования к общению при помощи компьютеров, также как существуют ана-

логичные требования при общении лицом к лицу. То, что кто-то является именно тем, за кого он себя выдает … что чьи-то документы - водительские права, медицинская степень или паспорт - настоящие … что документ, полученный от кого-то, получен именно от этого человека… Как раз это обеспечивают проверка подлинности, целостность и неотрицание авторства.

Алгоритмы и ключи

Криптографический алгоритм, также называемый **шифром**, представляет собой математическую функцию, используемую для шифрования и дешифрирования. (Обычно это две связанных функции: одна для шифрования, а другая для дешифрирования.)

Если безопасность алгоритма основана на сохранении самого алгоритма в тайне, это **ограниченный** алгоритм. Ограниченные алгоритмы представляют только исторический интерес, но они совершенно не соответствуют сегодняшним стандартам. Большая или изменяющаяся группа пользователей не может использовать такие алгоритмы, так как всякий раз, когда пользователь покидает группу, ее члены должны переходить на другой алгоритм. Алгоритм должен быть заменен и, если кто-нибудь извне случайно узнает секрет.

Что еще хуже, ограниченные алгоритмы не допускают качественного контроля или стандартизации. У каждой группы пользователей должен быть свой уникальный алгоритм. Такие группы не могут использовать открытые аппаратные или программные продукты - злоумышленник может купить такой же продукт и раскрыть алгоритм. Им приходится разрабатывать и реализовывать собственные алгоритмы. Если в группе нет хорошего криптографа, то как ее члены проверят, что они пользуются безопасным алгоритмом?

Несмотря на эти основные недостатки ограниченные алгоритмы необычайно популярны для приложений с низким уровнем безопасности. Пользователи либо не понимают проблем, связанных с безопасностью своих систем, либо не заботятся о них.

Современная криптография решает эти проблемы с помощью **ключа** K. Такой ключ может быть любым значением, выбранным из большого множества. Множество возможных ключей называют **пространством ключей**. И шифрование, и дешифрирование этот ключ (то есть, они зависят от ключа, что обозначается индексом K), и теперь эти функции выглядят как:

$E_K(M)=C$

$D_K(C)=M$

При этом выполняется следующее равенство (см -1-й):

$D_K(E_K(M))=M$

Для некоторых алгоритмов при шифровании и дешифрировании используются различные ключи (см -2-й). То есть ключ шифрования, K_1, отличается от соответствующего ключа дешифрирования, K_2. В этом случае:

$$E_{K_1}(M)=C$$

$$D_{K_2}(C)=M$$

$$D_{K_2}(E_{K_1}(M))=M$$

Безопасность этих алгоритмов полностью основана на ключах, а не на деталях алгоритмов. Это значит, что алгоритм может быть опубликован и проанализирован. Продукты, использующие этот алгоритм, могут широко тиражироваться. Не имеет значения, что злоумышленнику известен ваш алгоритм, если ему не известен конкретный ключ, то он не сможет прочесть ваши сообщения.

Криптосистема представляет собой алгоритм плюс все возможные открытые тексты, шифротексты и ключи.

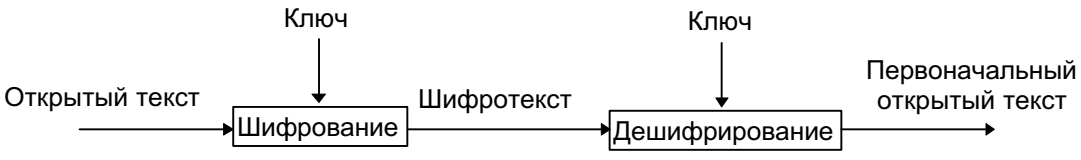

Рис. 1-2. Шифрование и дешифрирование с ключом

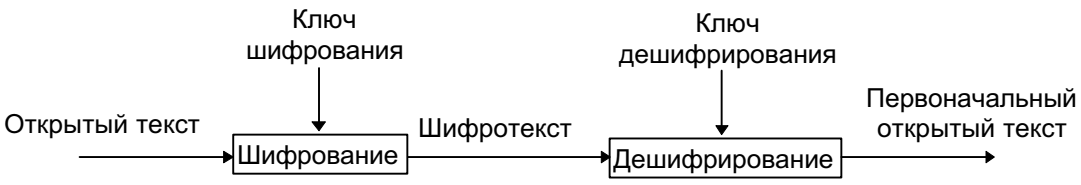

Рис. 1-3. Шифрование и дешифрирование с двумя различными ключами

Симметричные алгоритмы

Существует два основных типа алгоритмов, основанных на ключах: симметричные и с открытым ключом . **Симметричные алгоритмы**, иногда называемые условными алгоритмами , представляют собой алгоритмы, в которых ключ шифрования может быть рассчитан по ключу дешифрирования и наоборот . В большинстве симметричных алгоритмов кличи шифрования и дешифрирования одни и те же . Эти алгоритмы, также называемые алгоритмами с секретным ключом или алгоритмами с одним ключом, требуют, чтобы отправитель и получатель согласовали используемый ключ перед началом безопасной передачи сообщений . Безопасность симметричного алгоритма определяется ключом, раскрытие ключа означает, что кто угодно сможет шифровать и дешифрир овать сообщения. Пока передаваемые сообщения должны быть тайными, ключ должен храниться в секрете . Шифрование и дешифрирование с использованием симметричного алгоритма обозначается как :

$E_K(M)=C$

$D_K(C)=M$

Симметричные алгоритмы делятся на две категории . Одни алгоритмы обрабатывают открытый текст побитно (иногда побайтно), они называются **потоковыми алгоритмами** или **потоковыми шифрами**. Другие работаю с группами битов открытого текста. Группы битов называются блоками, а алгоритмы - **блочными алгоритмами** или **блочными шифрами**. Для алгоритмов, используемых в компьютерных модемах, типичный размер блока составляет 64 бита - достаточно большое значение, чтобы помешать анализу, и достаточно небольшое и удобное для работы. (До появления компьютеров алгоритмы обычно обрабатывали открытый текст посимвольно . Такой вариант может рассматриваться как потоковый алгоритм, обрабатывающий поток символов .)

Алгоритмы с открытым ключом

Алгоритмы с открытым ключом (называемые асимметричными алгоритмами) разработаны таким образом, что ключ, используемый для шифрования, отличается от ключа дешифрирования . Более того, ключ дешифрирования не может быть (по крайней мере в течение разумного интервала времени) рассчитан по ключу шифрования. Алгоритмы называются "с открытым ключом", потому что ключ шифрования может быть открытым: кто угодно может использовать ключ шифрования для шифрования сообщения, но только конкретный ч еловек с соответствующим ключом дешифрирования может расшифровать сообщение. В этих системах ключ шифрования часто называется **открытым** ключом, а ключ дешифрирования - **закрытым**. Закрытый ключ иногда называется секретным ключом, но чтобы не было путаницы с симметричными алгоритмами, этот термин не используется в данной книге. Шифрование с открытым ключом *K* обозначается как:

$E_K(M)=C$

Хотя открытый и закрытый ключи различны, дешифрирование с соответствующим закрытым ключом об означается как:

$D_K(C)=M$

Иногда сообщения шифруются закрытым ключом, а дешифрируются открытым, что используется для ци фровой подписи (см. раздел 2.6). Несмотря на возможную путаницу эти операции, соответственно, обозначаю тся как:

$E_K(M)=C$

$D_K(C)=M$

Криптоанализ

Смысл криптографии - в сохранении открытого текста (или ключа, или и того, и другого) в тайне от зл оумышленников (также называемых взломщиками , соперниками, врагами, перехватчиками). Предполагается, что злоумышленники полностью контролируют линии связи между отправителем и получателем .

Криптоанализ - это наука получения открытого текста, не имея ключа . Успешно проведенный криптоанализ может раскрыть открытый текст или ключ . Он также может обнаружить слабые места в криптосистемах, что в конце концов приведет к предыдущему результату . (Раскрытие ключа не криптологическими способами наз ы-

вается **компрометацией**.)

Попытка криптоанализа называется **вскрытием**. Основное предположение криптоанализа, впервые сформулированное в девятнадцатом веке Датчманом А. Керкхофсом (Dutchman A. Kerckhoffs), icостоит в том, что безопасность полностью определяется ключом [794]. Керкхофс предполагает, что у криптоаналитика есть полное описание алгоритма и его реализации. (Конечно же, у ЦРУ не в обычае сообщать Моссад о своих криптографических алгоритмах, но Моссад возможно все равно добудет их .) Хотя в реальном мире криптоаналитики не всегда обладают подробной информацией, такое предположение является хорошей рабочей гипотезой . Если противник не сможет взломать алгоритм, даже зная, как он работает, то тем более враг не сможет вскрыть а лгоритм без этого знания.

Существует четыре основных типа криптоаналитического вскрытия . Для каждого из них, конечно, предполагается, что криптоаналитик обладает всей полнотой знания об используемом алгоритме шифрования :

1. **Вскрытие с использованием только шифротекста** У криптоаналитика есть шифротексты нескольких сообщений, зашифрованных одним и тем же алгоритмом шифрования . Задача криптоаналитика состоит в раскрытии открытого текста как можно большего числа сообщений или, что лучше, получении ключа (ключей), использованного для шифрования сообщений, для дешифрировании других сообщений, зашифрованных теми же ключами.

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

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

2. **Вскрытие с использованием открытого текста** У криптоаналитика есть доступ не только к шифротекстам нескольких сообщений, но и к открытому тексту этих сообщений . Его задача состоит в получении ключа (или ключей), использованного для шифрования сообщений, для дешифрировании других сообщений, зашифрованных тем же ключом (ключами) .

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

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

3. **Вскрытие с использованием выбранного открытого текста** У криптоаналитика не только есть доступ к шифротекстам и открытым текстам нескольких сообщений, но и возможность выбирать открытый текст для шифрования. Это предоставляет больше вариантов чем вскрытие с использованием открытого текста, так как криптоаналитик может выбирать шифруемые блоки открытого текста, что может дать больше информации о ключе. Его задача состоит в получении ключа (или ключей), использованного для шифрования сообщений, или алгоритма, позволяющего дешифрировать новые сообщения, зашифрованные тем же ключом (или ключами) .

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

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

4. **Адаптивное вскрытие с использованием открытого текста** Это частный случай вскрытия с использованием выбранного открытого текста. Криптоаналитик не только может выбирать шифруемый текст, но также может строить свой последующий выбор на базе полученных результатов шифрования. При вскрытии с использованием выбранного открытого текста криптоаналитик мог выбрать для шифрования только один большой блок открытого текста, при адаптивном вскрытии с использованием выбранного открытого текста он может выбрать меньший блок открытого текста, затем выбрать следующий блок, используя результаты первого выбора и так далее .

Существует по крайней мере еше три типа криптоаналитической вскрытия .

5. **Вскрытие с использованием выбранного шифротекста** Криптоаналитик может выбрать различные шифротексты для дешифрирования и имеет доступ к дешифрированным открытым текстам . Например, у криптоаналитика есть доступ к "черному ящику", который выполняет автоматическое дешифрирование. Его задача состоит в получении ключа.

   Дано: C_1, $P_1=D_k(C_1)$, C_2, $P_2=D_k(C_2)$, . . . C_i, $P_i=D_k(C_i)$

   Получить: k

   Такой тип вскрытия обычно применим к алгоритмам с открытым ключом и обсуждается в разделе 19.3. Вскрытие с использование выбранного шифротекста иногда также эффективно против симметричных алгоритмов. (Иногда вскрытие с использованием выбранного открытого текста и вскрытие с использованием выбранного шифротекста вместе называют вскрытием с использованием выбранного текста.)

6. **Вскрытие с использованием выбранного ключа**. Такой тип вскрытия означает не то, что криптоаналитик может выбирать ключ, а что у него есть некоторая информация о связи между различными ключами. Этот странный, запутанный и не очень практичный тип вскрытия обсуждается в разделе 12.4.

7. **Бандитский криптоанализ**. Криптоаналитик угрожает, шантажирует или пытает кого-нибудь, пока не получит ключ. Взяточничество иногда называется **вскрытием с покупкой ключа**. Это очень мощные способы вскрытия, часто являющиеся наилучшим путем взломать алгоритм.

Вскрытия с известным открытым текстом и с использованием выбранного открытого текста встречаются чаще, чем можно подумать. Не является невозможным для криптоаналитика добыть открытый текст шифрованного сообщения или подкупить кого-нибудь, кто зашифрует выбранное сообщение. Может и не потребоваться никого подкупать - передав письмо послу, вы, возможно, обнаружите, что письмо будет зашифровано и отправлено в его страну для изучения. Многие сообщения имеют стандартные начало и окончание, что может быть известно криптоаналитику. Особенно уязвим шифрованный исходный код из-за частого использования ключевых слов: #define, struct, else, return. Те же проблемы и у шифрованного исполнимого кода: функции, циклические структуры и так далее. Вскрытия с известным открытым текстом (и вскрытия с выбранным шифротекстом) успешно использовались в борьбе с немцами и японцами в ходе Второй мировой войны. Исторические примеры вскрытий такого типа можно найти в книгах Дэвида Кана [794,795,796].

И не забывайте о предположении Керкхофса: если мощь вашей новой криптосистемы опирается на то, что взломщик не знает, как работает алгоритм, вы пропали. Если вы считаете, что хранение принципа работы алгоритма в секрете лучше защитит вашу криптосистему, чем предложение академическому сообществу проанализировать алгоритм, вы ошибаетесь. А если вы думаете, что кто-то не сможет дезассемблировать ваш исходный код и восстановить ваш алгоритм, вы наивны. (В 1994 году такое произошло с алгоритмом RC4, см. раздел 17.1.) Нашими лучшими алгоритмами являются те, которые были разработаны открыто, годами взламывались лучшими криптографами мира и все еще несокрушимы. (Агентство Национальной Безопасности хранит свои алгоритмы в секрете, но у них работают лучшие криптографы мира, а у вас - нет. Кроме того, они обсуждают свои алгоритмы друг с другом, полагаясь на способность товарища обнаружить все слабости в своей работе.)

У криптоаналитиков не всегда есть доступ к алгоритмам (например, вскрытие в ходе Второй мировой войны Соединенными Штатами японского дипломатического кода PURPLE [794]), но часто они его получают. Если алгоритм используется в коммерческой программе безопасности, то это просто вопрос времени и денег, удастся ли дезассемблировать программу и раскрыть алгоритм. Если же алгоритм используется в военной системе связи, то это просто вопрос времени и денег купить (или украсть) аппаратуру и реконструировать алгоритм.

Те, кто стремится получить нераскрываемый шифр, считая этот шифр таковым только потому, что они сами не смогли его взломать, либо гении, либо дураки. К несчастью, последних в мире достаточно много. Остерегайтесь людей, расхваливающих надежность своих алгоритмов, но отказывающихся их опубликовать. Доверять таким алгоритмам нельзя.

Хорошие криптографы опираются на мнение других, отделяя хорошие алгоритмы от плохих.

Безопасность алгоритмов

Различные алгоритмы предоставляют различные степени безопасности в зависимости от того, насколько трудно взломать алгоритм. Если стоимость взлома алгоритма выше, чем стоимость зашифрованных данных, вы, скорее всего, в безопасности. Если время взлома алгоритма больше, чем время, в течение которого зашифрованные данные должны сохраняться в секрете, то вы также, скорее всего, в безопасности. Если объем данных, зашифрованных одним ключом, меньше, чем объем данных, необходимый для взлома алгоритма, и тогда вы, скорее всего, в безопасности.

Я говорю "скорее всего", потому что существует вероятность новых прорывов в криптоанализе. С другой стороны, значимость большинства данных падает со временем. Важно, чтобы значимость данных всегда оставалась меньше, чем стоимость взлома системы безопасности, защищающей данные.

Ларс Кнудсен (Lars Knudsen) разбил вскрытия алгоритмов по следующим категориям, приведенным в порядке убывания значимости [858]:

1. **Полное вскрытие**. Криптоаналитик получил ключ, K, такой, что $D_K(C) = P$.

2. **Глобальная дедукция**. Криптоаналитик получил альтернативный алгоритм, A, эквивалентный $D_K(C)$ без знания K.

3. **Местная (или локальная) дедукция**. Криптоаналитик получил открытый текст для перехваченного шифротекста.

4. **Информационная дедукция**. Криптоаналитик получил некоторую информацию о ключе или открытом тексте. Такой информацией могут быть несколько бит ключа, сведения о форме открытого текста и так далее.

Алгоритм является **безусловно безопасным**, если, независимо от объема шифротекстов у криптоаналитика, информации для получения открытого текста недостаточно. По сути, только шифрование одноразовыми блокнотами (см. раздел 1.5) невозможно вскрыть при бесконечных ресурсах. Все остальные криптосистемы подвержены вскрытию с использованием только шифротекста простым перебором возможных ключей и проверкой осмысленности полученного открытого текста. Это называется вскрытием **грубой силой** (см. раздел 7.1).

Криптография больше интересуется криптосистемами, которые тяжело взломать вычислительным способом. Алгоритм считается **вычислительно безопасным** (или, как иногда называют, сильным), если он не может быть взломан с использованием доступных ресурсов сейчас или в будущем. Термин "доступные ресурсы" является достаточно расплывчатым. Сложность вскрытия можно измерить (см раздел 11.1) различными способами:

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

В качестве эмпирического метода сложность вскрытия определяется по максимальному из этих трех коэффициентов. Ряд операций вскрытия предполагают взаимосвязь коэффициентов: более быстрое вскрытие возможно за счет увеличения требований к памяти.

Сложность выражается порядком величины. Если сложность обработки для данного алгоритма составляет 2^{128}, то 2^{128} операций требуется для вскрытия алгоритма. (Эти операции могут быть сложными и длительными.) Так, если предполагается, что ваши вычислительные мощности способны выполнять миллион операций в секунду, и вы используете для решения задачи миллион параллельных процессоров, получение ключа займет у вас свыше 10^{19} лет, что в миллиард раз превышает время существования вселенной.

В то время, как сложность вскрытия остается постоянной (пока какой-нибудь криптоаналитик не придумает лучшего способа вскрытия), мощь компьютеров растет. За последние полвека вычислительные мощности феноменально выросли, и нет никаких причин подозревать, что эта тенденция не будет продолжена. Многие криптографические взломы пригодны для параллельных компьютеров: задача разбивается на миллиарды маленьких кусочков, решение которых не требует межпроцессорного взаимодействия. Объявление алгоритма безопасным просто потому, что его нелегко взломать, используя современную технику, в лучшем случае ненадежно. Хорошие криптосистемы проектируются устойчивыми к взлому с учетом развития вычислительных средств на много лет вперед.

Исторические термины

Исторически термин код относится к криптосистеме, связанной с лингвистическими единицами: словами, фразами, предложениями и так далее. Например, слово "ОЦЕЛОТ" может кодировать целую фразу "ПОВОРОТ НАЛЕВО НА 90 ГРАДУСОВ", слово "ЛЕДЕНЕЦ" - фразу "ПОВОРОТ НАПРАВО НА 90 ГРАДУСОВ", а слова "ПОДСТАВЬ УХО" могут кодировать слово "ГАУБИЦА". Коды такого типа не рассматриваются в данной книге, см. [794,795]. Коды полезны только при определенных обстоятельствах. Если у вас нет кода для "МУРАВЬЕДЫ", вы не сможете передать это понятие. А используя шифр можно сказать все.

1.2 Стеганография

Стеганография служит для передачи секретов в других сообщениях, так что спрятано само существование секрета. Как правило отправитель пишет какое-нибудь неприметное сообщение, а затем прячет секретное сообщение на том же листе бумаги. Исторические приемы включают невидимые чернила, невидимые простому глазу пометки у букв, плохо заметные отличия в написании букв, пометки карандашом машинописных символов, решетки, покрывающие большую часть сообщения кроме нескольких символов и тому подобное.

Ближе к сегодняшнему дню люди начали прятать секреты в графических изображениях, заменяя младший значащий бит изображения битом сообщения. Графическое изображение при этом менялось совсем незаметно - большинство графических стандартов определяют больше цветовых градаций, чем способен различить человеческий глаз - и сообщение извлекалось на противоположном конце. Так в черно-белой картинке 1024x1024 пиксела можно спрятать можно спрятать сообщение в 64 Кбайт. Многие общедоступные программы могут проделывать подобный фокус.

Имитационные функции Питера Уэйнера (Peter Wayner) маскируют сообщения. Эти функции изменяют сообщение так, что его статистический профиль становится похожим на что-нибудь еще: раздел *The New York*

Times, а пьесу Шекспира или телеконференцию в Internet [1584,1585]. Этот тип стеганографии не одурачит человека, но может обмануть большой компьютер, ищущий нужную информацию в Internet.

1.3 Подстановочные и перестановочные шифры

До появления компьютеров криптография состояла из алгоритмов на символьной основе . Различные криптографические алгоритмы либо заменяли одни символы другими, либо переставляли символы. Лучшие алгоритмы делали и то, и другое, и по много раз.

Сегодня все значительно сложнее, но философия остается прежней. Первое изменение заключается в том, что алгоритмы стали работать с битами, а не символами. Это важно хотя бы с точки зрения размера алфавита - с 26 элементов до двух. Большинство хороших криптографических алгоритмов до сих пор комбинирует подстановки и перестановки.

Подстановочные шифры

Подстановочным шифром называется шифр, который каждый символ открытого текста в шифротексте заменяет другим символом. Получатель инвертирует подстановку шифротекста, восстанавливая открытый текст . В классической криптографии существует четыре типа подстановочных шифров :

— **Простой подстановочный шифр**, или **моноалфавитный шифр**, - это шифр, который каждый символ открытого текста заменяет соответствующим символом шифротекста . Простыми подстановочными шифрами являются криптограммы в газетах.

— **Однозвучный подстановочный шифр** похож на простую подстановочную криптосистему за исключением того, что один символ открытого текста отображается на несколько символов шифротекста . Например, "A" может соответствовать 5, 13, 25 или 56, "B" - 7, 19, 31 или 42 и так далее.

— **Полиграмный подстановочный шифр** - это шифр, который блоки символов шифрует по группам . Например, "ABA" может соответствовать "RTQ", "ABB" может соответствовать "SLL" и так далее.

— **Полиалфавитный подстановочный шифр** состоит из нескольких простых подстановочных шифров . Например, могут быть использованы пять различных простых подстановочных фильтров ; каждый символ открытого текста заменяется с использованием одного конкретного шифра .

Знаменитый **шифр Цезаря**, в котором каждый символ открытого текста заменяется символом, находящегося тремя символами правее по модулю 26 ("A" заменяется на "D," "B" - на "E", ... "W" - на " Z ", "X" - на "A", "Y" - на "B", "Z" - на "C"), представляет собой простой подстановочный фильтр . Он действительно очень прост, так как алфавит шифротекста представляет собой смещенный, а не случайно распределенный алфавит открытого текста.

ROTI3 - это простая шифровальная программа, обычно поставляемая с системами UNIX. Она также является простым подстановочным шифром. В этом шифре "A" заменяется на "N," "B" - на "O" и так далее. Каждая буква смещается на 13 мест. Шифрование файла программой ROTI3 дважды восстанавливает первоначальный файл.

P = ROT13 (ROT13 (P))

ROTI3 не используется для безопасности, она часто применяется в почте, закрывая потенциально неприятный текст, решение головоломки и тому подобное .

Простые подстановочные шифры легко раскрываются, так как шифр не прячет частоты использования различных символов в открытом тексте. Чтобы восстановить открытый текст, хорошему криптоаналитику требуется только знать 26 символов английского алфавита [1434]. Алгоритм вскрытия таких шифров можно найти в [578, 587, 1600, 78, 1475, 1236, 880]. Хороший компьютерный алгоритм приведен в [703].

Однозвучные подстановочные шифры использовались уже в 1401 году в герцогстве Мантуа [794]. Они более сложны для вскрытия, чем простые подстановочные шифры, хотя и они не скрывают всех статистических свойств языка открытого текста. При помощи вскрытия с известным открытым текстом эти шифры раскрываются тривиально. Вскрытие с использованием только шифротекста более трудоемко, но и оно занимает на компьютере лишь несколько секунд. Подробности приведены в [1261].

Полиграмные подстановочные шифры - это шифры, которые кодируют сразу группы символов . Шифр Playfair ("Честная игра"), изобретенный в 1854 году, использовался англичанами в Первой мировой войне [794]. Он шифрует пары символов, и его криптоанализ обсуждается в [587,1475,880]. Другим примером полиграмного подстановочного шифра является шифр Хилла (Hill) [732]. Иногда можно видеть как вместо шифра используется кодирование по Хаффману (Huffman), это небезопасный полиграмный подстановочный шифр .

Полиалфавитные подстановочные шифры были изобретены Лином Баттистой (Lean Battista) в 1568 году

[794]. Они использовались армией Соединенных Штатов в ходе Гражданской войны в Америке . Несмотря на то, что они легко могут быть взломаны [819, 577, 587, 794] (особенно с помощью компьютеров), многие коммерческие продукты компьютерной безопасности используют такие шифры [1387,1390, 1502]. (Подробности того, как вскрыть эту схему шифрования, используемую программой WordPerfect, можно найти в [135,139].) Шифр Вигенера (Vigenere), впервые опубликованный в 1586 году, и шифр Бофора (Beaufort) также являются примерами полиалфавитных подстановочных шифров.

У полиалфавитных подстановочных шифров множественные однобуквенные ключи, каждый из которых используется для шифрования одного символа открытого текста . Первым ключом шифруется первый символ открытого текста, вторым ключом - второй символ, и так далее . После использования всех ключей они повторяются циклически. Если применяется 20 однобуквенных ключей, то каждая двадцатая буква шифруется тем же ключом. Этот параметр называется **периодом** шифра. В классической криптографии шифры с длинным периодом было труднее раскрыть, чем шифры с коротким периодом. Использование компьютеров позволяет легко раскрыть подстановочные шифры с очень длинным периодом .

Шифр с бегущим ключом (иногда называемый книжным шифром), использующий один текст для шифрования другого текста, представляет собой другой пример подобного шифра . И хотя период этого шифра равен длине текста, он также может быть легко взломан [576,794].

Перестановочные шифры

В **перестановочном шифре** меняется не открытый текст, а порядок символов. В **простом столбцовом перестановочном шифре** открытый текст пишется горизонтально на разграфленном листе бумаги фиксированной ширины, а шифротекст считывается по вертикали (см. -3-й). Дешифрирование представляет собой запись шифротекста вертикально на листе разграфленной бумаги фиксированной ширины и затем считывание открытого текста горизонтально.

Криптоанализ этих шифров обсуждается в [587,1475]. Так как символы шифротекста те же, что и в открытом тексте, частотный анализ шифротекста покажет, что каждая буква встречается приблизительно с той же частотой, что и обычно. Это даст криптоаналитику возможность применить различные методы, определяя правильный порядок символов для получения открытого текста . Применение к шифротексту второго перестановочного фильтра значительно повысит безопасность . Существуют и еще более сложные перестановочные фильтры, но компьютеры могут раскрыть почти все из них .

Немецкий шифр ADFCVX, использованный в ходе Первой мировой войны , представлял собой перестановочный фильтр в сочетании с простой подстановкой . Этот для своего времени очень сложный алгоритм был раскрыт Жоржем Пенвэном (Georges Painvin), французским криптоаналитиком [794].

Хотя многие современные алгоритмы используют перестановку, с этим связана проблема использования большого объема памяти, а также иногда требуется работа с сообщениями определенного размера . Подстановка более обычна.

Роторные машины

В 1920-х годах для автоматизации процесса шифрования были изобретены различные механические устройства. Большинство использовало понятие **ротора**, механического колеса, используемого для выполнения подстановки.

Роторная машина, включающая клавиатуру и набор роторов , реализует вариант шифра Вигенера. Каждый ротор представляет собой произвольное размещение алфавита, имеет 26 позиций и выполняет простую подстановку. Например, ротор может быть использован для замены "A" на " F", "B" на "U", "C" на "I" и так далее. Выходные штыри одного ротора соединены с входными штырями следующего ротора.

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

```
COMPUTERGR
APHICSMAYB
ESLOWBUTAT
LEASTITSEX
PENSIVE
```

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

Рис. 1-4. Столбцовый перестановочный фильтр.

Например, в четырехроторной машине первый ротор может заменять "A" на " F", второй - "F" на "Y", третий - "Y" на "E" и четвертый - "E" на "C", "C" и будет конечным шифротекстом. Затем некоторые роторы смещаются, и в следующий раз подстановки будут другими .

Именно комбинация нескольких роторов и механизмов, движущих роторами, и обеспечивает безопасность машины. Так как роторы вращаются с различной скоростью, период для n-роторной машины равен 26^n. Некоторые роторные машины также могут иметь различные положения для каждого ротора, что делает криптоанализ еще более бессмысленным.

Самым известным роторным устройство является Энигма (Enigma). Энигма использовалась немцами во Второй мировой войне. Сама идея пришла в голову Артуру Шербиусу (Arthur Scherbius) и Арвиду Герхарду Дамму (Arvid Gerhard Damm) в Европе. В Соединенных Штатах она была запатентована Артуром Шербиусом [1383]. Немцы значительно усовершенствовали базовый проект для использования во время войны.

У немецкой Энигмы было три ротора, котроые можно было выбрать из пяти возможных, коммутатор, который слегка тасовал открытый текст, и отражающий ротор, который заставлял каждый ротор обрабатывать открытый текст каждого письма дважды. Несмотря на сложность Энигмы, она была взломана в течение Второй мировой войны. Сначала группа польских криптографов взломала немецкую Энигму и объяснила раскрытый алгоритм англичанам. В ходе войны немцы модифицировали Энигму, а англичане продолжали криптоанализ новых версий. Объяснение работы роторных шифров и способов их раскрытия можно найти в [794, 86, 448, 498, 446, 880, 1315, 1587, 690]. В двух следующих отчетах увлекательно рассказывается о взломе Энигмы [735, 796].

Для дальнейшего чтения

Данная книга не является книгой по классической криптографии, поэтому далее я не буду подробно останавливаться на этих предметах. Прекрасными книгами по докомпьютерной криптологии являются [587, 1475]. [448] содержит современный криптоанализ шифровальных машин. Дороти Деннинг (Dorothy Denning) рассматривает многие из этих шифров в [456], а [880] содержит беспристрастный сложный математический анализ тех же самых шифров. Другим описанием старой криптографии, описывающим аналоговую криптографию, является [99]. Прекрасный обзор выполнен в статье [579]. Великолепны также книги по исторической криптографии Дэвида Кана [794, 795, 796].

1.4 Простое XOR

XOR представляет собой операцию "исключающее или": '^' в языке C или Q в математической нотации. Это обычная операция над битами:

$$0 \oplus 0 = 0$$

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

$$1 \oplus 1 = 0$$

Также заметим, что:

$$a \oplus a = 0$$

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

Казалось бы, запутанный алгоритм простого XOR по сути является ничем иным, как полиалфавитным шифром Вигенера. Здесь он упоминается только из-за распространенности в коммерческих программных продуктах, по крайней мере в мире MS-DOS и Macintosh [1502, 1387]. К сожалению, если о программе компьютерной безопасности заявляется, что это "патентованный" алгоритм шифрования, значительно более быстрый, чем DES, то скорее всего используется какой-то вариант следующего.

```
/* Использование: crypto key input_file output_file */
void main (int argc, char *argv[])
{
      FILE *fl, *fo;
      char *cp;
      int c;

      if ((cp = argv[l]) && *cp!= '\0') {
            if ((fi = fopen(argvl[2], "rb")) != NULL) {
                  if ((fo = fopen(argv[3], "wb")) != NULL) {
                        while ((c = getc(fi)) != EOF) {
                              if (!*cp) cp = argv[1];
                              c^= *(cp++);
                              putc(c,fo);
                        }
                        fclose(fo);
                  }
                  fclose(fi);
            }
```

```
            }
        }
```

Это симметричный алгоритм. Открытый текст подвергается операции "исключающее или" вместе с ключевым текстом для получения шифротекста. Так как повторное применение операции XOR восстанавливает оригинал для шифрования и дешифрирования используется одна и та же программа :

$$P \oplus K = C$$

$$C \oplus K = P$$

Настоящей безопасности здесь никогда не было. Этот тип шифрования легко вскрывается, даже без компьютера [587, 1475]. Его взлом на компьютере занимает несколько секунд.

Предположим, что открытый текст использует английский язык. Более того, пусть длина ключа любое небольшое число байт. Ниже описано, как взломать этот шифр:

1. Определим длину ключа с помощью процедуры, известной как **подсчет совпадений** [577]. Применим операцию XOR к шифротексту, используя в качестве ключа сам шифротекст с различными смещениями, и подсчитаем совпадающие байты. Если величина смещения кратна длине ключа, то совпадет свыше 6 процентов байтов. Если нет, то будут совпадать меньше чем 0.4 процента (считая, что обычный ASCII текст кодируется случайным ключом, для других типов открытых текстов числа будут другими). Это называется **показателем совпадений**. Минимальное смещение от одного значения, кратного длине ключа, к другому и есть длина ключа.
2. Сместим шифротекст на эту длину и проведем операцию XOR для смещенного и оригинального шифротекстов. Результатом операции будет удаления ключа и получение открытого текста, подвергнутого операции XOR с самим собой, смещенным на длину ключа. Так как в английском языке на один байт приходится 1.3 бита действительной информации (см раздел 11.1), существующая значительная избыточность позволяет определить способ шифрования.

Несмотря на это, количество поставщиков программного обеспечения, навязывающих этот игрушечный алгоритм в качестве "почти такого же безопасного как DES", впечатляет [1387]. Именно этот алгоритм (с 160-битным повторяющимся "ключом") NSA в конце концов разрешило использовать в цифровых телефонных сотовых сетях для закрытия голоса. XOR может защитить ваши файлы от младшей сестры, но настоящего криптоаналитика задержит лишь на считанные секунды.

1.5 Одноразовые блокноты

Поверите или нет, но идеальный способ шифрования существует. Он называется **одноразовым блокнотом** и был изобретен в 1917 году Мэйджором Джозефом Моборном (Major Joseph Mauborgne) и Гилбертом Вернамом (Gilbert Vernam) из AT&T [794]. (Фактически одноразовый блокнот представляет собой особый случай пороговой схемы, см. раздел 3.7.) В классическом понимании одноразовый блокнот является большой неповторяющейся последовательностью символов ключа, распределенных случайным образом, написанных на кусочках бумаги и приклеенных к листу блокнота. Первоначально это была одноразовая лента для телетайпов. Отправитель использовал каждый символ ключа блокнота для шифрования только одного символа открытого текста. Шифрование представляет собой сложение по модулю 26 символа открытого текста и символа ключа из одноразового блокнота.

Каждый символ ключа используется только единожды и для единственного сообщения . Отправитель шифрует сообщения и уничтожает использованные страницы блокнота или использованную часть ленты . Получатель, в свою очередь, используя точно такой же блокнот, дешифрирует каждый символ шифротекста . Расшифровав сообщение, получатель уничтожает соответствующие страницы блокнота или часть ленты . Новое сообщение - новые символы ключа. Например, если сообщением является:

ONETIMEPAD

а ключевая последовательность в блокноте:

TBFRGFARFM

то шифротекст будет выглядеть как:

IPKLPSFHGQ

так как

Q + T mod 26 = I

N + B mod 26 = P

E + F mod 26 = K

и т.д.

В предположении, что злоумышленник не сможет получить доступ к одноразовому блокноту, использова нному для шифрования сообщения, эта схема совершенно безопасна. Данное шифрованное сообщение на вид соответствует любому открытому сообщению того же размера.

Так как все ключевые последовательности совершенно одинаковы (помните, символы ключа генерируются случайным образом), у противника отсутствует информация, позволяющая подвергнуть шифротекст криптоан ализу. Кусочек шифротекста может быть похож на:

POYYAEAAZX

что дешифрируется как:

SALMONEGGS

или на:

BXEGBMTMXM

что дешифрируется как:

GREENFLUID

Повторю еще раз: так как все открытые тексты равновероятны, у криптоаналитика нет возможности опред елить, какой из открытых текстов является правильным. Случайная ключевая последовательность, сложенная с неслучайным открытым текстом, дает совершенно случайный шифротекст, и никакие вычислительные мощн ости не смогут это изменить.

Необходимо напомнить, что символы ключа должны генерироваться случайным образом. Любые попытки вскрыть такую схему сталкиваются со способом, которым создается последовательность символов ключа. Использование генераторов псевдослучайных чисел не считается, у них всегда неслучайные свойства. Если вы используете действительно случайный источник - это намного труднее, чем кажется на первый взгляд, см. ра здел 17.14 - это совершенно безопасно.

Другой важный момент: ключевую последовательность никогда нельзя использовать второй раз. Даже если вы используете блокнот размером в несколько гигабайт, то если криптоаналитик получит несколько текстов с перекрывающимися ключами, он сможет восстановить открытый текст. Он сдвинет каждую пару шифротекстов относительно друг друга и подсчитает число совпадений в каждой позиции. Если шифротексты смещены пр авильно, соотношение совпадений резко возрастет - точное значение зависит от языка открытого текста. С этой точки зрения криптоанализ не представляет труда. Это похоже на показатель совпадений, но сравниваются два различных "периода" [904]. Не используйте ключевую последовательность повторно.

Идея одноразового блокнота легко расширяется на двоичные данные. Вместо одноразового блокнота, с остоящего из букв, используется одноразовый блокнот из битов. Вместо сложения открытого текста с ключом одноразового блокнота используйте XOR. Для дешифрирования примените XOR к шифротексту с тем же одноразовым блокнотом. Все остальное не меняется, и безопасность остается такой же совершенной.

Все это хорошо, но существует несколько проблем. Так как ключевые биты должны быть случайными и не могут использоваться снова, длина ключевой последовательности должна равняться длине сообщения. Одноразовый блокнот удобен для нескольких небольших сообщений, но его нельзя использовать для работы по каналу связи с пропускной способностью 1.544 Мбит/с. Вы можете хранить 650 Мбайт случайных данных на CD-ROM, но и тут есть проблемы. Во первых, вам нужно только две копии случайных битов, но CD-ROM экономичны только при больших тиражах. И во вторых, вам нужно уничтожать использованные биты. Для CD-ROM нет другой возможности удалить информацию кроме как физически разрушить весь диск. Гораздо больше подходит цифровая лента.

Даже если проблемы распределения и хранения ключей решены, вам придется точно синхронизировать р аботу отправителя и получателя. Если получатель пропустит бит (или несколько бит пропадут при передаче), сообщение потеряет всякий смысл. С другой стороны, если несколько бит изменятся при передаче (и ни один бит не будет удален или добавлен - что гораздо больше похоже на влияние случайного шума), то лишь эти биты будут расшифрованы неправильно. Но одноразовый блокнот не обеспечивает проверку подлинности.

Одноразовые блокноты используются и сегодня, главным образом для сверхсекретных каналов связи с ни зкой пропускной способностью. По слухам "горячая линия" между Соединенными Штатами и бывшим Совет ским Союзом (а действует ли она сейчас?) шифруется с помощью одноразового блокнота. Многие сообщения советских шпионов зашифрованы с использованием одноразовых блокнотов. Эти сообщения нераскрыты сего дня и навсегда останутся нераскрытыми. На этот факт не повлияет время работы суперкомпьютеров над этой

проблемой. Даже когда враги из созвездия Андромеды приземлят свои тяжелые корабли с компьютерами н емыслимой мощности, и они не смогут прочесть сообщения советских шпионов, зашифрованные с помощью о дноразовых (если, конечно, они не смогут вернуться в прошлое и добыть нужные одноразовые блокноты).

1.6 Компьютерные алгоритмы

Существует множество компьютерных алгоритмов. Следующие три используются чаще всего :

— DES (Data Encryption Standard, стандарт шифрования данных) - самый популярный компьютерный алгоритм шифрования, является американским и международным стандартом . Это симметричный алгоритм, один и тот же ключ используется для шифрования и дешифрирования .

— RSA (назван в честь создателей - Ривеста (Rivest), Шамира (Sharnir) и Эдлмана (Adleman)) - самый популярный алгоритм с открытым ключом. Используется и для шифрования, и для цифровой подписи .

— DSA (Digital Signature Algorithm, алгоритм цифровой подписи, используется как часть стандарта цифровой подписи, Digital Signature Standard) - другой алгоритм с открытым ключом. Используется только для цифровой подписи, не может быть использован для шифрования .

Именно эти и подобные алгоритмы описываются в этой книге .

1.7 Большие числа

На протяжении всей книги я использую различные большие числа для описания различных вещей в крипт ографии. Так как легко заблудиться в этих числах и их значениях, физические аналоги некоторых чисел прив едены в 0-й.

Эти числа оцениваются по порядку величины и были отобраны из различных источников. Многие астроф изические значения объясняются в работе Фримана Дайсона (Freeman Dyson), "Время без конца: физика и биология в открытой Вселенной" ("Time Without End: Physics and Biology in an Open Universe") в *Reviews of Modem Physics,* v. 52, n. 3, July 1979, pp. 447-460. Смертность в результате автокатастроф рассчитана с помощью стат истики Министерства транспорта (163 смерти миллион человек в 1993 году и для средней продолжительности жизни 69.7 года.

Табл. 1-1. Большие числа

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

Если Вселенная конечна:

Полное время жизни вселенной	10^{11} (2^{37}) лет 10^{18} (2^{61}) секунд
Если Вселенная бесконечна:	
Время до остывания легких звезд	10^{14} (2^{47}) лет
Время до отрыва планет от звезд	10^{15} (2^{50}) лет
Время до отрыва звезд от галактик	10^{19} (2^{64}) лет
Время до разрушения орбит гравитационной радиацией	10^{20} (2^{67}) лет
Время до разрушения черных дыр процессами Хокинга	10^{64} (2^{213}) лет
Время до превращения материи в жидкость при нулевой температуре	10^{65} (2^{216}) лет
Время до превращения материи в твердое тело	$10^{10^{26}}$ лет
Время до превращения материи в черную дыру	$10^{10^{76}}$ лет

Часть 1

КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

Глава 2
Элементы протоколов

2.1 Введение в протоколы

Смысл криптографии - в решении проблем. (По сути, в этом состоит и смысл использования компьютеров, о чем многие пытаются забыть.) Криптография решает проблемы секретности, проверки подлинности, целостности и человеческой нечестности. Вы можете выучить все о криптографических алгоритмах и методах, но они представляют только академический интерес, если не используются для решения какой-нибудь проблемы. Именно поэтому мы собираемся сначала взглянуть на протоколы.

Протокол - это порядок действий, предпринимаемых двумя или более сторонами, предназначенный для решения определенной задачи. Это важное определение. "Порядок действий" означает, протокол выполняется в определенной последовательности, с начала до конца. Каждое действие должно выполняться в свою очередь и только после окончания предыдущего. "Предпринимаемых двумя или более сторонами" означает, что для реализации протокола требуется по крайней мере два человека, один человек не сможет реализовать протокол. Человек в одиночку может выполнить некоторые действия, решая задачу (например, покупая торт), но это не протокол. (Для того, чтобы получился настоящий протокол, кто-то должен съесть торт.) Наконец, "предназначенный для решения определенной задачи" означает, что протокол должен приводить к какому-то результату. Что-то, похожее на протокол, но не решающее никакой задачи - это не протокол, это потеря времени. У протоколов есть также и другие характеристики:

— Каждый участник протокола должен знать протокол и последовательность составляющих его действий.

— Каждый участник протокола должен согласиться следовать протоколу.

— Протокол должен быть непротиворечивым, каждое действие должно быть определено так, чтобы не было возможности непонимания.

— Протокол должен быть полным, каждой возможной ситуации должно соответствовать определенное действие.

В этой книге каждый протокол организован как некоторый порядок действий. Выполнение протокола происходит по действиям, линейно, пока не будет команды перейти к следующему действию. Каждое действие включает по крайней мере одно из двух: вычисления, выполняемые одной или несколькими сторонами, или сообщения, которыми обмениваются стороны.

Криптографический протокол - это протокол, использующий криптографию. Стороны могут быть друзьями и слепо доверять друг другу или врагами и не верить друг другу даже при сообщении времени суток. Криптографический протокол включает некоторый криптографический алгоритм, но, вообще говоря, предназначение протокола выходит за рамки простой безопасности. Участники протокола могут захотеть поделиться секретом друг с другом, совместно генерировать случайную последовательность, подтвердить друг другу свою подлинность или подписать контракт в один и тот же момент времени. Смысл использования криптографии в протоколе - в предотвращении или обнаружении вредительства и мошенничества. Если вы никогда не сталкивались с подобными протоколами, они могут радикально изменить ваше представление о том, что недоверяющие друг другу стороны могут выполнить, используя компьютерную сеть. Общее правило можно сформулировать следующим образом:

— Невозможно сделать или узнать больше, чем определено в протоколе.

Это гораздо сложнее, чем кажется. В следующих нескольких главах я рассматриваю множество протоколов. В некоторых из них один из участников может обмануть другого. В других, злоумышленник может взломать протокол или узнать секретную информацию. Ряд протоколов проваливаются, так как их разработчики недостаточно тщательно определяли требования. Другие проваливаются из-за того, что их разработчики недостаточно тщательно анализировали свои протоколы. Как и для алгоритмов, гораздо легче доказать возможную небезопасность протокола, чем его полную безопасность.

Смысл протоколов

В повседневной жизни почти для всего существуют неформальные протоколы: заказ товаров по телефону, игра в покер, голосование на выборах. Никто не задумывается об этих протоколах, они вырабатывались в течение длительного времени, все знают, как ими пользоваться и они работают достаточно хорошо.

Сегодня все больше и больше людей общаются не лично, а используя компьютерную сеть. Для тех же вещей, которые люди делают не задумываясь, компьютерам нужны формальные протоколы. Когда вы переезжаете из государства в государство и обнаруживаете кабинку, совершенно отличающуюся от той, к которой вы при-

выкли, вы легко адаптируетесь. Компьютеры далеко не так гибки .

Честность и безопасность многих протоколов человеческого общения основаны на личном присутствии . Разве вы дадите незнакомцу кучу денег, чтобы он купил для вас что-нибудь в бакалее ? Сядете ли вы играть в покер с тем, кто жульничает, сдавая карты? Пошлете ли вы свой избирательный бюллетень правительству, не будучи уверенным в тайности такого голосования?

Наивно считать, что пользователи компьютерных сетей всегда честны. Также наивно считать, что всегда честны разработчики компьютерных сетей. Для большинства из них это именно так, но даже несколько жуликов могут принести много вреда. Формализируя протоколы, можно проверить способы, используемые жуликами для взлома протоколов. Так мы можем разработать протоколы, устойчивые к взлому.

Кроме формализации действий, протоколы позволяют абстрагироваться при решении задачи от способа решения. Протокол связи один и тот же и на PC, и на VAX. Можно проверить протокол, не вдаваясь в детали его реализации. Когда мы убедимся в надежности протокола, его можно будет реализовать где угодно от компьютеров до телефонов и интеллектуальных тостеров .

Игроки

Для демонстрации работы протоколов я использую несколько игроков (см. 1-й). Первые двое - это Алиса и Боб. Они участвуют во всех двусторонних протоколах . Как правило, Алиса (Alice) начинает все протоколы, а Боб (Bob) отвечает. Если для протокола нужна третья или четвертая сторона, в игру вступают Кэрол (Êýðîë) и Дэйв (Dave). Другие игроки играют специальные вспомогательные роли, они будут представлены по зже.

Протоколы с посредником

Посредник - это незаинтересованная третья сторона, которой доверено завершение протокола (см. 1-й (a)). Незаинтересованность означает, что у посредника нет заинтересованности в результате работы протокола и склонности к одной из сторон . "Доверено" означает, что все участники протокола принимают все, что скажет посредник за истину, все его действия - как правильные, и уверены в том, что посредник выполнит свою часть протокола. Посредники помогают реализовать работу протоколов взаимодействия недоверяющих друг другу сторон.

В реальном мире в качестве посредников часто выступают юристы . Например, Алиса продает незнакомому ей Бобу машину. Боб хочет заплатить чеком, но у Алисы нет способа проверить, действителен ли чек . Алиса хочет, чтобы расчет по чеку был произведен прежде, чем право собственности перейдет к Бобу . Боб, который верит Алисе не больше, чем она ему, не хочет передавать чек, не получив права собственности .

Табл. 2-1. Действующие лица

Алиса	Первый участник всех протоколов
Боб	Второй участник всех протоколов
Кэрол	Третий участник в протоколах с участием трех и четырех сторон
Дэйв	Четвертый участник в протоколах с участием трех и четырех сторон
Ева	Злоумышленник (eavesdropper)
Мэллори	Взломщик протоколов
Трент	Заслуживающий доверия посредник
Уолтер	Контролер, защищает Алису и Боба в ряде протоколов
Пегги	Свидетель
Виктор	Проверяет подлинность

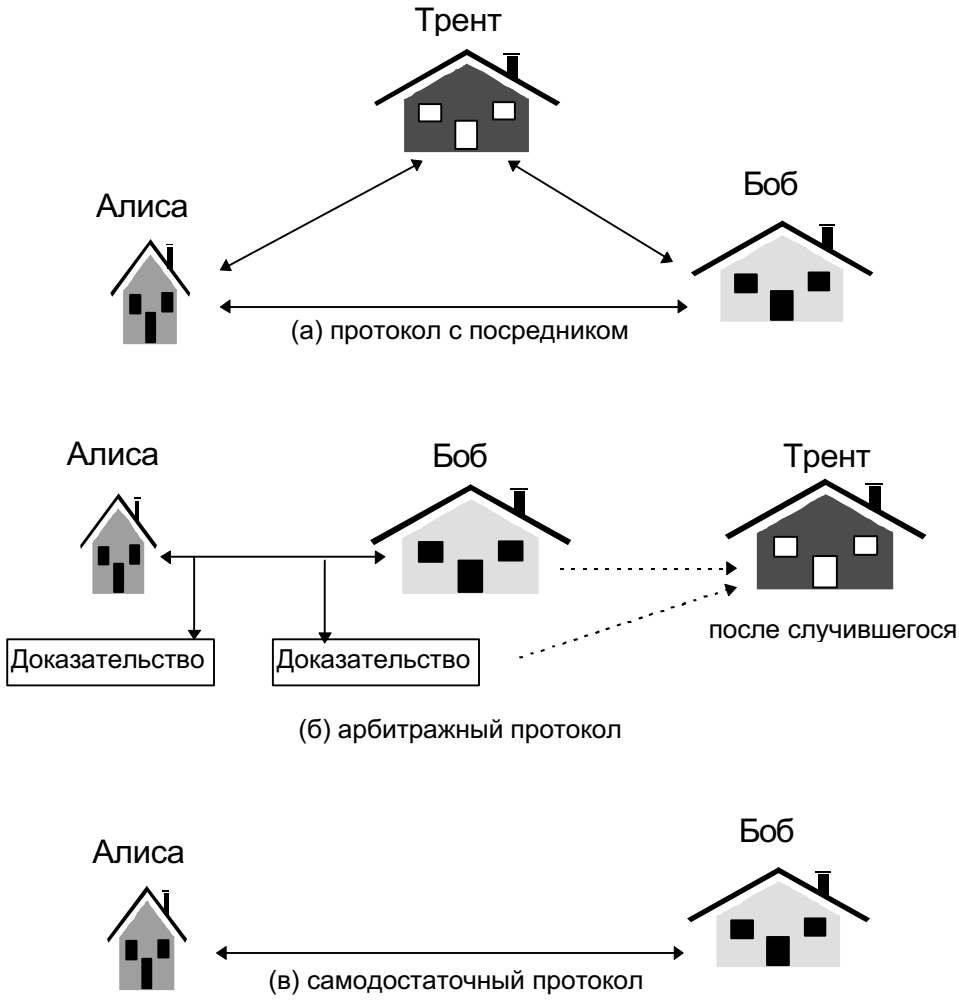

Рис. 2-1. Типы протоколов

Посредничество юриста устроит обоих. С его помощью Алиса и Боб могут выполнить следующий протокол, чтобы защитить себя от обмана:

(1) Алиса передает право собственности юристу.

(2) Боб передает чек юристу.

(3) Алиса депонирует чек.

(4) Дождавшись оплаты чека юрист передает право собственности Бобу. Если чек не оплачен в течение определенного времени, Алиса доказывает этот факт юристу, и тот возвращает право собственности Алисе.

В этом протоколе Алиса верит, что юрист не передаст Бобу право собственности до тех пор, пока чек не будет оплачен, и вернет право собственности Алисе, если чек оплачен не будет. Боб верит, что юрист будет обладать правом собственности до тех пор, пока чек не будет оплачен, и передаст право собственности Бобу сразу же после оплаты чека. Юрист не заботится об оплате чека. Он в любом случае выполнит свою часть протокола, ведь ему заплатят в любом случае.

В этом примере юрист играет роль посредника. Юристы часто выступают в роли посредников при завещаниях и иногда при переговорах о контракте. Различные биржи выступают в качестве посредников между покупателями и продавцами.

В качестве посредника может выступить и банк - для покупки машины:

(1) Боб заполняет чек и передает его в банк.

(2) Если на счету Боба достаточно денег для покрытия чека, банк заверяет чек и возвращает его Бобу.

(3) Алиса передает Бобу право собственности, а Боб передает Алисе заверенный чек.

(4) Алиса депонирует чек.

Этот протокол работает, потому что Алиса верит банковскому свидетельству. Алиса верит, что банк сохранит деньги Боба для нее и не использует их для финансирования сомнительных операций с недвижимостью в

банановых республиках.

Другим общепринятым посредником является нотариус. Когда Боб получает от Алисы заверенный нотариусом документ, он убежден, что Алиса подписала документ по своему желанию и собственноручно. При необходимости нотариус может выступить в суде и засвидетельствовать этот факт.

Понятие посредника старо как мир. Всегда существовали определенные люди - вожди, жрецы и тому подобное - обладавшие влиянием, позволяющим им действовать справедливо. Посредники играют определенную роль в нашем обществе, обман доверия подорвал бы занимаемое ими положение. Юристы-посредники, нарушающие правила игра, подвергаются наказанию - например, исключению из коллегии адвокатов. Это идеальная картина, в реальном мире положение, к сожалению, может отличаться от нее.

Этот идеал можно перенести на мир компьютеров, но с компьютерными посредниками существует ряд проблем:

— Легко найти нейтральную третью сторону, которой можно доверять, если вы знаете посредника и можете лично увидеть его. Две стороны, относящиеся друг к другу с подозрением, с тем же подозрением отнесутся и к безликому посреднику, затерянному где-то в сети.

— Компьютерная сеть должна обеспечить поддержку посредника. Занятость юристов общеизвестна, на кого в сети лягут дополнительные накладные расходы?

— Существует задержка, присущая всем протоколам с посредником.

— Посредник должен принимать участие в каждой транзакции, являясь узким местом в крупномасштабных реализациях любого протокола. Рост числа посредников смягчит эту проблему, но вырастет и цена этой услуги.

— Так как каждый в сети должен доверять посреднику, то посредник представляет собой слабое место сети при попытке ее взлома.

Несмотря на это посредничество все еще активно используется. В протоколах с использованием посредника эту роль будет играть Трент.

Арбитражные протоколы

Используемый из-за высокой стоимости найма посредников арбитражные протокол может быть разбит на два **подпротокола** нижнего уровня. Первый представляет собой протокол без посредника, используемый при желании сторон выполнить протокол. Другой представляет собой протокол с посредником, приглашаемым в исключительных обстоятельствах - при наличии разногласий между сторонами. Соответствующий специальный посредник называется **арбитром** (см. 1-й(б)).

Арбитр, как и посредник, представляет собой незаинтересованного участника протокола, которому доверяют обе стороны. В отличие от посредника он непосредственно не принимает участия в каждой отдельной реализации протокола и приглашается только для проверки честности выполнения протокола сторонами.

Профессиональными арбитрами являются судьи. В отличие от нотариусов к судьям обращаются только при появлении разногласий. Алиса и Боб могут заключить контракт без участия судьи. Судья никогда не узнает о контракте, если одна из сторон не подаст на другую в суд. Протокол подписания контракта можно формализовать следующим образом:

Подпротокол без посредника (выполняется всегда):

(1) Алиса и Боб договариваются об условиях контракта.

(2) Алиса подписывает контракт.

(3) Боб подписывает контракт.

Подпротокол с использованием арбитра (выполняется при наличии разногласий):

(4) Алиса и Боб предстают перед судьей.

(5) Алиса предоставляет свои доказательства.

(6) Боб предоставляет свои доказательства.

(7) Судья принимает решение на основании доказательств.

Различие используемых в этой книге понятий посредника и арбитра состоит в том, что участие арбитра происходит не всегда. Стороны обращаются к судье только при разногласиях. Если разногласий нет, судья не нужен.

Существуют арбитражные компьютерные протоколы. Они предполагают, что участвующие стороны честны, но при подозрении о возможном мошенничестве по существующему набору данных третья сторона, которой доверяют участники, сможет обнаружить факт мошенничества. Хороший арбитражный протокол позволяет арбитру установить и личность мошенника. Арбитражные протоколы обнаруживают, а не предупреждают мошенничество. Неотвратимость обнаружения выступает в качестве предупредительной меры, предотвращая мошенничество.

Самодостаточные протоколы

Самодостаточный протокол является лучшим типом протокола. Он полностью обеспечивает честность сторон (см. 1-й(в)). Для выполнения протокола не нужен ни посредник, не решающий споры арбитр. Само построение протокола обеспечивает отсутствие споров. Если одна из сторон попытается смошенничать, мошенничество будет немедленно обнаружено другой стороной, и протокол прекратит выполняться. Чего бы не пыталась добиться мошенничающая сторона, этому не суждено случиться.

В лучшем мире любой протокол должен быть самодостаточным, но, к несчастью, не существует самодостаточных протоколов для каждой ситуации.

Попытки вскрытия протоколов

Криптографические попытки взлома могут быть направлены против криптографических алгоритмов, используемых в протоколах, против криптографических методов, используемых для реализации алгоритмов и протоколов или непосредственно против протоколов. Поскольку в этом разделе книги обсуждаются именно протоколы, я предполагаю, что криптографические алгоритмы и методы безопасны, и рассматриваю только попытки вскрытия протоколов.

Люди могут использовать множество способов взломать протокол. Некоторые, не являясь участниками протокола, могут "подслушивать" какую-то часть или весь протокол. Это называется **пассивным вскрытием**, так как взломщик не воздействует на протокол. Все, что он может сделать - это проследить за протоколом и попытаться добыть информацию. Этот тип вскрытия соответствует вскрытию с использованием только шифротекста, обсуждавшемуся в разделе 1.1. Так как пассивные вскрытия трудно обнаружить, протоколы стремятся предотвращать, а не обнаруживать их. В этих протоколах роль злоумышленника будет играть Ева.

В другом случае взломщик может попытаться изменить протокол для собственной выгоды. Он может выдать себя за другого, ввести новые сообщения в протокол, заменить одно сообщение другим, повторно передать старые сообщения, разорвать канал связи или изменить хранящуюся в компьютере информацию. Такие действия называются активным вскрытием, так как они требуют активного вмешательства. Эти формы вскрытия зависят от вида сети.

Пассивные взломщики стараются получить информацию об участниках протокола. Они собирают сообщения, переданные различными сторонами, и пытаются криптоанализировать их. Попытки активного вскрытия, с другой стороны, преследуют более широкий набор целей. Взломщик может быть заинтересован в получении информации, ухудшении работы системы или получении несанкционированного доступа к ресурсам.

Активные вскрытия более серьезны, особенно в отношении протоколов, в которых стороны не обязательно доверяют друг другу. Взломщик не обязательно кто-то совсем посторонний, он может быть зарегистрированным пользователем системы и даже системным администратором. Может быть даже несколько активных взломщиков, работающих вместе. В этой книге роль злонамеренного активного взломщика будет играть Мэллори.

Взломщиком может быть и один из участников протокола. Он может обманывать, выполняя протокол, или вовсе не следовать правилам протокола. Такой взломщик называется **мошенником**. **Пассивные мошенники** выполняют правила протокола, но стараются получить больше информации, чем предусмотрено протоколом. **Активные мошенники** нарушают работу протокола, пытаясь смошенничать.

Очень трудно поддерживать безопасность протокола, если большинство его участников - активные мошенники, но иногда активное мошенничество может быть обнаружено законными участниками. Конечно, протоколы должны быть защищены и от пассивного мошенничества.

2.2 Передача информации с использованием симметричной криптографии

Как двум сторонам безопасно обмениваться информацией? Конечно же, шифрую свои сообщения. Посмотрим, что должно произойти, когда Алиса посылает шифрованное сообщение Бобу (полный протокол гораздо сложнее).

(1) Алиса и Боб выбирают систему шифрования.

(2) Алиса и Боб выбирают ключ.

(3) Алиса шифрует открытый текст своего сообщения с использованием алгоритма шифрования и ключа, п олучая шифрованное сообщение.

(4) Алиса посылает шифрованное сообщение Бобу.

(5) Боб дешифрирует шифротекст сообщения с использованием алгоритма шифрования и ключа, получая о ткрытый текст сообщения.

Что может Ева, находясь между Алисой и Бобом, узнать, подслушивая этот протокол ? Если она может подслушать только передачу на этапе (4), ей придется подвергнуть шифротекст криптоанализу . Это пассивное вскрытие представляет собой вскрытие с использованием только шифротекста, применяемые алгоритмы усто йчивы (насколько нам известно) по отношению к любым вычислительным мощностям, который может запол учить Ева для решения проблемы.

Ева, однако, не глупа. Она может также подслушать и этапы (1) и (2). Тогда ей станут известны алгоритм и ключ - также как и Бобу. Когда она перехватит сообщение на этапе (4), то ей останется только дешифровать его самостоятельно.

В хорошей криптосистеме безопасность полностью зависит от знания ключа и абсолютно не зависит от зн ания алгоритма. Именно поэтому управление ключами так важно в криптографии . Используя симметричный алгоритм, Алиса и Боб могут открыто выполнить этап (1), но этап (2) они должны сохранить в тайне . Ключ должен оставаться в секрете перед, после и в течение работы протокола - до тех пор, пока должно оставаться в тайне передаваемое сообщение - в противном случае сообщение тут же будет раскрыто . (О криптографии с открытыми ключами, решающей эту проблему иначе, рассказывается в разделе 2.5 .)

Мэллори, активный взломщик, может сделать кое-что другое. Он может попытаться нарушить линию связи не этапе (4), сделав так, что Алиса вообще не сможет передавать информацию Бобу . Мэллори также может перехватить сообщение Алисы и заменить его своим собственным . Если ему удалось узнать ключ (перехватив обмен информацией на этапе (2) или взломав криптосистему), он сможет зашифровать свое сообщение и отпр авить его Бобу вместо перехваченного, и Боб не сможет узнать, что сообщение отправлено не Алисой. Если Мэ ллори не знает ключа, он может только создать сообщение, превращающееся при дешифровке в бессмыслицу . Боб, считая, что сообщение отправлено Алисой, может решить, что либо у Алисы, либо в сети возникли серье зные проблемы.

А Алиса? Что она может сделать, чтобы испортить протокол? Она может передать копию ключа Еве, и тогда Ева сможет читать все, что говорит Боб, и напечатать его слова в *Нью-Йорк Таймс*. Это серьезно, но проблема не в протоколе. Алиса и так может передавать Еве любые открытые тексты, передаваемые с использованием протокола. Конечно, то же самое может сделать и Боб. Протокол предполагает, что Алиса и Боб доверяют друг другу. Итак, симметричным криптосистемам присущи следующие проблемы :

— Распределение ключей должно проводиться в секрете . Ключи столь же важны, как и все сообщения, з ашифрованные этими ключами, так как знание ключа позволяет раскрыть все сообщения . Для распространенных систем шифрования задача распределения ключей - серьезнейшая задача . Часто курьеры лично доставляют ключи по назначению.

— Если ключ скомпрометирован (украден, разгадан, выпытан, получен за взятку и т.д.), то Ева сможет расшифровать все сообщения, зашифрованные этим ключом . Она сможет также выступить в качестве одной из сторон и создавать ложные сообщения, дурача другую сторону .

— В предположении, что каждая пара пользователей сети использует отдельный ключ, общее число ключей быстро возрастает с ростом числа пользователей . Сеть из *n* пользователей требует *n{n - l)/2* ключей. Например, для общения 10 пользователей между собой нужно 45 различных ключей, для 100 пользователей потребуется 4950 ключей. Решение проблемы - в уменьшении числа пользователей, но это не всегда во зможно.

2.3 Однонаправленные функции

Понятие **однонаправленной функции** является центральным в криптографии с открытыми ключами . Не являясь протоколами непосредственно однонаправленные функции представляют собой краеугольный камень большинства протоколов, обсуждаемых в этой книге .

Однонаправленные функции относительно легко вычисляются, но инвертируются с большим трудом. То есть, зная *x* просто рассчитать *f(x)*, но по известному *f(x)* нелегко вычислить *x.* Здесь, "нелегко" означает, что для вычисления *x* по *f(x)* могут потребоваться миллионы лет, даже если над этой проблемой будут биться все компь ютеры мира.

Хорошим примером однонаправленной функции служит разбитая тарелка . Легко разбить тарелку на тысячу крошечных кусочков.. Однако, нелегко снова сложить тарелку из этих кусочков .

Это звучит красиво, но туманно и непонятно. Математически строгого доказательства существования однонаправленных функций нет, нет и реальных свидетельств возможности их построения [230, 530, 600, 661]. Несмотря на это, многие функции выглядят в точности как однонаправленные: мы можем рассчитать их и, до сих пор, не знаем простого способа инвертировать их. Например, в ограниченной окрестности легко вычислить x^2, но намного сложнее $x^{1/2}$. В оставшейся части раздела я собираюсь притвориться, что однонаправленные функции существуют. Мы поговорим об этом в еще разделе 11.2.

Итак, что же хорошего в однонаправленных функциях? Непосредственно их нельзя использовать для шифрования. Сообщение, зашифрованное однонаправленной функцией бесполезно - его невозможно дешифровать. (Упражнение: напишите на тарелке что-нибудь, разбейте тарелку на крошечные осколки и затем отдайте их приятелю. Попросите его прочитать сообщение. Посмотрите, как он будет озадачен однонаправленной функцией.) Для криптографии с открытыми ключами нам нужно что-то другое (хотя существуют и непосредственные криптографические применения однонаправленных функций - см. раздел 3.2).

Однонаправленная функция с люком - это особый тип однонаправленной функции, с секретной лазейкой. Ее легко вычислить в одном направлении и трудно - в обратном. Но если вам известен секрет, вы можете легко рассчитать и обратную функцию. То есть, легко вычислить *f(x)* по заданному *x*, но трудно по известному *f(x)* вычислить *x*. Однако, существует небольшая секретная информация, *y*, позволяющая, при знании *f(x)* и *y*, легко вычислить *x*.

В качестве хорошего примера однонаправленной функции с люком рассмотрим часы. Легко разобрать часы на сотни малюсеньких кусочков и трудно снова собрать из этих деталей работающие часы. Но, с секретной информацией - инструкцией по сборке - намного легче решить эту задачу.

2.4 Однонаправленные хэш-функции

У **однонаправленной хэш-функции** может быть множество имен: функция сжатия, функция сокращения contraction function, краткое изложение, характерный признак, криптографическая контрольная сумма, код целостности сообщения (message integrity check, MIC) и код обнаружения манипуляции (manipulation detection code, MDC). Как бы она не называлась эта функция является центральной в современной криптографии. Однонаправленные хэш-функции - это другая часть фундамента многих протоколов.

Хэш-функции, долгое время использующиеся в компьютерных науках, представляют собой функции, математические или иные, которые получают на вход строку переменной длины (называемую **прообразом**) и преобразуют ее в строку фиксированной, обычно меньшей, длины (называемую значением хэш-функции). В качестве простой хэш-функции можно рассматривать функцию, которая получает прообраз и возвращает байт, представляющий собой XOR всех входных байтов.

Смысл хэш-функции состоит в получении характерного признака прообраза - значения, по которому анализируются различные прообразы при решении обратной задачи. Так как обычно хэш-функция представляет собой соотношение "многие к одному", невозможно со всей определенностью сказать, что две строки совпадают, но их можно использовать, получая приемлемую оценку точности.

Однонаправленная хэш-функция - это хэш-функция, которая работает только в одном направлении: легко вычислить значение хэш-функции по прообразу, но трудно создать прообраз, значение хэш-функции которого равно заданной величине. Упоминавшиеся ранее хэш-функции, вообще говоря, не являются однонаправленными: задав конкретный байт, не представляет труда создать строку байтов, XOR которых дает заданное значение. С однонаправленной хэш-функцией такого не выйдет. Хорошей однонаправленной хэш-функцией является хэш-функция **без столкновений** - трудно создать два прообраза с одинаковым значением хэш-функции.

Хэш-функция является открытой, тайны ее расчета не существует. Безопасность однонаправленной хэш-функцией заключается именно в ее однонаправленности. У выхода нет видимой зависимости от входа. Изменение одного бита прообраза приводи к изменению, в среднем, половины битов значения хэш-функции. Вычислительно невозможно найти прообраз, соответствующий заданному значению хэш-функции.

Посмотрите на это как на способ получить характерные признаки файлов. Если вы хотите проверить, что у кого-то есть тот же файл, что и у вас, но вы не хотите, чтобы этот файл был передан вам, попросите послать вам значение хэш-функции. Если присланное значение хэш-функции совпадет с рассчитанным вами, то почти наверняка чужой файл совпадает с вашим. Это особенно полезно при финансовых транзакциях, когда вы не хотите где-то в сети превратить снятие со счета $100 в снятие $1000. В обычных условиях вы можете использовать однонаправленную хэш-функцию без ключа, так что кто угодно может проверить значение хэш-функции. Если нужно, чтобы проверить значение хэш-функции мог только один получатель, прочтите следующий раздел.

Коды проверки подлинности сообщения

Код проверки подлинности сообщения (message authentication code, MAC), известный также как код про-

верки подлинности данных (data authentication code, DAG), представляет собой однонаправленную хэш-функцию с добавлением секретного ключа (см. раздел 18.14). Значение хэш-функции является функцией и прообраза, и ключа. Теория остается той же, что и для хэш-функций, но только тот, кто знает ключ, может проверить значение хэш-функции. MAC можно создать с помощью хэш-функции или блочного алгоритма шифрования, существуют также и специализированные MAC.

2.5 Передача информации с использованием криптографии с открытыми ключами

Взгляните на симметричный алгоритм как на сейф. Ключ является комбинацией. Знающий комбинацию человек может открыть сейф, положить в него документ и снова закрыть. Кто-то другой при помощи той же комбинации может открыть сейф и забрать документ. Тем, кто не знает комбинации, придется научиться взламывать сейфы.

В 1976 году Уитфилд Диффи и Мартин Хеллман навсегда изменили эту парадигму криптографии [496]. (NSA заявило, что знало о такой возможности еще в 1966 году, но доказательств не представило.) Они описали **криптографию с открытыми ключами**, используя два различных ключа - один открытый и один закрытый. Определение закрытого ключа по открытому требует огромных вычислительных затрат. Кто угодно, используя открытый ключ может зашифровать сообщение, но не расшифровать его. Расшифровать сообщение может только владелец закрытого ключа. Это похоже на превращение криптографического сейфа в почтовый ящик. Шифрование с открытым ключом аналогично опусканию письма в почтовый ящик, любой может сделать это, опустив письмо в прорезь почтового ящика. Дешифрирование с закрытым ключом напоминает извлечение почты из почтового ящика. Обычно это гораздо сложнее - вам может понадобиться сварочный агрегат. Однако, если вы знаете секрет (у вас есть ключ от почтового ящика), вы без труда достанете вашу почту.

Математической основой процесса являются ранее обсуждавшиеся однонаправленные хэш-функции с люком. Шифрование выполняется в прямом направлении. Указания по шифрованию открыты, каждый может зашифровать сообщение. Дешифрирование выполняется в обратном направлении. Оно настолько трудоемко, что, не зная секрета, даже на компьютерах Cray за тысячи (и миллионы) лет невозможно расшифровать сообщение. Секретом, или люком, и служит закрытый ключ, он делает дешифрирование таким же простым, как и шифрование. Вот как, используя криптографию с открытыми ключами, Алиса может послать сообщение Бобу:

(1) Алиса и Боб согласовывают криптосистему с открытыми ключами.

(2) Боб посылает Алисе свой открытый ключ.

(3) Алиса шифрует свое сообщение и отправляет его Бобу.

(4) Боб расшифровывает сообщение Алисы с помощью своего закрытого ключа.

Обратите внимание, что криптография с открытыми ключами устраняет проблему распределения ключей, присущую симметричным криптосистемам. Раньше Алиса и Боб должны были тайно договориться о ключе. Алиса могла выбрать любой ключ, но ей нужно было передать его Бобу. Она могла сделать это заранее, но это требует от нее определенной предусмотрительности. Она могла бы послать ключ с секретным курьером, но для этого нужно время. Криптография с открытыми ключами все упрощает. Алиса может отправить Бобу секретное сообщение без каких-либо предварительных действий. У Евы, подслушивающей абсолютно все, есть открытый ключ Боба и сообщение, зашифрованное этим ключом, но она не сможет получить ни закрытый ключ Боба, ни текст сообщения.

Обычно целая сеть пользователей согласовывает используемую криптосистему. У каждого из них есть открытый и закрытый ключ, открытые ключи помещаются в общедоступной базе данных. Теперь протокол выглядит еще проще:

(1) Алиса извлекает открытый ключ Боба из базы данных.

(2) Алиса шифрует свое сообщение с помощью открытого ключа Боба и посылает его Бобу.

(3) Боб расшифровывает сообщение Алисы с помощью своего закрытого ключа.

В первом протоколе Боб должен был послать Алисе ее открытый ключ прежде, чем она могла отправить ему сообщение. Второй протокол больше похож на обычную почту. Боб не участвует в протоколе до тех пор, пока он не начнет читать сообщение.

Смешанные криптосистемы

Первые алгоритмы с открытым ключом стали известны в то же время, когда проходило DES обсуждение как предполагаемого стандарта. Это привело к известной партизанщине в криптографическом сообществе. Как это описывал Диффи [494]:

Прекрасные криптосистемы с открытым ключом, обсуждаемые в популярной и научной печати, тем не менее, не нашли соответствующего отклика среди криптографических чиновников. В том же году, когда была открыта криптография с открытыми ключами, Агентство национальной безопасности (NSA) предложило удобную криптографическую систему, разработанную фирмой IBM, в качестве федерального *Стандарта шифрования данных* (Data Encryption Standard, DES). Марти Хеллман и я критиковали это предложение из-за недостаточной длины ключа, но производители подготовились поддержать стандарт, и наша критика была воспринята многими как попытка помешать введению стандарта ради продвижения нашей собственной работы. Криптография с открытым ключом, в свою очередь, также подвергалась критике в популярной литературе [1125] и технических статьях [849, 1159], словно это был конкурирующий продукт, а не недавнее научное открытие. Это, однако, не помешало NSA объявить о своих заслугах в этой области. Его директор в одной из статей *Encyclopedia Britannica* [1461] указал, что "двухключевая криптография была открыта в Агентстве на десять лет раньше", хотя доказательства этого утверждения не были публично представлены.

В реальном мире алгоритмы с открытыми ключами не заменяют симметричные алгоритмы и используются не для шифрования сообщений, а для шифрования ключей по следующим двум причинам:

1. Алгоритмы с открытыми ключами работают медленно. Симметричные алгоритмы по крайней мере в 1000 раз быстрее, чем алгоритмы с открытыми ключами. Да, компьютеры становятся все быстрее и быстрее и лет через 15 криптография с открытыми ключами достигнет скоростей, сравнимых с сегодняшней скоростью симметричной криптографии. Но требования к объему передаваемой информации также возрастают, и всегда будет требоваться шифровать данные быстрее, чем это сможет сделать криптография с открытыми ключами.
2. Криптосистемы с открытыми ключами уязвимы по отношению к вскрытию с выбранным открытым текстом. Если $C = E(P)$, где P - открытый текст из n возможных открытых текстов, то криптоаналитику нужно только зашифровать все n возможных открытых текстов и сравнить результаты с C (помните, ключ шифрования общедоступен). Он не сможет раскрыть ключ дешифрирования, но он сможет определить P.

Вскрытие с выбранным открытым текстом может быть особенно эффективным, если число возможных шифрованных сообщений относительно мало. Например, если P - это денежная сумма в долларах, меньшая чем $1000000, то такое вскрытие сработает, криптоаналитик переберет весь миллион значений. (Эта проблема решается с помощью вероятностного шифрования, см. раздел 23.15.) Даже если P не так хорошо определено, такое вскрытие может быть очень эффективно. Полезным может быть простое знание, что шифротекст не соответствует конкретному открытому тексту. Симметричные криптосистемы не чувствительны к вскрытиям такого типа, так как криптоаналитик не может выполнить тестовых дешифровок с неизвестным ключом.

В большинстве реализаций криптография с открытыми ключами используется для засекречивания и распространения сеансовых ключей, которые используются симметричными алгоритмами для закрытия потока сообщений [879]. Иногда такие реализации называются смешанными (гибридными) криптосистемами

(1) Боб посылает Алисе свой открытый ключ

(2) Алиса создает случайный сеансовый ключ, шифрует его с помощью открытого ключа Боба и передает его Бобу.

$E_B(K)$

(3) Боб расшифровывает сообщение Алисы, используя свой закрытый ключ, для получения сеансового ключа.

$D_B(E_B(K))=K$

(4) Оба участника шифруют свои сообщения с помощью одного сеансового ключа.

Использование криптографии с открытыми ключами для распределения ключей решает очень важную проблему распределения ключей. В симметричной криптографии ключ шифрования данных, если он не используется, валяется без дела. Если Ева заполучит его, она сможет расшифровать все закрытые этим ключом сообщения. С помощью приведенного протокола при необходимости зашифровать сообщения создается сеансовый ключ, который уничтожается по окончании сеанса связи. Это значительно уменьшает риск компрометации сеансового ключа. Конечно, к компрометации чувствителен и закрытый ключ, но риска значительно меньше, так как в течение сеанса этот ключ используется только один раз для шифрования сеансового ключа. Подробно связанные с этим вопросы обсуждаются в разделе 3.1.

Головоломки Меркла

Ральф Меркл (Ralph Merkle) изобрел первую схему криптографии с открытыми ключами. В 1974 году он записался на курс по компьютерной безопасности в Калифорнийском университете, Беркли, который вел Ланс Хоффман (Lance Hoffman). Темой его курсовой работы, поданной раньше срока, была "Безопасная передача данных по небезопасным каналам" [1064]. Хоффман не понял предложения Меркла, и в конце концов Меркл прекратил занятия. Он продолжал работать над проблемой несмотря на продолжающееся непонимание его результатов.

Техника Меркла основывалась на головоломках ("puzzle"), которые отправителю и получателю решить лег-

че чем злоумышленнику. Вот как Алиса может послать шифрованное сообщение Бобу, не обмениваясь с ним ключом до того.

(1) Боб создает 2^{20} (другими словами, больше миллиона) сообщений типа: "Это головоломка номер x. Это секретный ключ номер y.", где x - случайное число, а y - случайный секретный ключ. И x, и y отличаются в каждом сообщении. Используя симметричный алгоритм, он шифрует каждое сообщение своим 20 би т-ным ключом и все их отправляет Алисе.

(2) Алиса выбирает одно сообщение и приступает к вскрытию **грубой силой**, пытаясь получить открытый текст. Эта работа является объемной, но не невозможной.

(3) Алиса шифрует свое секретное сообщение при помощи некоторого симметричного алгоритма полученным ею ключом и посылает это сообщение Бобу вместе с x.

(4) Боб знает, какой секретный ключ y он использовал в сообщении x, следовательно он может расшифровать сообщение Алисы.

Ева может взломать эту систему, но ей придется выполнить гораздо больше работы чем Алисе и Бобу . Для раскрытия сообщения на этапе (3) она должна будет вскрыть грубой силой каждое из 2^{20} сообщений, отправленных Бобом на этапе (1). Сложность этого вскрытия составит 2^{40}. Значения x также не помогут Еве, ведь они на этапе (1) присвоены случайным образом. В общем случае, вычислительные затраты Евы будут равны возв е-денным в квадрат вычислительным затратам Алисы.

Это выигрыш (n по отношению к n^2) невелик по криптографическим стандартам, но при определенных усл о-виях может быть достаточен. Если Алиса и Боб могут проверить десять тысяч ключей в секунду, каждому из них потребуется минута для выполнения своих действий и еще одна минута для передачи головоломок от Боба к Алисе по линии связи 1.544 Мбит/с. Если вычислительные возможности Евы сравнимы с приведенными, ей потребуется около года для взлома системы. Другие алгоритмы еще более устойчивы к вскрытию .

2.6 Цифровые подписи

Рукописные подписи издавна используются как доказательство авторства документа или, по крайней мере, согласия с ним. Что же так притягательно в подписи [1392]?

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

В действительности, ни одно из этих утверждений не является полностью справедливым . Подписи можно подделать, свести с одного листа бумаги на другой, документы могут быть изменены после подписания . Однако, мы миримся с этими проблемами из-за того, что мошенничество затруднительно и может быть обнар ужено.

Хотелось бы реализовать что-нибудь подобное и на компьютерах, но есть ряд проблем . Во первых, компьютерные файлы скопировать не просто, а очень просто. Даже если подпись человека трудно подделать (например, графическое изображение рукописной подписи), можно легко вырезать правильную подпись из одного докуме н-та и вставить в другой. Простое наличие такой подписи ничего не означает. Во вторых, компьютерные файлы очень легко можно изменить после того, как они подписаны, не оставляя ни малейшего следа изменения .

Подпись документа с помощью симметричных криптосистем и посредника

Алиса хочет подписать цифровое сообщение и отправить его Бобу. Она может это сделать с помощью Тре н-та и симметричной криптосистемы.

Трент - это обладающий властью посредник, которому доверяют . Он может связываться и с Алисой, и с Б о-бом (и со всеми другими желающими подписывать цифровые документы). Он выдает секретный ключ, K_A, Али-се и другой секретный ключ, K_B, - Бобу. Эти ключи определяются задолго до начала действия протокола и могут быть использованы многократно для многих подписей.

(1) Алиса шифрует свое сообщение Бобу ключом K_A и посылает его Тренту.

(2) Трент, зная ключ K_A, расшифровывает сообщение.

(3) Трент добавляет к расшифрованному сообщению утверждение, что он получил это сообщение от Алисы, и шифрует это новое сообщение ключом K_B.

(4) Трент посылает новое сообщение Бобу.

(5) Боб расшифровывает сообщение ключом K_B. Он может прочитать и сообщение Алисы, и подтверждение Трента, что сообщение отправлено именно Алисой.

Откуда Трент узнает, что сообщение пришло именно от Алисы, а не от какого-то самозванца ? Он делает этот вывод из шифрования сообщения.

Также ли это хорошо, как подпись на бумаге? Посмотрим на требуемые свойства:

1. Эта подпись достоверна. Трент - это заслуживающий доверия посредник, и Трент знает, что сообщ е-ние получено от Алисы. Подтверждение Трента служит доказательством для Боба.
2. Эта подпись неподдельна. Только Алиса (и Трент, но ему все верят) знает K_A, поэтому только Алиса могла послать Тренту сообщение, зашифрованное ключом K_A. Если кто-нибудь попытается выдать с е-бя за Алису, Трент сразу заметит это на этапе (2) и не заверит подлинность.
3. Эту подпись нельзя использовать повторно. Если Боб попытается взять подтверждение Трента и пр и-соединить его к другому сообщению, Алиса закричит "Караул!" Посредник (Трент или кто-то совсем другой, имеющий доступ к той же информации) попросит Боба предъявить его сообщение и шифр о-ванное сообщение Алисы. Затем посредник зашифрует сообщение ключом K_A и увидит, что оно не со-ответствует шифрованному сообщению, переданному Бобом . Боб, конечно же, не сможет создать пр а-вильное шифрованное сообщение, потому что он не знает кл юча K_A.
4. Подписанный документ нельзя изменить. Если Боб попытается, получив документ, изменить его, Трент обнаружит мошенничество уже описанным способом.
5. От подписи невозможно отказаться. Если впоследствии Алиса заявит, что она никогда не посылала сообщение, подтверждение Трента докажет обратное. Помните, все доверяют Тренту, все, сказанное им - истина.

Если Боб захочет показать Кэрол документ, подписанный Алисой, он не сможет раскрыть ей свой секретный ключ. Ему придется снова о братиться к Тренту:

(1) Боб берет сообщение и утверждение Трента, что сообщение получено от Алисы, шифрует их ключом K_B и посылает обратно Тренту.

(2) Трент расшифровывает полученный пакет с помощью ключа K_B.

(3) Трент проверяет свою базу данных и подтверждает, что отправителем оригинального сообщения была Алиса.

(4) Трент шифрует полученный от Боба пакет ключом K_C, который он выделил для Кэрол, и посылает Кэрол шифрованный пакет.

(5) Трент расшифровывает полученный пакет с помощью ключа K_C. Теперь она может прочитать и сообщ е-ние, и подтверждение Трента, что сообщение отправлено Алисой.

Эти протоколы работают, но они требуют от Трента немалых затрат времени. Он должен целыми днями расшифровывать и шифровать сообщения, посредничая между каждой парой людей, которые хотят обмен и-ваться подписанными документами. Он должен хранить сообщения в базе данных (хотя этого можно избежать, посылая получателю копию шифрованного сообщения отправителя). Он будет узким местом любой системы связи, даже если он - просто бесчувственная компьютерная программа. .

Такого посредника как Трент, которому будут доверять все корреспонденты, тяжело найти и тяжело сохр а-нить. Трент должен быть непогрешим, если он сделает хотя бы одну ошибку на миллион подписей, никто не будет верить ему. Трент должен быть абсолютно безопасен . Если его база данных с секретными ключами когда-нибудь раскроется, или кто-нибудь сможет перепрограммировать его, все подписи станут бесполезными . Поя-вятся документы будто бы подписанные годы назад . Это приведет к хаосу. Правительства падут, и станет пр а-вить анархия. Такая схема теоретически может работать, но она недостаточно хороша для практического пр и-менения.

Деревья цифровых подписей

Ральф Меркл предложил систему цифровых подписей, основанную на криптографии с секретным ключом, создающей бесконечное количество одноразовых подписей, используя древовидную структуру [1067,1068]. Ос-новной идеей этой схемы является поместить корень дерева в некий открытый файл , удостоверяя его таким о б-

разом. Корень подписывает одно сообщение и удостоверяет подузлы дерева . Каждый из этих узлов подписывает одно сообщение и удостоверяет свои подузлы, и так далее .

Подпись документа с помощью криптографии с открытыми ключами

Существуют алгоритмы с открытыми ключами, которые можно использовать для цифровых подписей. В н е-которых алгоритмах - примером является RSA (см. раздел 19.3) - для шифрования может быть использован или открытый, или закрытый ключ. Зашифруйте документ своим закрытым ключом, и вы получите надежную ци ф-ровую подпись. В других случаях - примером является DSA (см. раздел 20.1) - для цифровых подписей исполь-зуется отдельный алгоритм, который невозможно использовать для шифрования . Эта идея впервые была изо-бретена Диффи и Хеллманом [496] и в дальнейшем была расширена и углублена в других работах [1282, 1328, 1024, 1283, 426]. Хороший обзор этой области приведен в [1099]. Основной протокол прост:

(1) Алиса шифрует документ своим закрытым ключом, таким образом подписывая его.

(2) Алиса посылает подписанный документ Бобу.

(3) Боб расшифровывает документ, используя открытый ключ Алисы, таким образом проверяя подпись.

Этот протокол гораздо лучше предыдущего. Трент не нужен ни для подписи документов, ни для ее проверки. (Он нужен для подтверждения, что открытый ключ принадлежит именно Алисе .) Трент не нужен сторонам даже для разрешения споров: Если Боб не смог осуществить этап (3), то он знает, что подпись неправильна. Такая подпись соответствует всем требованиям:

1. Эта подпись достоверна. Когда Боб расшифровывает сообщение с помощью открытого ключа Алисы, он знает что она подписала это сообщение.
2. Эта подпись неподдельна. Только Алиса знает свой закрытый ключ.
3. Эту подпись нельзя использовать повторно. Подпись является функцией документа и не может быть перенесена на другой документ.
4. Подписанный документ нельзя изменить. После любого изменения документа подпись не сможет больше подтверждаться открытым ключом Алисы.
5. От подписи невозможно отказаться. Бобу не требуется помощь Алисы при проверке ее подписи.

Подпись документа и метки времени

На самом деле, при определенных условиях Боб сможет смошенничать . Он может повторно использовать документ и подпись совместно. Это не имеет значения, если Алиса подписала контракт (одной копией подпи-санного контракта больше, одной меньше), но что если Алиса поставила цифровую подпись под чеком?

Предположим, что Алиса послала Бобу подписанный чек на $100. Боб отнес чек в банк, который проверил подпись и перевел деньги с одного счета на другой. Боб, выступающий в роли жулика, сохранил копию эле к-тронного чека. На следующей неделе он снова отнес его в этот или другой банк . Банк подтвердил подпись и пе-ревел деньги с одного счета на другой. Если Алиса не проверяет свою чековую книжку, Боб сможет продел ы-вать это годами.

Поэтому в цифровые подписи часто включают метки времени . Дата и время подписания документа доба в-ляются к документу и подписываются вместе со всем содержанием сообщения . Банк сохраняет эту метку врем е-ни в базе данных. Теперь, если Боб попытается получить наличные по чеку Алисы во второй раз, банк проверит метку времени по своей базе данных. Так как банк уже оплатил чек Алисы с той же меткой времени, то будет вызвана полиция. Затем Боб проведет лет 15 в тюрьме Ливенворт, изучая криптографические протоколы .

Подпись документа с помощью криптографии с открытыми ключами и однонаправленных хэш-функций

На практике алгоритмы с открытыми ключами часто недостаточно эффективны для подписи больших док у-ментов. Для экономии времени протоколы цифровой подписи нередко используют вместе с однонаправленными хэш-функциями [432, 433]. Алиса подписывает не документ, а значение хэш-функции для данного документа . В этом протоколе однонаправленная хэш-функция и алгоритм цифровой подписи согласовываются заранее .

(1) Алиса получает значение однонаправленной хэш-функции для документа.

(2) Алиса шифрует это значение своим закрытым ключом, таким образом подписывая документ.

(3) Алиса посылает Бобу документ и подписанное значение хэш-функции.

(4) Боб получает значение однонаправленной хэш-функции для документа, присланного Алисой. Затем, и с-пользуя алгоритм цифровой подписи, он расшифровывает подписанное значение хэш-функции с помощью

открытого ключа Алисы. Если подписанное значение хэш-функции совпадает с рассчитанным, подпись правильна.

Скорость заметно возрастает и, так как вероятность получить для двух различных документов одинаковое 160-битное значение хэш-функции составляет только один шанс из 2^{160}, можно безопасно приравнять подпись значения хэш-функции и подпись документа. Должна использоваться только однонаправленная хэш-функция, иначе создать разные документы с одним и тем же значением хэш-функции нетрудно, и подпись одного документа приведет к ошибочной подписи сразу многих документов.

У протокола есть и другие выгоды. Во первых, подпись может быть отделена от документа. Во вторых, значительно уменьшаются требования к объему памяти получателя, в котором хранятся документы и подписи. Архивная система может использовать этот протокол для подтверждения существования документов, не храня их содержания. В центральной базе данных могут храниться лишь значения хэш-функции для файлов. Вовсе не нужно просматривать файлы, пользователи помещают свои значения хэш-функции в базу данных, а база данных хранит эти значения, помечая их временем получения документа. Если в будущем возникнет какое-нибудь разногласие по поводу автора и времени создания документа, база данных сможет разрешить его при помощи хранящегося в ней значения хэш-функции. Подобная система имеет большое значение при хранении секретной информации: Алиса может подписать документ и сохранить его в секрете. Ей понадобится опубликовать документ, только если она захочет доказать свое авторство. (См. раздел 4.1).

Алгоритмы и терминология

Существует множество алгоритмов цифровой подписи. Все они представляют собой алгоритмы с открытыми ключами с закрытой частью для подписи документов и с открытой - для проверки подписи. Иногда процесс подписи называют **шифрованием с закрытым ключом**, а процесс проверки подписи - **дешифрированием с открытым ключом**. Это может ввести в заблуждение, являясь справедливым только для одного алгоритма, RSA. У других алгоритмов - другие реализации. Например, использование однонаправленных хэш-функций и меток времени иногда приводит к появлению дополнительных этапов при подписании и проверке подписи. Многие алгоритмы можно использовать для цифровой подписи, но нельзя для шифрования.

В общем случае я буду ссылаться на процессы подписи и проверки, не вдаваясь в подробности алгоритмов. Подпись сообщения с закрытым ключом K будет обозначаться как:

$S_K(M)$

а проверка подписи с помощью соответствующего открытого ключа как:

$V_K(M)$

Строку битов, присоединенную к документу после его подписания (в предыдущем примере, значение однонаправленной хэш-функции документа, зашифрованное зарытым ключом), будем называть **цифровой подписью** или просто **подписью**. Весь протокол, с помощью которого получатель сообщения проверяет личность отправителя и целостность сообщения, называется удостоверением подлинности. Более подробно эти протоколы рассматриваются в разделе 3.2.

Несколько подписей

Как Алисе и Бобу одновременно подписать один и тот же документ? В отсутствие однонаправленных хэш-функций существует две возможности. Алиса и Боб могут подписать различные копии одного и того же документа. Полученное сообщение будет в два раза длиннее первоначального документа. Или Алиса подписывает документ, а затем Боб подписывает подпись Алисы. Этот способ работает, но проверить подпись Алисы, не проверяя при этом подписи Боба, невозможно.

С помощью однонаправленных реализовать несколько подписей просто:

(1) Алиса подписывает значение хэш-функции документа.

(2) Боб подписывает значение хэш-функции документа.

(3) Боб посылает свою подпись Алисе.

(4) Алиса посылает Кэрол документ, свою подпись и подпись Боба.

(5) Кэрол проверяет подписи Алисы и Боба.

Алиса и Боб могут выполнить этапа (1) и (2) или параллельно, или последовательно. На этапе (5) Кэрол может проверить любую подпись независимо от другой.

Невозможность отказаться от цифровой подписи

Алиса может смошенничать с цифровыми подписями, и с этим ничего нельзя поделать. Она может подписать документ и затем утверждать, что она этого не делала. Сначала она, как обычно, подписывает письмо. Затем она анонимно раскрывает свой закрытый ключ или теряет в людном месте. Теперь Алиса утверждает, что ее подпись была скомпрометирована и использована кем-то другим, выдающим себя за нее. Она дезавуирует свою подпись под всеми документами, подписанными с помощью этого закрытого ключа. Это называется отказ от подписи.

Метки времени могут снизить эффект такого мошенничества, но Алиса всегда может заявить, что ее ключ был скомпрометирован раньше. Если Алиса правильно рассчитает время, она сможет подписать документ и затем успешно заявить, что она этого не делала. Поэтому так много говорится о хранении закрытых ключей в надежных местах - чтобы Алиса не могла добраться до своего и злоупотребить им.

Хотя с подобным злоупотреблением ничего нельзя сделать, можно предпринять некоторые действия, гарантирующие то, что старые подписи не будут признаны недостоверными из-за разногласий по новым подписям. (Например, Алиса может "потерять" свой ключ, чтобы не платить Бобу за подержанную машину, которую он вчера ей продал и, в результате, сделает недействительным свой банковский счет.) Получателю нужно проставлять метки времени для полученных документов [453]. Общая схема протокола приведена в [2, 8]:

(1) Алиса подписывает сообщение.

(2) Алиса создает заголовок, содержащий некоторую идентификационную информацию. Она присоединяет к заголовку подписанное сообщение, подписывает все вместе и посылает Тренту.

(3) Трент проверяет внешнюю подпись и подтверждает идентификационную информацию. Он добавляет метку времени к подписанному сообщению Алисы и идентификационной информации. Затем он подписывает все вместе и посылает пакет Алисе и Бобу.

(4) Боб проверяет подпись Трента, идентификационную информацию и подпись Алисы.

(5) Алиса проверяет сообщение, которое Трент послал Бобу. Если она не признает свое авторство, она быстро заявляет об этом.

В другой схеме Трент используется в качестве арбитра [209]. Получив подписанное сообщение, Боб посылает копию Тренту для проверки. Трент может подтвердить подпись Алисы.

Использование цифровых подписей

Одним из самых ранних предложенных применений цифровых подписей было упрощение проверки соблюдения договоров о ядерных испытаниях [1454, 1467]. Соединенные Штаты и Советский Союз (кто-нибудь помнит Советский Союз?) разрешили друг другу разместить на чужой территории сейсмографы для слежения за ядерными испытаниями. Проблема была в том, что каждая из сторон должна была уверена в том, что другая сторона не подделала данные этих сейсмографов. Одновременно, другая сторона должна была быть уверена, что эти датчики посылают только ту информацию, которая нужна для слежения за ядерными испытаниями.

Метод условного удостоверения подлинности может решить первую проблему, но только цифровые подписи могут решить обе проблемы. Сторона, на территории которой стоит сейсмограф, может прочесть, но не изменить данные сейсмографа, а следящая сторона знает, что данные не были подделаны.

2.7 Цифровые подписи и шифрование

Объединив цифровые подписи и криптографию с открытыми ключами, мы разрабатываем протокол, комбинирующий безопасность шифрования и достоверность цифровых подписей. Сравните с письмом от вашей маиы. Подпись удостоверяет авторство а конверт обеспечивает тайну.

(1) Алиса подписывает сообщение с помощью своего закрытого ключа.

$S_A(M)$

(2) Алиса шифрует подписанное сообщение открытым ключом Боба и посылает его Бобу.

$E_B(S_A(M))$

(3) Боб расшифровывает сообщение с помощью своего закрытого ключа.

$D_B(E_B(S_A(M))) = S_A(M)$

(4) Боб проверяет подпись с помощью открытого ключа Алисы и восстанавливает сообщение.

$V_A(S_A(M)) = M$

Подпись перед шифрованием выглядит естественно. Когда Алиса пишет письмо, она подписывает его и затем кладет в конверт. Если она положит письмо в конверт неподписанным, то Боб может забеспокоиться, вдруг письмо было тайно подменено. Если Боб покажет Кэрол письмо Алисы и конверт, Кэрол может обвинить Боба, что он врет о том, какое письмо в каком конверте пришло.

В электронной корреспонденции точно также является разумным использование подписи перед шифрованием [48]. Это не только более безопасно - враг не сможет удалить подпись из шифрованного сообщения и добавить свою собственную - но существуют и юридические соображения: если подписываемый текст не виден подписывающему, когда он ставит подпись, то юридическая сила подписи невелика [1312]. Существуют также некоторые криптографические способы вскрытия такой последовательности действий, использующей подписи RSA (см. раздел 19.3).

Для Алисы не существует причин использовать одну пару ключей - открытый/закрытый - для шифрования и подписи. У нее может быть две пары ключей: одна для шифрования и одна для подписи. У такого разделения есть свои преимущества: Алиса может передать свой ключ шифрования полиции, не компрометируя свою подпись, один ключ может быть условно передан (см. раздел 4.13), не влияя на другой. У ключей могут быть различные длины и сроки действия.

Конечно же, для предотвращения повторного использования сообщений с этим протоколом должны быть использованы метки времени. Метки времени также могут защитить от других возможных ловушек, пример одной из которых приведен ниже.

Возвращение сообщения при приеме

Рассмотрим реализацию этого протокола с дополнительной возможностью подтверждения сообщений - получив сообщение, Боб обязательно возвращает подтверждение приема.

(1) Алиса подписывает сообщение с помощью своего закрытого ключа, шифрует подписанное сообщение открытым ключом Боба и посылает его Бобу.

$E_B(S_A(M))$

(2) Боб расшифровывает сообщение с помощью своего закрытого ключа, проверяет подпись с помощью открытого ключа Алисы и восстанавливает сообщение.

$V_A(D_B(E_B(S_A(M)))) = M$

(3) Боб подписывает сообщение с помощью своего закрытого ключа, шифрует подписанное сообщение открытым ключом Алисы и посылает его Алисе обратно.

$E_A(S_B(M))$

(4) Алиса расшифровывает сообщение с помощью своего закрытого ключа и проверяет подпись с помощью открытого ключа Боба. Если полученное сообщение совпадает с отправленным, она знает, что Боб получил правильное сообщение.

Если для шифрования и проверки цифровой подписи используется один и тот же алгоритм, то существует возможность вскрытия [506]. В таких случаях операция цифровой подписи - противоположность операции шифрования: $V_X = E_X$ и $S_X = D_X$.

Пусть Мэллори - зарегистрированный пользователь со своей парой ключей: открытым и закрытым. Теперь посмотрим, как он сможет читать почту Боба. Сначала он запишет сообщение Алисы Бобу - этап (1). Затем, немного погодя, он пошлет это сообщение Бобу, утверждая, что оно отправлено самим Мэллори. Боб, думая, что это обычное сообщение от Мэллори, дешифрирует это сообщение своим закрытым ключом и пытается проверить подпись Мэллори, дешифрируя ее с помощью открытого ключа Мэллори. В результате получается полная чепуха:

$E_A(D_B(E_B(D_A(M)))) = E_M(D_A(M))$

Даже в этом случае, следуя протоколу, Боб посылает Мэллори полученное сообщение:

$E_M(D_B(E_M(D_A(M))))$

Теперь Мэллори остается только расшифровать сообщение с помощью своего закрытого ключа, зашифровать его открытым ключом Боба, расшифровать снова с помощью своего закрытого ключа и зашифровать открытым ключом Алисы. *Voilà!* Мэллори получает *M*.

Отнюдь не глупо предположить, что Боб может автоматически посылать Мэллори квитанцию. Этот протокол, например, может быть встроен в его коммуникационное программное обеспечение и посылать квитанции автоматически. Именно готовность сообщить о приеме чепухи и нарушает безопасность. Если Боб проверит сообщение на осмысленность перед отправкой квитанции, он сможет избежать таких проблем с безопасностью.

Существуют модернизации этого способа вскрытия, предполагающие, что Мэллори пошлет Бобу сообщение, отличное от того, которое он желает перехватить. Никогда не подписывайте произвольных сообщений от других людей и не передавайте результаты дешифровки произвольных сообщений иным людям.

Обнаружение вскрытия, основанного на возвращении сообщения

Только что описанное вскрытие работает потому, что операция шифрования совпадает с операцией проверки подписи, а операция дешифрирования - с операцией подписи. Операции шифрования и цифровой подписи в безопасном протоколе должны хотя бы слегка отличаться. Проблему решает использование различных ключей для каждой операции, или использование для каждой операции различных алгоритмов, или применение меток времени, которые делают различными принятое и отправляемое сообщения, или цифровая подпись с помощью однонаправленной хэш-функции (см. раздел 2.6). Тогда, в общем случае, следующий протокол, использующий алгоритм с открытым ключом, является безопасным:

(1) Алиса подписывает сообщение.

(2) Алиса шифрует подписанное сообщение открытым ключом Боба (используя алгоритм, отличающийся от алгоритма цифровой подписи) и посылает его Бобу.

(3) Боб расшифровывает сообщение с помощью своего закрытого ключа

(4) Боб проверяет подпись Алисы.

Вскрытия криптографии с открытыми ключами

Во всех подобных протоколах криптографии с открытыми ключами я не рассказал, как Алиса получает открытый ключ Боба. Подробно этот вопрос описан в разделе 3.1, но о нем стоит упомянуть и здесь.

Проще всего узнать чей-то открытый ключ, считав его откуда-то из безопасной базы данных. Эта база данных должна быть общедоступна, чтобы каждый мог получить нужный ему ключ. База данных должна быть защищена от несанкционированной записи, в противном случае Мэллори сможет подменить открытый ключ Боба. После этого Боб уже не сумеет читать адресованные ему сообщения, зато это сможет сделать Мэллори.

Даже если открытые ключи хранятся в надежной базе данных, Мэллори может подменить их при передаче. Чтобы воспрепятствовать этому, Трент должен подписывать каждый открытый ключ, используя свой собственный закрытый ключ. Трента, который действует подобным образом, часто называют **Органом сертификации ключей** или **Центром распределения ключей** (Key Distribution Center, KDC). На практике KDC подписывает сложное сообщение, состоящее из имени пользователя, его открытого ключа и другой информации о пользователе. Это подписанное сложное сообщение и хранится в базе данных KDC. Когда Алиса получает ключ Боба, она проверяет подпись KDC, удостоверяясь в правильности ключа.

При окончательном анализе видно, что и это только затрудняет, но не делает невозможным мошенничество Мэллори. Алиса же должна откуда-то получить открытый ключ KDC. Мэллори нужно подменить этот ключ своим открытым ключом, испортить базу данных и заменить правильные ключи своими (подписанными его закрытым ключом, как если бы он и был KDC), и его дело сделано. Но, даже подписи на бумаге могут быть подделаны, если Мэллори всерьез возьмется за дело. Подробно обмен ключами рассматривается в разделе 3.1.

2.8. Генерация случайных и псевдослучайных последовательностей

Почему даже в книге по криптографии снова эти докучливые рассуждения о генерации случайных чисел? Генератор случайных чисел встроен в каждый компилятор, обычный вызов функции. Почему бы не использовать его? К сожалению, эти генераторы случайных чисел почти наверняка недостаточно безопасны для криптографии и, возможно, даже не совсем случайны. Большинство из них весьма плохи.

Генераторы случайных чисел на самом деле совсем не случайны, потому что им и не нужно быть такими. Для большинства приложений, например, компьютерных игр, требуется так мало случайных чисел, что их неслучайность вряд ли будет заметна. Однако, криптография очень чувствительна к свойствам генераторов случайных чисел. Примените плохой генератор, и у вас появятся таинственные корреляции и странные результаты [1231, 1238]. Если ваша безопасность зависит от генератора случайных чисел, таинственные корреляции и странные результаты являются абсолютно не тем, чего бы вы желали добиться.

Проблема в том, что генератор случайных чисел не создает случайной последовательности. Он, возможно, не выдает ничего даже отдаленно напоминающего случайную последовательность. Конечно, невозможно создавать на компьютере что-то по настоящему случайное. Дональд Кнут приписывал фон Нейману следующие слова: "Каждый, кто занимается арифметическими методами получения случайных чисел, определенно грешит" [863]. Компьютеры - это детерминированные бестии: закладывается известный материал, выполняются полностью предсказуемые действия, и что-то отличное выползает с другого конца. Подача одного и того же на вход в двух различных случаях приведет к одному и тому же результату. Заложите одинаковые исходные данные в два

идентичных компьютера, и оба они подсчитают одно и то же. Компьютер может находиться только в ограниченном числе состояний (очень большом, но все же ограниченном), и выдаваемый результат всегда будет строго определяться исходными данными и текущим состоянием компьютера. Это значит, что любой генератор случайных чисел на компьютере (по меньшей мере, на конечном автомате), по определению, периодичен. А все, что периодично, по определению, предсказуемо. А все, что предсказуемо, не может быть случайным. Для настоящего генератора случайных чисел нужно подавать на вход что-нибудь случайное, компьютер же не может обеспечить это требование.

Псевдослучайные последовательности

Лучшее, что может сделать компьютер - это **генератор псевдослучайных последовательностей** Что это такое? Многие пытались дать его формальное определение, но я уклонюсь от этого. Псевдослучайная последовательность - это что-то, выглядящее как случайное. Период последовательности должен быть достаточно велик, поэтому конечная последовательность разумной длины - которая в действительности и используется - не периодична. Если вам нужен миллиард случайных бит, не пользуйтесь генератором последовательности, повторяющейся каждые шестнадцать тысяч бит. Эти относительно короткие непериодические подпоследовательности должны быть, насколько это возможно, неотличимы от случайных последовательностей. Например, в них должно быть примерно одинаковое количество единиц и нулей, около половины серий (последовательностей одинаковых бит) должны быть единичной длины, четверть - состоять из двух бит, восьмая часть - из трех, и т.д. Эти последовательности должны быть несжимаемы. Распределение длин серий для нулей и единиц должно быть одинаковым [643, 863, 99, 1357]. Эти свойства могут быть измерены опытным путем и затем сравнены с ожидаемыми статистически с помощью статистики хи-квадрат. Для наших целей генератор последовательности считается псевдослучайным, если он обладает следующим свойством:

1. Он выглядит случайно. Это означает, что он проходит все тесты на случайность, которые нам удалось найти. (Начните с приведенных в [863].)

Множество усилий было затрачено на создание хороших псевдослучайных последовательностей на компьютере. Обсуждение генераторов в большом количестве можно найти в академической литературе вместе с различными тестами на случайность. Все эти генераторы периодичны (этого невозможно избежать), но, если их период 2^{256} и выше, они могут быть использованы в самых серьезных приложениях.

Проблема именно в этих таинственных корреляциях и странных результатах. Каждый генератор псевдослучайных последовательностей создает такие странности, если вы используете его определенным образом. А это именно то, что нужно криптоаналитику для взлома системы.

Криптографически безопасные псевдослучайные последовательности

Криптографические приложения предъявляют к генератору псевдослучайных последовательностей более высокие требования по сравнению с другими приложениями. Криптографическая случайность не ограничивается статистической случайностью, хотя и включает ее. Чтобы последовательность была **криптографически безопасной псевдослучайной** последовательностью, она должна обладать следующим свойством:

2. Она непредсказуема. Должно быть очень трудно (с точки зрения применения вычислительных мощностей) предсказать, каким будет следующий случайный бит, даже если полностью известен алгоритм или устройство, генерирующее последовательность, и все предыдущие биты потока.

Криптографически безопасные псевдослучайные последовательности не должны сжиматься..., если вам неизвестен ключ. Ключом обычно является заданное начальное состояние генератора.

Как и любой криптографический алгоритм, генераторы криптографически безопасных псевдослучайных последовательностей представляют собой предмет вскрытия. Так же как криптографический алгоритм, может быть взломан и генератор криптографически безопасных псевдослучайных последовательностей. Создание устойчивых к вскрытию генераторов является основой криптографии.

Настоящие случайные последовательности

Теперь мы вторгаемся в область, принадлежащую философам. Существует ли такая вещь как случайность? Что такое случайная последовательность? Как узнать, что последовательность случайна? Является ли "101110100" более случайной чем "l01010101"? Квантовая механика убеждает нас в том, что в реальном мире существует настоящая случайность. Но как сохранить эту случайность в предопределенном мире компьютерных микросхем и конечных автоматов?

В сторону философию, с нашей точки зрения генератор последовательности **действительно случаен**, если он обладает третьим свойством:

3. Создаваемая им последовательность не может быть уверенно воспроизведена. Если вы запускаете г е-

нератор случайных чисел дважды с одним и тем же входом (по крайней мере, насколько это в челов е-ческих силах), то вы получите две совершенно независимые случайные последовательности.

Выход генератора, удовлетворяющего всем трем приведенным требованиям, будет достаточно хорош для одноразового блокнота, генерации ключа и других криптографических применений, требующих генерации де й-ствительно случайных последовательностей. Трудность в том, чтобы понять, действительно ли последовател ь-ность случайна? Если я повторно зашифрую строку, используя DES и заданный ключ, я получу хороший, вы-глядящий случайным образом результат, вы не сможете сказать, что он не случаен, пока вы не наймете взло м-щика DES из NSA.

Глава 3
Основные протоколы

3.1 Обмен ключами

Общепринятой криптографической техникой является шифрование каждого индивидуального обмена сообщениями отдельным ключом. Такой ключ называется сеансовым, так как он используется для единственного отдельного сеанса обмена информацией. В разделе 8.5 говорится о том, что сеансовые ключи полезны, так как время их существования определяется длительностью сеанса связи. Передача этого общего сеансового ключа в руки обменивающихся информацией представляет собой сложную проблему.

Обмен ключами с помощью симметричной криптографии

Этот протокол предполагает, что пользователи сети, Алиса и Боб, получают секретный ключ от Центра распределения ключей (Key Distribution Center, KDC) [1260] - Трента наших протоколов. Перед началом протокола эти ключи уже должны быть у пользователей. (Протокол игнорирует очень насущную проблему доставки этих секретных ключей, предполагается, что ключи уже у пользователей, и Мэллори не имеет о них никакой информации.)

(1) Алиса обращается к Тренту и запрашивает сеансовый ключ для связи с Бобом.

(2) Трент генерирует случайный сеансовый ключ. Он зашифровывает две копии ключа: одну для Алисы, а другую - для Боба. Затем Трент посылает обе копии Алисе.

(3) Алиса расшифровывает свою копию сеансового ключа.

(4) Алиса посылает Бобу его копию сеансового ключа.

(5) Боб расшифровывает свою копию сеансового ключа.

(6) Алиса и Боб используют этот сеансовый ключ для безопасного обмена информацией.

Этот протокол основан на абсолютной надежности Трента, для роли которого больше подходит заслуживающая доверия компьютерная программа, чем заслуживающий доверия человек. Если Мэллори получит доступ к Тренту, скомпрометированной окажется вся сеть. В его руках окажутся все секретные ключи, выделенные пользователям Трентом, он сможет прочесть все переданные сообщения, которые ему удалось перехватить, и все будущие сообщения. Ему останется только подключиться к линиям связи и подслушивать зашифрованный поток сообщений.

Другой проблемой такой системы является то, что Трент потенциально является ее узким местом. Он должен участвовать в каждом обмене ключами. Если с ним что-то случится, это разрушит всю систему.

Обмен ключами, используя криптографию с открытыми ключами

Базовая смешанная криптосистема обсуждалась в разделе 1.5. Для согласования сеансового ключа Алиса и Боб применяют криптографию с открытыми ключами, а затем используют этот сеансовый ключ для шифрования данных. В некоторых реализациях подписанные ключи Алисы и Боба доступны в некоторой базе данных. Это значительно облегчает протокол, теперь Алиса, даже если Боб о ней никогда не слышал, может безопасно послать Бобу сообщение:

(1) Алиса получает открытый ключ Боба из KDC.

(2) Алиса генерирует случайный сеансовый ключ, зашифровывает его открытым ключом Боба и посылает его Бобу.

(3) Боб расшифровывает сообщение Алисы с помощью своего закрытого ключа.

(4) Алиса и Боб шифруют свой обмен информацией этим сеансовым ключом.

Вскрытие "человек-в-середине"

В то время, как Ева не может сделать ничего лучшего, чем пытаться взломать алгоритм с открытыми ключами или выполнить вскрытие с использованием только шифротекста, у Мэллори гораздо больше возможностей. Он не только может подслушать сообщения Алисы и Боба, но и изменить сообщения, удалить сообщения и создать совершенно новые. Мэллори может выдать себя за Боба, сообщающего что-то Алисе, или за Алису, сообщающую что-то Бобу. Вот как будет выполнено вскрытие:

(1) Алиса посылает Бобу свой открытый ключ. Мэллори перехватывает его и посылает Бобу свой собственный открытый ключ.

(2) Боб посылает Алисе свой открытый ключ. Мэллори перехватывает его и посылает Алисе Бобу собстве нный открытый ключ.

(3) Когда Алиса посылает сообщение Бобу, зашифрованное открытым ключом "Боба", Мэллори перехватыв ает его. Так как сообщение в действительности зашифровано его собственным открытым ключом, он ра сшифровывает его, снова зашифровывает открытым ключом Боба и посылает Бобу.

(4) Когда Боб посылает сообщение Алисе, зашифрованное открытым ключом "Алисы", Мэллори перехват ывает его. Так как сообщение в действительности зашифровано его собственным открытым ключом, он расшифровывает его, снова зашифровывает открытым ключом Алисы и посылает Алисе.

Это вскрытие будет работать, даже если открытые ключи Алисы и Боба хранятся в базе данных . Мэллори может перехватить запрос Алисы к базе данных и подменить открытый ключ Боба своим собственным . То же самое он может сделать и с открытым ключом Алисы . Или, еще лучше, он может исподтишка взломать базу данных и подменить открытые ключи Бобо и Алисы своим . Теперь он может преуспеть, просто дождавшись, пока Алиса и Боб начнут обмениваться сообщениями, и начав перехватывать и изменять эти сообщения .

Такое **вскрытие "человек-в-середине"** работает, так как у Алисы и Боба нет способа проверить, действ ительно ли они общаются именно друг с другом . Если вмешательство Мэллори не приводит к заметным задер жкам в сети, оба корреспондента и не подумают, что кто-то, сидящий между ними, читает всю их секретную по чту.

Протокол "держась за руки"

Протокол "держась за руки", изобретенный Роном Ривестом (Ron Rivest) и Эди Шамиром (Adi Shamir) [1327], предоставляет неплохую возможность избежать вскрытия "человек-в-середине" . Вот как он работает:

(1) Алиса посылает Бобу свой открытый ключ.

(2) Боб посылает Алисе свой открытый ключ.

(3) Алиса зашифровывает свое сообщение открытым ключом Боба. Половину зашифрованного сообщения она отправляет Бобу.

(4) Боб зашифровывает свое сообщение открытым ключом Алисы. Половину зашифрованного сообщения он отправляет Алисе.

(5) Алиса отправляет Бобу вторую половину зашифрованного сообщения.

(6) Боб складывает две части сообщения Алисы и расшифровывает его с помощью своего закрытого ключа. Боб отправляет Алисе вторую половину своего зашифрованного сообщения.

(7) Алиса складывает две части сообщения Боба и расшифровывает его с помощью своего закрытого ключа.

Идея в том, что половина зашифрованного сообщения бесполезна без второй половины, она не может быть дешифрирована. Боб не сможет прочитать ни одной части сообщения Алисы до этапа (6), а Алиса не сможет прочитать ни одной части сообщения Боба до этапа (7). Существует множество способов разбить сообщение на части:

— Если используется блочный алгоритм шифрования, половина каждого блока (например, каждый второй бит) может быть передана в каждой половине сообщения.

— Дешифрирование сообщения может зависеть от вектора инициализации (см. раздел 9.3), который может быть передан во второй части сообщения.

— Первая половина сообщения может быть однонаправленной хэш -функцией шифрованного сообщения (см. раздел 2.4), а во вторая половина - собственно шифрованным сообщением.

Чтобы понять, как такой протокол помешает Мэллори, давайте рассмотрим его попытку нарушить протокол . Как и раньше, он может подменить открытые ключи Алисы и Боба своим на этапах (1) и (2) . Но теперь, перехватив половину сообщения Алисы на этапе (3), он не сможет расшифровать ее своим закрытым ключом и сн ова зашифровать открытым ключом Боба . Он может создать совершенно новое сообщение и отправить половину его Бобу. Перехватив половину сообщения Боба Алисе на этапе (4), Мэллори столкнется с этой же проблемой . Он не сможет расшифровать ее своим закрытым ключом и снова зашифровать открытым ключом Алисы . Ему придется создать совершенно новое сообщение и отправить половину его Алисе . К тому времени, когда он пе рехватит вторые половины настоящих сообщений на этапах (5) и (6), подменять созданные им новые сообщения будет слишком поздно . Обмен данными между Алисой и Бобом изменится радикально .

Мэллори может попытаться избежать такого результата. Если он достаточно хорошо знает обоих корреспо ндентов, чтобы сымитировать их при обмене данными , они могут никогда не заметить подмены . Но все-таки это сложнее, чем просто сидеть между корреспондентами, перехватывая и читая их сообщения .

Обмен ключами с помощью цифровых подписей

Использование цифровой подписи в протоколе обмена сеансовым ключом также позволяет избежать вскр ытия "человек-в-середине". Трент подписывает открытые ключи Алисы и Боба. Подписанные ключи включают подписанное заверение подлинности. Получив ключи, и Алиса, и Боб проверяют подпись Трента. Теперь они уверены, что присланный открытый ключ принадлежит именно указанному корреспонденту. Затем выполняется протокол обмена ключами.

Мэллори сталкивается с серьезными проблемами. Он не может выдать себя за Алису или Боба, ведь он не знает их закрытых ключей. Он не может подменить их открытые ключи своим, потому что при подписи его ключа Трент указал, что это ключ Мэллори. Все, что ему остается - это прослушивать зашифрованный поток сообщений или испортить линии связи, мешая обмену информации Алисы и Боба.

Трент выступает участником этого протокола, но риск компрометации KDC меньше, чем в первом протоколе. Если Мэллори компрометирует Трента (взламывает KDC), он получает только закрытый ключ Тре нта. Этот ключ позволит ему только подписывать новые ключи, а не расшифровывать сеансовые ключи ил ч итать произвольный поток сообщений. Для чтения сообщений Мэллори придется выдать себя за пользователя сети и обманывать честных пользователей, шифруя сообщения своим поддельным открытым ключом.

Мэллори может предпринять такое вскрытие. Используя закрытый ключ Трента, он может создать поддел ьные подписанные ключи, чтобы обмануть Алису и Боба. Затем он может либо подменить этими ключами н астоящие ключи в базе данных, либо перехватывать запросы пользователей к базе данных и посылать в ответ поддельные ключи. Это позволит ему осуществить вскрытие "человек-в-середине" и читать сообщения польз ователей.

Такое вскрытие будет работать, но помните, что для этого Мэллори должен уметь перехватывать и изменять сообщения. В ряде сетей это намного сложнее, чем просто пассивно сидеть, просматривая сообщения в сети по мере их поступления. В широковещательных каналах, таких как радиосеть, почти невозможно подменить одно сообщение другим - хотя можно забить всю сеть. В компьютерных сетях это менее сложно и, кажется, с каждым днем становится проще и проще. Обратите внимание на подмену IP-адреса, вскрытие маршрутизатора и т.п. Активное вскрытие не обязательно означает, что кто-то засовывает зонд в люк, да и выполнять их теперь могут не только правительственные агентства.

Передача ключей и сообщений

Алисе и Бобу не обязательно выполнять протокол обмена ключами перед обменом сообщениями. В этом протоколе Алиса отправляет Бобу сообщение без предварительного протокола обмена ключами:

(1) Алиса генерирует случайный сеансовый ключ, *K*, и зашифровывает *M* этим ключом.

$E_K(M)$

(2) Алиса получает открытый ключ Боба из базы данных.

(3) Алиса шифрует *K* открытым ключом Боба.

$E_B(K)$

(4) Алиса посылает Бобу шифрованные сообщение и сеансовый ключ.

$E_K(M)$, $E_B(K)$

Для дополнительной защиты от вскрытия "человек-в-середине" Алиса подписывает передачу.

(5) Боб расшифровывает сеансовый ключ Алисы, *K*, используя свой закрытый ключ.

(6) Боб, используя сеансовый ключ, расшифровывает сообщение Алисы.

Подобная смешанная система и употребляется чаще всего в системах связи. Ее можно соединить с цифровыми подписями, метками времени и другими протоколами обеспечения безопасн ости.

Широковещательная рассылка ключей и сообщений

Не существует причин, запрещающих Алисе посылать шифрованное сообщение нескольким людям. В следующем примере Алиса посылает шифрованное сообщение Бобу, Кэрол и Дэйву:

(1) Алиса генерирует случайный сеансовый ключ, *K*, и зашифровывает *M* этим ключом.

$E_K(M)$

(2) Алиса получает из базы данных открытые ключи Боба, Кэрол и Дэйва.

(3) Алиса шифрует *K* открытыми ключами Боба, Кэрол и Дэйва.

$E_B(K)$, $E_C(K)$, $E_D(K)$

(4) Алиса широковещательно посылает шифрованное сообщение и все шифрованные ключи своим корреспондентам.

$E_K(M)$, $E_B(K)$, $E_C(K)$, $E_D(K)$

(5) Только Боб, Кэрол и Дэйв могут, каждый при помощи своего закрытого ключа, расшифровать ключ K.

(6) Только Боб, Кэрол и Дэйв могут расшифровать сообщение Алисы, используя K.

Этот протокол может быть реализован для сетей электронной почты. Центральный сервер может отправить сообщение Алисы Бобу, Кэрол и Дэйву вместе с конкретным шифрованным ключом. Сервер не должен быть надежным и безопасным, так как он не может расшифровать ни одно из сообщений.

3.2 Удостоверение подлинности

Когда Алиса подключается к главному компьютеру (или к автоматическому, или к телефонной банковской системе, или к какому-нибудь другому терминалу), как главный компьютер узнает, кто она? Откуда главный компьютер узнает, что это не Ева, пытающаяся выдать себя за Алису? Обычно эта проблема решается с помощью паролей. Алиса вводит свой пароль, и главный компьютер проверяет его правильность. Таким образом, и Алисе, и главному компьютеру известна некоторая секретная информация, которую главный компьютер запрашивает всякий раз, когда Алиса пытается подключится.

Удостоверение подлинности с помощью однонаправленных функций

Роджер Needхэм (Roger Needham) и Майк Гай (Mike Guy) показали, что главному компьютеру не нужно знать сами пароли, вполне достаточно, чтобы главный компьютер мог отличать правильные пароли от неправильных. Этого легко достичь с помощью однонаправленных функций [1599, 526,1274, 1121]. При этом на главном компьютере хранятся значения однонаправленных функций паролей, а не сами пароли.

(1) Алиса посылает главному компьютеру свой пароль.

(2) Главный компьютер вычисляет однонаправленную функцию пароля.

(3) Главный компьютер сравнивает полученное значение с хранящимся.

Раз главный компьютер больше не хранит таблицу правильных паролей всех пользователей, снижается угроза того, что кто-то проникнет в главный компьютер и выкрадет таблицу паролей. Список паролей, обработанный однонаправленной функцией, бесполезен, так как однонаправленную функцию не удастся инвертировать для получения паролей.

Вскрытия с помощью словаря и "соль"

Файл паролей, зашифрованных однонаправленной функцией, тем не менее, уязвим. Имея запас времени, Мэллори может составить список из миллиона наиболее часто встречающихся паролей. Он обработает весь миллион однонаправленной функцией и сохранит результат. Если каждый пароль состоит из восьми байт, размер получившегося файла не превысит 8 Мбайт, и этот файл может быть размещен всего на нескольких на дискетах. Теперь Мэллори добывает шифрованный файл паролей. Он сравнивает этот файл со своим файлом шифрованных возможных паролей и ищет совпадения.

Это **вскрытие с помощью словаря** может быть удивительно успешным (см. раздел 8.1). "**Соль**" - это способ затруднить его. "Соль" представляет собой случайную строку, добавляемую к паролям перед обработкой их однонаправленной функцией. Затем в базе данных главного компьютера сохраняются и значение "соли", и результат однонаправленной функции. Использование достаточно большого числа возможных значений "соли" практически устраняет возможность вскрытия с помощью словаря, так как Мэллори придется вычислять значение однонаправленной хэш-функции для каждого возможного значения "соли". Это простейший пример использование вектора инициализации (см. раздел 9.3).

Идея состоит в том, чтобы заставить Мэллори выполнить пробное шифрование каждого пароля из его словаря при каждой попытке узнать чей-то чужой пароль вместо одноразовой обработки всех возможных паролей.

Для этого нужно много "соли". Большинство UNIX-систем используют для "соли" 12 бит. Несмотря на это Дэниел Кляйн (Daniel Klein) написал программу разгадывания паролей, которая в некоторых системах за неделю часто вскрывала 40 процентов паролей [847,848] (см. раздел 8.1). Дэвид Фельдмайер (David Feldmeier) и Филип Кан (Philip Karn) составили список из 732000 наиболее часто используемых паролей, присоединив к каждому из них 4096 возможных значений "соли". По их оценкам 30 процентов паролей на любом главном компьютере могут быть взломаны с помощью этого списка [561].

"Соль" не является панацеей, увеличение числа бит "соли" не решит всех проблем. "Соль" предохраняет

только от самых обычных вскрытий файла паролей с использованием словаря, а не от согласованной атаке о дного пароля. Она защищает людей, использующих один и тот же пароль на различных машинах, но не делает лучше плохо выбранный пароль.

SKEY

SKEY - это программа удостоверения подлинности, обеспечивающая безопасность с помощью однонапра вленной функции. Это легко объяснить.

Регистрируясь в системе, Алиса задает случайное число, R. Компьютер вычисляет $f(R)$, $f(f(R))$, $f(f(f(R)))$, и так далее, около сотни раз. Обозначим эти значения как x_1, x_2, x_3, ..., x_{100}. Компьютер печатает список этих чисел, и Алиса прячет его в безопасное место. Компьютер также открытым текстом ставит в базе данных подключения к системе в соответствие Алисе число x_{101}.

Подключаясь впервые, Алиса вводит свое имя и x_{100}. Компьютер рассчитывает $f(x_{100})$ и сравнивает его с x_{101}, если значения совпадают, права Алисы подтверждаются. Затем компьютер заменяет в базе данных x_{101} на x_{100}. Алиса вычеркивает x_{100} из своего списка.

Алиса, при каждом подключении к системе, вводит последнее невычеркнутое число из своего списка: x_i. Компьютер рассчитывает $f(x_i)$ и сравнивает его с x_{i+1}, хранившемся в базе данных. Так как каждый номер используется только один раз, Ева не сможет добыть никакой полезной информации. Аналогично, база данных бесполезна и для взломщика. Конечно же, как только список Алисы исчерпается ей придется перерегистрир оваться в системе.

Удостоверение подлинности с помощью криптографии с открытыми ключами

Даже с использованием "соли" у первого протокола есть серьезные проблемы с безопасностью. Когда Алиса посылает свой пароль главному компьютеру, любой, у кого есть доступ пути передачи ее данных, может пр очесть пароль. Она может получить доступ к своему главному компьютеру посредством запутанного пути пер едачи информации, проложив его через четырех промышленных конкурентов, три других страны и два перед овых университета. Ева может находиться в любой из этих точек, подслушивая передаваемую Алисой послед овательность. Если у Евы есть доступ к оперативной памяти главного компьютера, она сможет подсмотреть п ароль до того, как главный компьютер сможет его хэшировать.

Криптография с открытыми ключами может решить эту проблему. Главный компьютер хранит файл откр ытых ключей всех пользователей, а все пользователи хранят свои закрытые ключи. Вот как выглядит упрощенная попытка организовать протокол подключения:

(1) Главный компьютер посылает Алисе случайную строку.

(2) Алиса шифрует эту строку своим закрытым ключом и посылает ее обратно главному компьютеру вместе со своим именем.

(3) Главный компьютер находит в базе данных открытый ключ Алисы и дешифрирует сообщение, используя этот открытый ключ.

(4) Если отправленная сначала и расшифрованная строки совпадают, главный компьютер предоставляет Алисе доступ к системе

Никто другой не может воспользоваться закрытым ключом Алисы, следовательно никто не сможет выдать себя за нее. Что более важно, Алиса никогда не посылает на компьютер свой закрытый ключ. Ева, подслушивая взаимодействие, не получит никаких сведений, которые позволили бы ей вычислить закрытый ключ Алисы и выдать себя за нее.

Закрытый ключ должен быть достаточно длинным и не должен быть мнемоническим. Он будет автоматич ески обрабатываться аппаратурой пользователя или программным обеспечением связи. Это требует использования "умного" терминала, которому Алиса доверяет, но не главный компьютер, ни линии связи не обязаны быть безопасными.

Глупо шифровать произвольные строки - не только посланные подозрительным авторов, но и вообще любые. Иначе может быть использовано схема вскрытия, обсуждаемая в разделе 19.3. Безопасные идентификационные протоколы имеют следующую, более сложную форму:

(1) Алиса выполняет вычисление, основанное на некоторых случайных числах и своем закрытом ключе, и п осылает результат на главный компьютер.

(2) Главный компьютер посылает другое случайное число.

(3) Алиса выполняет некоторое вычисление, основанное на случайных числах (как созданном ею, так и пол ученном от главного компьютера) и своем закрытом ключе, и посылает результат на главный компьютер.

(4) Главный компьютер выполняет некоторое вычисление для различных чисел, полученных от Алисы, и ее открытого ключа, проверяя, что ей известен ее закрытый ключ.

(5) Если проверка завершается успешно, личность Алисы подтверждается.

Если Алисы доверяет главному компьютеру не в большей степени, чем тот доверяет Алисе, то она должна потребовать подтверждения подлинности главного компьютера аналогичным образом .

Этап (1) может показаться ненужным и запутанным, но он необходим для защиты протокола от вскрытия . Различные протоколы и алгоритмы подтверждения подлинности математически подробно описываются в ра зделах 21.1 и 21.2. См. также [935].

Обоюдное удостоверение подлинности с использованием протокола "держась за руки"

Пусть два пользователя, Алиса и Боб, хотят проверить подлинность друг друга . У каждого из них есть па роль, известный другому пользователю: P_A у Алисы и P_B у Боба. Вот как выглядит протокол, который *не* будет работать:

(1) Алиса и Боб обмениваются открытыми ключами.

(2) Алиса шифрует P_A открытым ключом Боба и посылает его ему.

(3) Боб шифрует P_B открытым ключом Алисы и посылает его ей.

(4) Алиса расшифровывает полученное на этапе (3) и подтверждает правильность пароля.

(5) Боб расшифровывает полученное на этапе (2) и подтверждает правильность пароля.

Мэллори может предпринять успешное вскрытие "человек-в-середине" (см. раздел 3.1):

(1) Алиса и Боб обмениваются открытыми ключами. Мэллори перехватывает оба сообщения, и посылает обоим корреспондентам свой собственный открытый ключ, подменив им их ключи.

(2) Алиса шифрует P_A открытым ключом "Боба" и посылает его ему. Мэллори перехватывает сообщение, расшифровывает P_A с помощью своего закрытого ключа, снова шифрует P_A открытым ключом Боба и по сылает его ему.

(3) Боб шифрует P_B открытым ключом "Алисы" и посылает его ей. Мэллори перехватывает сообщение, ра с шифровывает P_B с помощью своего закрытого ключа, снова шифрует P_B открытым ключом Алисы и по сылает его ей.

(4) Алиса расшифровывает P_B и подтверждает его правильность.

(5) Боб расшифровывает P_A и подтверждает его правильность.

Для Алисы и Боба ничего не изменилось. Однако, Мэллори знает и P_A, и P_B. Дональд Дэвис (Donald Davies) и Вильям Прайс (William Price) описывают, как протокол "держась-за-руки" (см. раздел 3.1) противодействует такому вскрытию [435]. Стив Белловин (Steve Bellovin) и Майкл Мерритт (Michael Merritt) рассматривают спо собы вскрытия этого протокола в [110]. Если Алиса - это пользователь, а Боб - хост-компьютер , Мэллори может предпочесть быть Бобом, выполнить первые этапа протокола с Алисой и разорвать соединение . Симулирование шума на линии или сетевого отказа потребует от Мэллори настоящего артистизма, но в результате Мэллори получит пароль Алисы. Затем он сможет соединиться с Бобом и завершить протокол, получая и пароль Боба .

Протокол можно изменить так, чтобы Боб передавал свой пароль перед Алисой в предположении, что п ароль пользователя более важен чем пароль главного компьютера . Это приведет к усложнению способа вскр ытия, также описанного в [110].

SKID

SKID2 и SKID3 - это симметричные криптографические протоколы идентификации, разработанные для пр оекта RACE RIPE [1305] (см. раздел 25.7). Они используют MAC (см. раздел 2.4) для обеспечения безопасности и предполагают, что Алиса и Боб используют общий секретный ключ , K. SKID2 позволяет Бобу доказать свою подлинность Алисе. Вот этот протокол:

(1) Алиса выбирает случайное число, R_A. (Документами RIPE определяется 64-битовое число). Она посылает это число Бобу.

(2) Боб выбирает случайное число, R_B. (Документами RIPE определяется 64-битовое число). Он посылает Алисе.

R_B, $H_K(R_A, R_B, B)$

H_K - это MAC. (В документах RIPE предлагается функция RIPE-MAC, см. раздел 18.4.) B - это имя Боба.

(3) Алиса рассчитывает $H_K(R_A, R_B, B)$ и сравнивает результат со значением, полученным от Боба. Если результаты совпадают, Алиса убеждается в том, что она соединилась именно с Бобом.

SKID3 обеспечивает совместную проверку подлинности Алисой и Бобом. Этапы (1) - (3) совпадают с протоколом SKID2, а затем выполняются следующие действия:

(4) Алиса посылает Бобу:

$H_K(R_B, A)$

A - это имя Алисы.

(5) Боб рассчитывает $H_K(R_B, A)$ и сравнивает результат со значением, полученным от Алисы. Если результаты совпадают, Боб убеждается в том, что она соединилась именно с Алисой.

Этот протокол неустойчив к вскрытию "человек-в-середине". В общем случае, вскрытие "человек-в-середине" может угрожать любому протоколу, в который не входит какой-нибудь секрет.

Удостоверение подлинности сообщений

Когда Боб получает сообщение от Алисы, как ему узнать, что это сообщение подлинно? Если Алиса подписала свое сообщение, то все просто. Цифровая подпись Алисы достаточна, чтобы подтвердить кому угодно подлинность ее сообщения.

Некоторую проверку подлинности предоставляют и симметричные алгоритмы. Когда Боб получает сообщение от Алисы, шифрованное их общим ключом, он знает, что это сообщение от Алисы. Никто больше не знает их ключа. Однако, у Боба нет возможности убедить в этом кого-то еще. Боб не может показать сообщение Тренту и убедить его, что оно отправлено Алисой. Трент может сделать вывод, что сообщение отправлено или Алисой, или Бобом (так как их секретный ключ никому больше не принадлежит), но у него нет способа определить, кто же конкретно автор сообщения.

Если сообщение не шифровано, Алиса может также использовать MAC. Это также убедит Боба в подлинности сообщения, но возникнут те же проблемы, что и для решений симметричной криптографии.

3.3 Удостоверение подлинности и обмен ключами

Эти протоколы объединяют удостоверение подлинности и обмен ключами для решения основной компьютерной проблемы: Алиса и Боб хотят безопасно обмениваться сообщениями, находясь на различных концах сети. Как могут Алиса и Боб обменяться секретным ключом, при этом сохраняя уверенность, что они обмениваются сообщениями друг с другом, а не с Мэллори? В большинстве протоколов предполагается, что каждому пользователю Трент выделяет отдельный секретный ключ, и перед началом работы протокола все ключи уже находятся у пользователей. Символы, используемые в этих протоколах, сведены в 2-й.

Табл. 3-1.
Символы, используемые в протоколах удостоверения подлинности и обмена ключами

A	Имя Алисы
B	Имя Боба
E_A	Шифрование ключом, выделенном Трентом Алисе
E_B	Шифрование ключом, выделенном Трентом Бобу
I	Порядковый номер
K	Случайное сеансовое число
L	Время жизни
T_A, T_B	Метки времени
R_A, R_B	Случайные числа, выбранные Алисой и Бобом, соответственно

Лягушка с широким ртом

Протокол "Лягушка с широким ртом" (Wide-Mouth Frog) [283,284], возможно, является простейшим симметричным протоколом управления ключами, в котором используется заслуживающий доверия сервер. Алиса и Боб делят свой секретный ключ с Трентом. Эти ключи используются только для распределения ключей, а не для шифрования пользовательских сообщений. Вот как, используя два сообщения, Алиса передает Бобу сеансовый

ключ:

(1) Алиса объединяет метку времени, имя Боба и случайный сеансовый ключ, затем шифрует созданное с о-общение общим с Трентом ключом и посылает его Тренту вместе со своим именем.

$A, E_A(T_A, B, K)$

(2) Трент расшифровывает сообщение от Алисы. Затем он добавляет новую метку времени, имя Алисы и сл у-чайный сеансовый ключ, шифрует полученное сообщение общим с Бобом ключом. Трент посылает Бобу:

$E_B(T_B, B, K)$

Наибольшим допущением, сделанным в этом протоколе, является то, что Алиса обладает достаточной ко м-петентностью для генерации хороших сеансовых ключей . Вспомните, что случайные числа генерировать совсем не просто, для этого может потребоваться кто-нибудь понадежнее Алисы .

Yahalom

В этом протоколе Алисы и Боб делят с Трентом секретный ключ [283,284].

(1) Алиса объединяет свое имя и случайное число, и отправляет созданное сообщение Бобу.

A, R_A

(2) Боб объединяет имя Алисы, ее случайное число, свое случайное число, шифрует созданное сообщение о б-щим с Трентом ключом и посылает его Тренту, добавляя свое имя:

$B, E_B(A, R_A, R_B)$

(3) Трент создает два сообщения. Первое включает имя Боба, случайный сеансовый ключ, случайные числа Боба и Алисы и шифруется ключом, общим для Трента и Алисы. Второе состоит из имени Алисы, сл у-чайного сеансового ключа и шифруется ключом, общим для Трента и Боба. Трент посылает оба сообщ е-ния Алисе:

$E_A(B, K, R_A, R_B), E_B(A, K)$

(4) Алиса расшифровывает первое сообщение, извлекает K и убеждается, что R_A совпадает со значением, от-правленным на этапе (1). Алиса посылает Бобу два сообщения. Одним является сообщение Трента, з а-шифрованное ключом Боба. Второе - это R_B, зашифрованное сеансовым ключом.

$E_B(A, K), E_K(R_B),$

(5) Боб расшифровывает первое сообщение, извлекает K и убеждается, что R_B совпадает с отправленным на этапе (2).

В результате Алиса и Боб убеждены, что они общаются именно друг с другом, а не с третьей стороной . Нововведение состоит в том, что именно Боб первым обращается к Тренту, который только посылает одно соо б-щение Алисе.

Needham-Schroeder

В этом протоколе, изобретенном Роджером Неедхэмом (Roger Needham) и Майклом Шредером (Michael Schroeder) [1159], также используются симметричная криптография и Трент .

(1) Алиса посылает Тренту сообщение, содержащее ее имя, имя Боба и случайное число.

A, B, R_A

(2) Трент генерирует случайный сеансовый ключ. Он шифрует сообщение, содержащее случайный сеансовый ключ и имя Алисы, секретным ключом, общим для него и Боба. Затем он шифрует случайное число Ал и-сы, имя Боба, ключ, и шифрованное сообщение секретным ключом, общим для него и Алисы. Наконец, он отправляет шифрованное сообщение Алисе:

$E_A(R_A, B, K, E_B(K, A))$

(3) Алиса расшифровывает сообщение и извлекает K. Она убеждается, что R_A совпадает со значением, от-правленным Тренту на этапе (1). Затем она посылает Бобу сообщение, зашифрованное Трентом ключом Боба.

$E_B(K, A)$

(4) Боб расшифровывает сообщение и извлекает K. Затем он генерирует другое случайное число, R_B. Он шифрует это число ключом K и отправляет его Алисе.

$E_K(R_B)$

(5) Алиса расшифровывает сообщение с помощью ключа K. Она создает число R_B-1 и шифрует это число ключом K. Затем она посылает это сообщение обратно Бобу.

$E_K(R_B\text{-}1)$

(6) Боб расшифровывает сообщение с помощью ключа K и проверяет значение R_B-1.

Вся эта возня с R_A, R_B, и R_B-1 служит для предотвращения **вскрытия с повторной передачей**. При таком способе вскрытия Мэллори может записать старые сообщения и впоследствии использовать их при попытке взломать протокол. Присутствие R_A на этапе (2) убеждает Алису, что сообщение Трента достоверно и не является повторной передачей отклика, использованного при одном из предыдущих применений протокола. Когда Алиса успешно расшифрует R_B и передает Бобу R_B-1 на этапе (5), Боб убеждается, что сообщения Алисы не является повторной передачей сообщений, использованных при одном из предыдущих применений протокола.

Главной прорехой этого протокола является важность использованных сеансовых ключей. Если Мэллори получит доступ к старому K, он сможет предпринять успешное вскрытие [461]. Ему нужно только записать сообщения Алисы Бобу на этапе (3). Тогда, имея K, он может выдать себя за Алису:

(1) Мэллори посылает Бобу следующее сообщение:

$E_B(K, A)$

(2) Боб извлекает K, генерирует R_B и отправляет Алисе:

$E_K(R_B)$

(3) Мэллори перехватывает сообщение, расшифровывает его с помощью ключа K и посылает Бобу:

$E_K(R_B\text{-}1)$

(4) Боб убеждается, что сообщение "Алисы" состоит из R_B-1.

Теперь Мэллори убедил Боб, что он и есть "Алиса". Более защищенный протокол, использующий метки времени, может противостоять этому вскрытию [461,456]. Метки времени добавляются к сообщению Трента на этапе (2) и шифруются ключом Боба: $E_B(K, A, T)$. Метки времени требуют надежной и точной системы единого времени, что само по себе нетривиальная проблема.

Если ключ, общий для Трента и Алисы будет скомпрометирован, последствия будут драматичны. Мэллори сможет использовать его, для получения сеансовых ключей для обмена сообщениями с Бобом (или с кем-нибудь еще). Даже хуже, Мэллори продолжать подобные действия даже после замены ключа Алисы [90].

Неедхэм и Шредер пытались исправить эти проблемы в модифицированной версии своего протокола [1160]. Их новый протокол по существу совпадает с протоколом Отуэя- Риса (Otway-Rees), опубликованном в том же выпуске того же журнала.

Otway-Rees

Этот протокол также использует симметричную криптографию [1224].

(1) Алиса создает сообщение, состоящее из порядкового номера, ее имени, имени Боба и случайного числа. Сообщение шифруется ключом, общим для Алисы и Трента. Она посылает это сообщение Бобу вместе с порядковым номером, ее и его именами:

$I, A, B, E_A(R_A, I, A, B)$

(2) Боб создает сообщение, состоящее из нового случайного числа, порядкового номера, имени Алисы и имени Боба. Сообщение шифруется ключом, общим для Алисы и Боба. Он посылает это сообщение Тренту вместе шифрованным сообщением Алисы, порядковым номером, ее и его именами:

$I, A, B, E_A(R_A, I, A, B), E_B(R_B, I, A, B)$

(3) Трент генерирует случайный сеансовый ключ. Затем он создает два сообщения. Одно, состоящее из случайного числа Алисы и сеансового ключа, шифруется ключом, общим для него и Алисы. Другое, состоящее из случайного числа Боба и сеансового ключа, шифруется ключом, общим для него и Боба. Он отправляет два этих сообщения вместе с порядковым номером Бобу:

$I, E_A(R_A, K), E_B(R_B, K)$

(4) Боб отправляет Алисе сообщение, шифрованное ее ключом, и порядковый номер:

$I, E_A(R_A, K)$

(5) Алиса расшифровывает сообщение, получая свои ключ и случайное число. Алиса убеждается, что при выполнении протокола они не изменились Боб отправляет Алисе сообщение, шифрованное ее ключом, и по-

рядковый номер.

Если все случайные числа правильны, а порядковый номер не изменился при выполнении протокола, Алиса и Боб убеждаются в подлинности друг друга и получают секретный ключ для обмена сообщениями .

Kerberos

Kerberos - вариант протокола Needham-Schroeder - подробно обсуждается в разделе 24.5. В базовом протоколе Kerberos Version 5 у Алисы и Боба общие ключи с Трентом. Алиса хочет генерировать сеансовый ключ для сеанса связи с Бобом.

(1) Алиса посылает Тренту сообщение со своим именем и именем Боба:

A, B

(2) Трент создает сообщение, состоящее из метки времени, время жизни, L, случайного сеансового ключа и имени Алисы. Он шифрует сообщение ключом, общим для него и Боба. Затем он объединяет метку времени, время жизни, сеансовый ключ, имя Боба, и шифрует полученное сообщение ключом, общим для него и Алисы. Оба шифрованных сообщения он отправляет Алисе.

$E_A(T, L, K, B), E_B(T, L, K, A)$

(3) Алиса создает сообщение, состоящее из ее имени и метки времени, шифрует его ключом K и отправляет Бобу. Алиса также посылает Бобу сообщение от Трента, шифрованное ключом Боба:

$E_A(A, T), E_B(T, L, K, A)$

(4) Боб создает сообщение, состоящее из метки времени плюс единица, шифрует его ключом K и отправляет Алисе:

$E_K(T+1)$

Этот протокол работает, но только если часы каждого пользователя синхронизированы с часами Трента . На практике эффект достигается синхронизацией с надежным сервером времени с точностью в несколько минут и обнаружением повторной передачи в течение определенного интервала времени .

Neuman-Stubblebine

Из-за недостатков системы или саботажа синхронизация часов может быть нарушена . Если часы сбиваются, против большинства протоколов может быть использован определенный способ вскрытия [644]. Если часы отправителя опережают часы получателя, Мэллори может перехватить сообщение отправителя и повторно использовать его позднее, когда метка времени станет текущей в месте нахождения получателя . Этот способ, называющийся вскрытием с **подавлением повторной передачи**, может привести к неприятным последствиям.

Этот протокол, впервые опубликованный в [820] и исправлен в [1162], пытается противостоять вскрытию с подавлением повторной передачи. Этот отличный протокол является улучшением Yahalom.

(1) Алиса объединяет свое имя и случайное число, и отправляет созданное сообщение Бобу.

A, R_A

(2) Боб объединяет имя Алисы, ее случайное число и метку времени, шифрует созданное сообщение общим с Трентом ключом и посылает его Тренту, добавляя свое имя и новое случайное число:

$B, R_B, E_B(A, R_A, T_B)$

(3) Трент генерирует случайный сеансовый ключ. Затем он создает два сообщения. Первое включает имя Боба, случайное число Алисы, случайный сеансовый ключ, метку времени и шифруется ключом, общим для Трента и Алисы. Второе состоит из имени Алисы, сеансового ключа, метки времени и шифруется ключом, общим для Трента и Боба. Трент посылает оба сообщения Алисе вместе со случайным числом Боба:

$E_A(B, R_A, K, T_B), E_B(A, K, T_B), R_B$

(4) Алиса расшифровывает сообщение, зашифрованное ее ключом, извлекает K и убеждается, что R_A совпадает со значением, отправленным на этапе (1). Алиса посылает Бобу два сообщения. Одним является сообщение Трента, зашифрованное ключом Боба. Второе - это R_B, зашифрованное сеансовым ключом.

$E_B(A, K), E_K(R_B),$

(5) Боб расшифровывает сообщение, зашифрованное его ключом, извлекает K и убеждается, что значения T_B и R_B те же, что и отправленные на этапе (2).

Если оба случайных числа и метка времени совпадают, Алиса и Боб убеждаются в подлинности дуг друга и получают секретный ключ. Синхронизация часов не требуется, так как метка времени определяется только по

часам Боба, и только Боб проверяет созданную им метку времени .

У этого протокола есть еще одно полезное свойство - Алиса может использовать полученное от Трента с о-общение для последующей проверки подлинности Боба в пределах некоторого времени . Предположим, что Алиса и Боб выполнили приведенный выше протокол, провели и завершили сеанс связи . Алиса и Боб могут повторно проверить подлинность друг друга, не обращаясь к Тренту .

(1) Алиса посылает Бобу сообщение, присланное ей Трентом на этапе (3) и новое случайное число.

$E_B(A, K, T_B), R'_A$

(2) Боб посылает Алисе другое новое случайное число и случайное число, присланное Алисой, шифруя их с е-ансовым ключом связи.

$R'_B, E_K(R'_A)$

(3) Алиса посылает Бобу его новое случайное число, шифруя его сеансовым ключом связи.

$E_K(R'_B)$

Новые случайные числа защищают от вскрытия с повторно передачей .

DASS

Протоколы Распределенной служба безопасности и проверки подлинности (Distributed Authentication Sec u-rity Service, DASS), созданные в Digital Equipment Corporation, также обеспечивают обоюдную проверку по д-линности и обмен ключами [604, 1519, 1518]. В отличие от предыдущего протокола DASS использует как крип-тографию с открытыми ключами, так и симметричную криптографию . И у Алисы, и у Боба есть свой закрытый ключ. Трент подписывает копии их открытых кл ючей.

(1) Алиса посылает Тренту сообщение, состоящее из имени Боба.

B

(2) Трент посылает Алисе открытый ключ Боба, K_B, подписанный закрытым ключом Трента, T. Подписанное сообщение содержит имя Боба.

$S_T(B, K_B)$

(3) Алиса проверяет подпись Трента, убеждаясь, что она действительно получила открытый ключ Боба. Она генерирует случайный сеансовый ключ, K, и случайную пару ключей открытый/закрытый, K_p. Она шиф-рует метку времени ключом K, а затем подписывает время жизни, L, свое имя и своим закрытым ключом, K_A. Наконец, она зашифровывает K открытым ключом Боба и подписывает его с помощью K_p. Все это она отправляет Бобу.

$E_K(T_A), S_{K_A}(L, A, K_p), S_{K_p}(E_{K_B}(K))$

(4) Боб посылает Тренту (это может быть другой Трент) сообщение, состоящее из имени Алисы.

A

(5) Трент посылает Бобу открытый ключ Алисы, K_A, подписанный закрытым ключом Трента. Подписанное сообщение содержит имя Алисы.

$S_T(A, K_A)$

(6) Боб проверяет подпись Трента, убеждаясь, что он действительно получила открытый ключ Алисы. Затем он проверяет подпись Алисы и извлекает K_p. Боб использует свой закрытый ключ, извлекая K. Затем он расшифровывает T_A, проверяя, что это сообщение - текущее.

(7) Если требуется обоюдная проверка подлинности, Боб шифрует новую метку времени ключом K и посыла-ет ее Алисе.

$E_K(T_B)$

(8) Алиса расшифровывает T_B ключом K, проверяя, что это сообщение - текущее.

SPX, продукт DEC, основан на DASS. Дополнительную информацию можно найти в [34].

Denning-Sacco

В этом протоколе также используется криптография с открытыми ключами [461]. Трент ведет базу данных, хранящую открытые ключи всех пользователей .

(1) Алиса посылает Тренту сообщение, состоящее из ее имени и имени Боба.

A, B

(2) Трент посылает Алисе открытый ключ Боба, K_B, подписанный закрытым ключом Трента, T. Трент также посылает Алисе ее собственный открытый ключ, K_A, подписанный закрытым ключом Трента.

$S_T(B, K_B), S_T(A, K_A)$

(3) Алиса посылает Бобу случайный сеансовый ключ и метку времени, подписав их своим закрытым ключом и зашифровав открытым ключом Боба, вместе с обоими подписанными ключами.

$E_B(S_A(K, T_A)), S_T(A, K_A), S_T(B, K_B)$

(4) Боб расшифровывает сообщение Алисы с помощью своего закрытого ключа и проверяет подпись Алисы с помощью ее открытого ключа. Он также убеждается, что метка времени правильна.

С этого момента Алиса и Боб получили K и могут провести безопасный сеанс связи. Это выгляди красиво, но есть одна тонкость - выполнив протокол с Алисой, Боб сможет выдать себя за Алису [5]. Смотрите:

(1) Боб посылает Тренту свое имя и имя Кэрол.

B, C

(2) Трент посылает Бобу подписанные открытые ключи Боба и Кэрол.

$S_T(B, K_B), S_T(C, K_C)$

(3) Боб посылает Кэрол подписанный случайный сеансовый ключ и метку времени, ранее полученные от Алисы, зашифровав их открытым ключом Кэрол, вместе с подтверждением Алисы и подтверждением К эрол.

$E_C(S_A(K, T_A)), S_T(A, K_A), S_T(C, K_C)$

(4) Кэрол расшифровывает сообщение Алисы с помощью своего закрытого ключа и проверяет подпись Ал исы с помощью ее открытого ключа. Он также убеждается, что метка времени правильна.

Теперь Кэрол считает, что она соединилась с Алисой, Боб успешно одурачил ее . Действительно, Боб сможет дурачить любого пользователя сети, пока не закончится срок действия метки времени . Но это легко можно исправить. Просто вставьте имена в шифрованное сообщение на этапе (3) :

$E_B(S_A(A, B, K, T_A)), S_T(A, K_A), S_T(B, K_B)$

Теперь Боб не сможет повторно послать Кэрол старое сообщение, потому что оно явно предназначено для сеанса связи между Алисой и Бобом .

Woo-Lam

В этом протоколе также используется криптография с открытыми ключами [1610, 1611]:

(1) Алиса посылает Тренту сообщение, состоящее из ее имени и имени Боба.

A, B

(2) Трент посылает Алисе открытый ключ Боба, K_B, подписанный закрытым ключом Трента, T.

$S_T(K_B)$

(3) Алиса проверяет подпись Трента. Затем она посылает Бобу свое имя и случайное число, шифрованное о ткрытым ключом Боба.

$A, E_B(R_B)$

(4) Боб посылает Тренту свое имя, имя Алисы и случайное число Алисы, шифрованное открытым ключом Трента, K_T.

$A, B, E_{K_T}(R_A)$

(5) Трент посылает Бобу открытый ключ Алисы, K_A, подписанный закрытым ключом Трента. Он также посылает Бобу случайное число Алисы, случайный сеансовый ключ, имена Алисы и Боба, подписав все это закрытым ключом Трента и зашифровав открытым ключом Боба.

$S_T(K_A), E_{K_B}(S_T(R_A, K, A, B))$

(6) Боб проверяет подписи Трента. Затем он посылает Алисе вторую часть сообщения Трента, полученного на этапе (5), и новое случайное число, зашифровав все открытым ключом Алисы.

$E_{K_A}(S_T(R_A, K, A, B), R_B)$

(7) Алиса проверяет подпись Трента и свое случайное число. Затем она посылает Бобу второе случайное чи сло, шифрованное сеансовым ключом.

$E_K(R_B)$

(8) Боб расшифровывает свое случайное число и проверяет, что оно не изменилось.

Другие протоколы

В литературе описано множество протоколов. Протоколы X.509 рассматриваются в разделе 24.9, KryptoKnight - в разделе 24.6, а Шифрованный обмен ключами (Encrypted Key Exchange) - в разделе 22.5.

Другим новым протоколом с открытыми ключами является Kuperee [694]. Ведется работа на протоколами, использующими **маяки** - заслуживающие доверия узлы сети, которые непрерывно и широковещательно пер едают достоверные метки времени [783].

Выводы

Из приведенных протоколов, как из тех, которые вскрываются, так и из надежных, можно извлечь ряд ва жных уроков:

— Многие протоколы терпят неудачу, так как их разработчики пытались быть слишком умными. Они опт имизировали протоколы, убирая важные элементы - имена, случайные числа и т.п. - и пытаясь сделать протоколы как можно более прозрачными [43, 44].

— Оптимизация - эта манящая ловушка - сильно зависит от сделанных предположений. Пример: наличие достоверного времени позволяет вам реализовать многие вещи, невозможные в проти вном случае.

— Выбираемый протокол зависит от архитектуры используемых средств связи. Хотите ли вы минимизир овать размер сообщений или их количество? Могут ли сторона взаимодействовать каждый с каждым или круг их общения будет ограничен?

Именно подобные вопросы и привели к созданию формальных методов анализа протоколов .

3.4 Формальный анализ протоколов проверки подлинности и обмена ключами

Проблема выделения безопасного сеансового ключа для пары компьютеров (или людей) в сети настолько фундаментальна, что стала причиной многих исследований. Некоторые исследования заключались в разработке протоколов, подобных рассматриваемым в разделах 3.1, 3.2 и 3.3. Это, в свою очередь, привело к появлению более важной и интересной задачи: формальному анализу протоколов проверки подлинности и обмена ключами. Иногда прорехи в протоколах, кажущихся вполне надежными, обнаруживались спустя много лет п осле их разработки, и разработчикам потребовались средства, позволяющие сразу же проверять безопасность протокола. Хотя большая часть этого инструментария применима и к более общим криптографическим прот околам, особое внимание уделялось проверке подлинности и обмену ключами. Существует четыре основных подхода к анализу криптографических протоколов [1045]:

1. Моделирование и проверка работы протокола с использованием языков описания и средств проверки, не разработанных специально для анализа криптографических протоколов.
2. Создание экспертных систем, позволяющих конструктору протокола разрабатывать и исследовать различные сценарии.
3. Выработка требований к семейству протоколов, используя некую логику для анализа понятий "знание" и "доверие".
4. Разработка формальных методов, основанных на записи свойств криптографических систем в алге браическом виде.

Полное описание этих четырех подходов и связанных с ними исследований выходит за рамки данной книги. Хорошее введение в эту тему дано в [1047, 1355], я же собираюсь коснуться только основных вопросов.

Первый из подходов пытается доказать правильность протокола, рассматривая его как обычную компьюте рную программу. Ряд исследователей представляют протокол как конечный автомат [1449, 1565], другие используют расширения методов исчисления предиката первого порядка [822], а третьи для анализа протоколов используют языки описания [1566]. Однако, доказательство правильности отнюдь не является доказательством безопасности, и этот подход потерпел неудачу при анализе многих "дырявых" протоколов. И хотя его применение поначалу широко изучалось, с ростом популярности третьего из подходов работы в этой области были п ереориентированы.

Во втором подходе для определения того, может ли протокол перейти в нежелательное состояние (например, потеря ключа), используются экспертные системы. Хотя этот подход дает лучшие результаты при поиске "дыр", он не гарантирует безопасности и не предоставляет методик разработки вскрытий. Он хорош для проверки того, содержит ли протокол конкретную "дыру", но вряд ли способен обнаружить неизвестные "дыры" в протоколе. Примеры такого подхода можно найти в [987,1521], а в [1092] обсуждается экспертная система, разработанная армией США и названная Следователем (Interrogator).

Третий подход гораздо популярнее. Он был впервые введен Майклом Бэрроузом (Michael Burrows), Мартином Абэди (Martin Abadi) и Роджером Needхэмом. Они разработали формальную логическую модель для анализа знания и доверия, названную **БАН-логикой** [283, 284]. БАН-логика является наиболее широко распространена при анализе протоколов проверки подлинности. Она рассматривает подлинность как функцию от целостности и новизны, используя логические правила для отслеживания состояния этих атрибутов на протяжении всего протокола. Хотя были предложены различные варианты и расширения, большинство разработчиков протоколов до сих пор обращаются к оригинальной работе.

БАН-логика не предоставляет доказательство безопасности, она может только рассуждать о проверке подлинности. Она является простой, прямолинейной логикой, легкой в применении и полезной при поиске "дыр". Вот некоторые предложения БАН-логики:

Алиса считает X. (Алиса действует, как если бы X являлось истиной.)

Алиса видит X. (Кто-то послал сообщение, содержащее X, Алисе, которая может прочитать и снова передать X - возможно после дешифрирования.)

Алиса сказала X. (В некоторый момент времени Алиса послала сообщение, которое содержит предложение X. Не известно, как давно было послано сообщение, и было ли оно послано в течении текущего выполнения протокола. Известно, что Алиса считала X, когда говорила X.)

X ново. (X никогда не было послано в сообщении до текущего выполнения протокола.)

И так далее. БАН-логика также предоставляет правила для рассуждения о доверии протоколу. Для доказательства чего-либо в протоколе или для ответа на какие-то вопросы к логическим предложениям о протоколе можно применить эти правила. Например, одним из правил является правило о значении сообщения:

ЕСЛИ Алиса считает, что у Алисы и Боба общий секретный ключ, *K,* и Алиса видит *X*, шифрованное *K,* и Алиса не шифровала *X* с помощью *K*, ТО Алиса считает, что Боб сказал *X.*

Другим является правило подтверждения метки времени:

ЕСЛИ Алиса считает, что *X* могло быть сказано только недавно, и, что Боб *X* когда-то сказал *X*, ТО Алиса считает, что Боб считает *X.*

БАН-анализ делится на четыре этапа:

(1) Преобразуйте протокол к идеальной форме, используя описанные выше предложения.

(2) Добавьте все предположения о начальном состоянии протокола.

(3) Присоедините логические формулы к предложениям, получая утверждения о состоянии системы после каждого предложения.

(4) Примените логические постулаты к утверждениям и предположениям, чтобы раскрыть состояние доверия участников протокола.

Авторы БАН-логики "рассматривают идеализированные протоколы как более ясные и полные описания, чем традиционные, найденные в литературе..." [283, 284]. Другие исследователи не так оптимистичны и подвергают это действие критике, так как при этом реальный протокол может быть искажен [1161, 1612]. Дальнейшие споры отражены в [221, 1557]. Ряд критиков пытается показать, что БАН-логика может и получить очевидно неправильные характеристики протоколов [1161] - см. контрдоводы в [285, 1509] - и что БАН-логика занимается только доверием, а не безопасностью [1509]. Подробное обсуждение приведено в [1488, 706, 1002].

Несмотря на эту критику БАН-логика достигла определенных успехов. Ей удалось обнаружить "дыры" в нескольких протоколах, включая Needham-Schroeder и раннюю черновую версию протокола CCITT X.509 [303]. Она обнаружила избыточность во многих протоколах, включая Yahalom, Needham-Schroeder и Kerberos. Во многих опубликованных работах БАН-логика используется для заявления претензии о безопасности описываемых протоколов [40, 1162, 73].

Были опубликованы и другие логические системы, некоторые из них разрабатывались как расширения БАН-логики [645, 586, 1556, 828], а другие основывались на БАН-логике для исправления ощутимых слабостей [1488, 1002]. Из них наиболее успешной оказалась CNY [645], хотя у нее есть ряд изъянов [40]. В [292,474] к БАН-логике с переменным успехом были добавлены вероятностные доверия. Другие формальные логики описаны в [156, 798,288]. [1514] пытается объединить черты нескольких логик. А в [1124, 1511] представлены логики, в которых доверия изменяются со временем.

Четвертый подход к анализу криптографических протоколов предлагает моделировать протокол как алгебраическую систему, выразить состояние знания участников о протоколе и затем проанализировать достиж-

мость определенных состояний. Этот подход пока не привлек столько внимания, как формальная логика, но состояние дел меняется. Он впервые был использован Майклом Мерриттом [1076], который показал, что для анализа криптографических протоколов можно использовать алгебраическую модель. Другие подходы рассмотрены в [473, 1508, 1530, 1531, 1532, 1510, 1612].

Анализатор протоколов Исследовательской лаборатория ВМС (Navy Research Laboratory, NRL), возможно, является наиболее успешным применением этих методов [1512, 823, 1046, 1513]. Он был использован для поиска как новых, так и известных "дыр" во множестве протоколов [1044, 1045, 1047]. Анализатор протоколов определяет следующие действия:

— Принять (Боб, Алиса, *M*, *N*). (Боб принимает сообщение *M* как пришедшее от Алисы в течение локального раунда Боба *N*.)

— Узнать (Ева, *M*). (Ева узнает *M*.)

— Послать (Алиса, Боб, *Q*, *M*). (Алиса посылает *M* Бобу в ответ на запрос, *Q*.)

— Запросить (Боб, Алиса, *Q*, *N*). (Боб посылает *Q* Алисе в течение локального раунда Боба *N*.)

Используя эти действия, можно задать требования. Например:

— Если Боб принял сообщение *M* от Алисы в какой-то прошедший момент времени, то Ева не знает *M* в какой-то прошедший момент времени.

— Если Боб принял сообщение *M* от Алисы в течение локального раунда Боба *N*, то Алиса послала *M* Бобу в ответ на запрос Боба в локальном раунде Боба *N*.

Для анализа Анализатором протоколов NRL исследуемый протокол должен быть описан с помощью приведенных конструкций. Затем выполняются четыре фазы анализа: определение правил перехода для честных участников, описание операций, возможных и для полностью честных, и для нечестных участников, описание базовых блоков протокола и описание правил преобразования. Смысл всего этого в том, чтобы показать, что данный протокол удовлетворяет необходимым требованиями. Использование инструментов, подобных Анализатору протоколов NRL, в итоге могло бы привести к созданию протокола, который был бы обоснованно признан безопасным.

Хотя формальные методы в основном применяются к уже существующим протоколам, сегодня есть тенденция использовать их и при проектировании протоколов. Ряд предварительных шагов в этом направлении сделан в [711]. Это же пытается сделать и анализатор протоколов NRL [1512, 222, 1513].

Применение формальных методов к криптографическим протоколам представляет собой качественно новую идею, и трудно обрисовать, к чему может привести ее реализация. С этой точки зрения слабейшим звеном кажется процесс формализации.

3.5 Криптография с несколькими открытыми ключами

Обычная криптография с открытыми ключами использует два ключа. Сообщение, зашифрованное одним ключом, может быть расшифровано другим. Обычно один ключ является закрытым, а другой - открытым. Пусть, один ключ находится у Алисы, а другой - у Боба. Мы хотим реализовать следующую схему: Алиса может зашифровать сообщение так, что только Боб сможет расшифровать его, а Боб может зашифровать сообщение так, что только Алиса сможет прочитать его.

Эта концепция была обобщены Конном Бойдом (Conn Boyd) [217]. Представьте себе вариант криптографии с открытыми ключами, использующий три ключа: K_A, K_B и K_C, распределение которых показано в 1-й.

Алиса может зашифровать сообщение ключом K_A так, что Эллен может расшифровать его, используя ключи K_B и K_C. То же самое, сговорившись, могут сделать Боб и Кэрол. Боб может зашифровать сообщение так, что Фрэнк сможет прочесть его а Кэрол сможет зашифровать сообщение так, что его сможет прочесть Дэйв. Дэйв может зашифровать сообщение ключом K_A так, что Эллен сможет прочесть его, ключом K_B так, что его сможет прочесть Фрэнк, или обоими ключами, K_A и K_B, так, что сообщение сможет прочесть Кэрол. Аналогично, Эллен может зашифровать сообщение так, что Алиса, или Дэйв, или Фрэнк сможет прочесть его. Все возможные комбинации показаны в , других не существует.

Табл. 3-2.
Распределение ключей в трехключевой системе.

Алиса	K_A
Боб	K_B

Кэрол	K_C
Дэйв	K_A и K_B
Эллен	K_B и Kc
Франк	K_C и K_A

Такая схема может быть расширена на n ключей. Если для шифрования сообщения используется заданное подмножество ключей, то для дешифрирования сообщения потребуются оставшиеся ключи .

Широковещательная передача сообщения

Представьте, что в некоей операции занято 100 ваших тайных агентов . Вы хотите иметь возможность посылать сообщения группам агентов, но вы не знаете заранее состав групп . Можно либо шифровать сообщение отдельно для каждого корреспондента, либо распределить ключи для всех возможных комбинаций агентов . Для реализации первого способа потребуется множество сообщений, для второго - множество ключей .

Криптография с несколькими ключами позволяет решить эту задачу намного проще. Мы будем использовать трех агентов: Алису, Боба и Кэрол. Вы выдадите Алисе ключ K_A и K_B, Бобу - K_B и K_C, Кэрол - K_C и K_A. Теперь вы сможете говорить с любым нужным подмножеством агентов. Если вы хотите, чтобы сообщение могла прочитать только Алиса, зашифруйте его ключом K_C. Когда Алиса получит сообщение, она расшифрует его, последовательно используя ключи K_A и K_B. Если вы хотите послать сообщение только Бобу, зашифруйте его ключом K_A, а сообщение для Кэрол - ключом K_B. Если вы хотите, чтобы посланное сообщение могли прочитать Алиса и Боб, зашифруйте его ключами K_A и K_C.

Для трех агентов это не слишком впечатляет, но для 100 преимущество достаточно ощутимо. Индивидуальные сообщения означают использование отдельного ключа для каждого агента (всего 100 ключей) и каждого сообщения. Передача сообщений всем возможным подмножествам означает использование 2^{100}-2 различных ключей (исключены случаи сообщения всем агентам и никому из них) . Для схемы, использующий криптографию с несколькими открытыми ключами, нужно только одно шифрованное сообщение и сто различных ключей . Недостатком этой схемы является то, что вам также придется широковещательно передавать, какое подмножество агентов может читать сообщение, иначе каждому их них придется перебирать все возможные комбинации ключей в поисках подходящей. Даже только перечисление имен получателей может быть весьма внушительным. Кроме того, каждому агенту придется хранить немаленький объем информации о ключах, по крайней мере при прямолинейной реализации этой схемы .

Существуют и другие способы широковещательной передачи , ряд из них позволяет избежать описанной проблемы. Эти способы обсуждаются в разделе 22.7.

Табл. 3-3.
Шифрование сообщения в трехключевой системе.

Шифруется ключами	Должно быть расшифровано ключами
K_A	K_B и Kc
K_B	K_A и K_C
K_C	K_A и K_B
K_A и K_B	K_C
K_A и K_C	K_B
K_B и Kc	K_A

3.6 Разделение секрета

Вообразите, что вы изобрели новый, сверхлипкую, сверхсладкую сливочную тянучку или соус для гамбургеров, который еще безвкуснее, чем у ваших конкурентов. Это очень важно, и вы хотите сохранить изобретение в секрете. Только самым надежным работникам вы можете сообщить точный состав ингредиентов , но вдруг и кто-то из них подкуплен конкурентами? Секрет выкрадут, и немного погодя каждый в квартале будет делать гамбургеры с таким же безвкусным соусом, как ваш.

Предлагаемая схема называется **разделением секрета**. Есть способы взять сообщение и разделить его на части [551]. Каждая часть сама по себе ничего не значит, но сложите их - и вы получите сообщение . Если это

рецепт, и у каждого работника находится только его часть, то лишь собравшись все вместе ваши служащие см огут сделать соус. Если кто-нибудь из работников уволится, прихватив с собой свою часть рецепта, раскрытая информация по себе будет бесполезной.

По простейшей схеме сообщение делится между двумя людьми. Вот протокол, используя который Трент д елит сообщение между Алисой и Бобом:

(1) Трент генерирует строку случайных битов, R, такой же длины, что и сообщение, M.

(2) Трент выполняет "исключающее или" (XOR) над M и R, создавая S.

$$R \oplus M = S$$

(3) Трент передает Алисе R, а Бобу - S.

Чтобы получить сообщение, Алисе и Бобу нужно выполнить единственное действие:

(4) Алиса и Боб выполняют операцию над имеющимися у них частями, восстанавливая сообщение.

$$R \oplus S = M$$

Этот метод при правильном выполнении абсолютно безопасен. Каждая часть в отдельности абсолютно бе ссмысленна. Что существенно, Трент шифрует сообщение одноразовым блокнотом и дает шифротекст одному человеку, а блокнот - другому. Одноразовые блокноты, обладающие абсолютной безопасностью, обсуждаются в разделе 1.5. Никакие вычислительные средства не смогут восстановить сообщение только по одной его части.

Эту схему легко расширить на большее число людей. Чтобы разделить сообщение между более чем двумя людьми, выполните операцию XOR с большим числом строк случайных битов. В следующем примере Трент делит сообщение на четыре части:

(1) Трент генерирует три строки случайных битов, R, S и T, такой же длины, что и сообщение, M.

(2) Трент выполняет "исключающее или" (XOR) над M и созданными тремя строками, создавая U.

$$M \oplus R \oplus S \oplus T = U$$

(3) Трент передает Алисе R, Бобу - S, Кэрол - T, а Дэйву - U.

Вместе Алиса, Боб, Кэрол и Дэйв могут восстановить сообщение:

(4) Алиса, Боб, Кэрол и Дэйв собираются вместе и вычисляют:

$$R \oplus S \oplus T \oplus U = M$$

Это арбитражный протокол. Трент обладает абсолютной властью и может делать все, что он хочет. Он может раздать чепуху и утверждать, что это настоящие части секретной информации, никто не сможет это пров ерить, пока, собравшись вместе, участники протокола не попробуют прочитать письмо. Он может выдать части секрета Алисе, Бобу, Кэрол и Дэйву и позже заявить всем, что только Алиса, Кэрол и Дэйв нужны для восст ановления секрета, застрелив при этом Боба. Но это не является проблемой, так как делимый секрет принадл ежит Тренту.

Однако, одна проблема у этого протокола существует. Если любая из частей будет потеряна, а Трента не б удет поблизости, пропадет и все сообщение. Если Кэрол, обладая частью рецепта соуса, перейдет работать к ко нкуренту, оставив свою часть секрета у себя, значит, остальным не повезло. Она не сможет восстановить рецепт, но не смогут, собравшись, и Алиса, Боб и Дэйв. Ее часть также критична для восстановления сообщения, как и любая другая. Все, что известно Алисе, Бобу и Дэйву - это длина сообщения, и ничего больше. Это истинно, так как у R, S, T, U и M одинаковая длина, следовательно, каждому из участников известна длина M. Помните, сообщение M делится не в обычном смысле этого слова, а подвергается операции XOR со случайными величинами.

3.7 Совместное использование секрета

Вы вводите программу запуска ядерной ракеты и хотите быть уверенным, что никакой псих в одиночку не сможет вызвать пуск. Вы хотите быть уверенным, что и два психа не смогут вызвать пуск. Вы хотите, чтобы пуск произошел только, если не меньше трех из пяти офицеров будут психами.

Эта проблема легко может быть решена. Сделайте механическое устройство контроля запуска. Выдайте ключ каждому из пяти офицеров и потребуйте, чтобы по меньше мере три офицера вставили свои ключи в соо тветствующие гнезда, прежде чем вы разрешите им взорвать того, кого мы взрываем на этой неделе. (Если вы действительно волнуетесь, сделайте гнезда подальше друг от друга и потребуйте, чтобы офицеры вставляли ключи одновременно - вы ведь не хотели бы, чтобы офицер, выкравший недостающую пару ключей, смог бы испепелить Толедо.)

Можно сделать еще сложнее. Пусть только генерал у и паре полковников разрешено запустить ракету, но е сли генерал занят игрой в гольф, то запустить ракету имеют право только пять полковников . Сделайте контрольное устройство с пятью ключами. Выдайте генералу три ключа, а полковникам по одному. Генерал вместе с двумя полковниками или пять полковников смогут запустить ракету. Однако ни генерал в одиночку, ни четыре полковника не смогут этого сделать.

Более сложная схема совместного использования, называемая **пороговой схемой**, может решить и эти задачи, и более сложные - математически. На ее простейшем уровне вы можете взять любое сообщение (секретный рецепт, коды запуска, ваш список для прачечной и т.п.) и разделить его на *n* частей, называемых **тенями** или долями, так, что по любым *m* из них можно восстановить сообщение. Более точно, это называется **(m,n)-пороговой схемой**.

Используя (3,4)-пороговую схему, Трент может разделить свой секретный рецепт между Алисой, Бобом, К эрол и Дэйвом так, что любые трое из них могут сложить свои тени вместе и восстановить сообщение. Если Кэрол в отпуске, то Алиса, Боб и Дэйв смогут восстановить сообщение. Если Боб попал под автобус, то сообщение смогут восстановить Алиса, Кэрол и Дэйв. Но если Боб попал под автобус, а Кэрол в отпуске, то Алиса и Дэйв самостоятельно не смогут восстановить сообщение.

Вообще, пороговые схемы могут быть еще более гибкими. Можно отмоделировать любые сценарии совм естного использования, которые вы только сможете вообразить. Можно разделить сообщение между людьми в вашем здании так, что для его восстановления, если нет никого с третьего этажа, потребуется семь человек с первого этажа и пять со второго, в противном случае достаточно представителя третьего этажа вместе с тремя человеками с первого этажа и двумя со второго. Если же есть кто-то с четвертого этажа, то для восстановления сообщения достаточно этого человека и одного с третьего этажа или этого человека вместе с двумя с первого этажа и одного со второго. Если же ... ну вы уловили идею.

Эта идея была независимо выдвинута Ади Шамиром [1414] и Джорджем Блэкли (George Blakley) [182] и интенсивно была изучена Гусом Симмонсом (Gus Simmons) [1466]. Множество различных алгоритмов обсуждается в разделе 23.2.

Совместное использование с мошенниками

Существует множество способов обмануть пороговую схему. Вот только несколько из них. Сценарий 1 : Полковники Алиса, Боб и Кэрол сидят в изолированном бункере где-то глубоко под землей. Однажды они получают закодированное сообщение от президента: "Запустить ракеты. Мы собираемся стереть с лица Земли любые сл еды исследований противника в области нейронных сетей". Алиса, Боб и Кэрол открывают свои тени, но Кэрол вводит случайное число. Он на самом деле пацифист и не хочет, чтобы ракеты были запущены. Поскольку Кэрол не ввела правильное тени, секретная информация, которую они хотели получить, оказалась неправильной. Ракеты остались в своих шахтах. И самое плохое, никто не знает почему. Даже объединившись Алиса и Боб не смогут доказать, что тень Кэрол неправильна.

Сценарий 2: Полковники Алиса и Боб сидят в бункере вместе с Мэллори. Мэллори ложно выдает себя за полковника. От президента приходит то же самое сообщение и все открывают свои тени. "Ха-ха-ха!" кричит Мэллори. "Я подделал это сообщение президента. Теперь я знаю обе ваших доли." Он убегает вверх по лестнице и исчезает прежде, чем его успеют поймать.

Сценарий 3: Полковники Алиса, Боб и Кэрол сидят в бункере вместе с Мэллори, который снова замаскировался. (Помните, у Мэллори нет правильной тени.) От президента приходит то же самое сообщение и все о ткрывают свои тени. Мэллори открывает свою тень, только услышав все остальные. Так как для восстановления секрета требуется только три тени, он может быстро создать правильную тень и открыть ее. Итак, он не только заполучил секрет, но и никто не догадался, что он не является частью этой системы. Некоторые протоколы, которые позволяют бороться с подобными мошенниками, рассматриваются в разделе 23.2.

Совместное использование секрета без Трента

Банк хочет, чтобы его подвал могли открыть трое из пяти офицеров, введя свои ключи. Это выглядит как типичная (3,5)-пороговая схема, но с одной тонкостью. Никто не знает секрета целиком. Трента, которые делит секрет на пять частей, нет. Существуют протоколы, используя которые пять офицеров могут создать секрет и поделить его на части так, что никто из офицеров не узнает секрета, пока он не будет восстановлен . В этой книге я не рассматриваю эти протоколы, подробности см. в [756].

Совместное использование секрета без раскрытия долей

У этих схем есть одна проблема. Когда участники протокола собираются, чтобы восстановить секрет, они открывают свои части. Но раскрытие секрета не всегда желательно. Если разделяемый секрет является закр ытым ключом (например, к цифровой подписи), то каждый из *n* участников может выполнить частичную подпись

документа. После *n*-ой частичной подписи документ оказывается подписан совместно используемым закрытым ключом, а ни один из участников не может узнать содержания части, используемой другим участником . Смысл в том, что вы можете повторно использовать секрет, и для работы с ним вам не понадобится надежный посре д-ник. Дальнейшее развитие эта идея получила в работах Иво Десмедта (Yvo Desmedt) и Йера Френкеля (Yair Frankel) [483, 484].

Подтверждаемое совместное использование секрета

Трент передает Алисе, Бобу, Кэрол и Дэйву часть секрета (или, по крайней мере, заявляет, что он это делает). Единственный способ убедиться, что их части правильны - это попытаться восстановить секрет . Может быть Трент послал Бобу поддельную часть, или часть Боба случайно испортилась при передаче по линиям связи. Подтверждаемое совместное использование секрета позволяет каждому из участников лично убедиться, что их часть правильна, без необходимости восстанавливать секрет [558, 1235].

Схемы совместного использования секрета с мерами предохранения

Секрет делится среди 50 человек так, чтобы любые 10 могли собраться вместе и восстановить секрет . Это нетрудно. Но, можем ли мы реализовать ту же схему совместного использования секрета, добавив требование, чтобы 20 человек могли собраться вместе и *помешать* остальным, независимо от их числа, восстановить секрет? Оказывается, что да [153].

Математика достаточно сложна, но основная идея в том, что каждый получает две части : часть "да" и часть "нет". Когда приходит время восстановить секрет, люди предоставляют одну из своих частей . Какую конкретно зависит от того, хотят ли они, чтобы секрет был раскрыт. Если предоставлено *m* или больше долей "да" и меньше чем *n* долей "нет", то секрет может быть восстановлен. В противном случае, это невозможно.

Конечно же, ничего не мешает достаточному числу людей "да" отойти в уголок, уединившись от людей "нет" (если они знают, кто есть кто) и восстановить секрет. Но при условии, что все передают свои части в централ ь-ный компьютер эта схема будет работать.

Совместное использование секрета с вычеркиванием из списка

Вы создали систему совместного использования секрета и теперь хотите застрелить одного из владельцев части секрета. Вы могли бы создать новую схему, исключив этого несчастного, но время поджимает. Для п о-добной системы существуют способы копирования. Они позволяют активизировать новую схему совместного использования секрета сразу же после того, как вы перестали доверять одному из участников [1004].

3.8 Криптографическая защита баз данных

База данных членов организации - это весьма важная вещь. С одной стороны вы хотите предоставить к не доступ всем членам, желая, чтобы они общались друг с другом, обменивались идеями и делились друг с другом бутербродами. С другой стороны, если вы пустите в вашу базу данных кого угодно, сведения обязательно поп а-дут в руки надоедливых страховых агентов и докучливых поставщиков всякого хлама по почте .

Криптография может облегчить эту проблему. Можно зашифровать базу данных так, чтобы получить адрес одного человека было легко, а извлечь список почтовых адресов всех членов - трудно .

Схема, предложенная в [550, 549], прямолинейна. Выберите однонаправленную хэш-функцию и симметри ч-ный алгоритм шифрования. У каждой записи в базе данных два поля. Индексным полем является фамилия чл е-на, и именно оно обрабатывается однонаправленной хэш-функцией . Поле данных, в котором хранится полное имя и адрес члена, шифруется с помощью используемой в качестве ключа фамилии . Если вы не знаете фам и-лии, вы никогда не сможете расшифровать поле данных .

Поиск по конкретной фамилии прост. Сначала хэшируется фамилия, и выполняется поиск значения хэш-функции в базе данных. Наличие нескольких совпадений означает, что база данных содержит информацию о нескольких людях с такой фамилией.

В [550] авторы используют эту систему для защиты словаря из 6000 испанских слов. Они сообщают о том, что потеря производительности, вызванная шифрованием, минимальна . В более сложной схеме [549] использ у-ется поиск по нескольким индексам, но идея остается той же. Основная проблема, связанная с этой системой, состоит в том, что вы не сможете найти человека, не зная, как пишется его фамилия . Можно попробовать не-сколько вариантов, пока не будет найден правильный, но неудобно перебирать всех, чьи фамилии начинаются на "Sch" при поиске "Schneier."

Эта защита несовершенна. Очень назойливый страховой агент восстановит базу данных членов организации с помощью грубого взлома, перебирая все возможные фамилии . Если у него есть телефонная база данных, он может использовать имеющийся в ней список фамилий. Это пережевывание номеров может занять несколько

недель, но дело будет сделано. Тем не менее такая схема усложнит работу взломщика (в мире продаже всякой чепухи по почте "усложнит" быстро превращается в "сделает слишком дорогой". Другой подход, предложенный в [185], предлагает набирать статистику по шифрованным данным .

Глава 4
Промежуточные протоколы

4.1 Службы меток времени

Во многих ситуациях людям нужно убедиться, что определенный документ уже существовал в определенный момент времени. Примером является спор об авторских правах или патенте. Дело выигрывает сторона, которая представит более раннюю копию спорной работы. Бумажные документы заверяются нотариусами и хранятся у юристов. Если возникает спор, нотариус или юрист свидетельствует, что письмо существовало в определенный момент времени.

В цифровом мире все гораздо сложнее. Нет способов обнаружить признаки подделки электронного докуме нта. Его можно бесконечно копировать и изменять, не оставляя никаких следов. Несложно и изменить время со здания компьютерного файла. Никто не может взглянуть на документ и с полной уверенностью сказать: "Да, этот документ был создан раньше 4 ноября 1952 года"

Этой проблемой задались Стюарт Хабер (Stuart Haber) и В. Скотт Сторнетта (W. Scott Stornetta) из Bellcore [682, 683, 92]. Им потребовался протокол цифровых меток времени со следующими свойствами:

— Метка времени должна существовать сама по себе, независ ия от физической среды, используемой для ее хранения.

— Должно быть невозможно тайно изменить ни единого бита документа.

— Должно быть невозможно задать для документа метку времени, отличного от текущего.

Решение с посредником

В этом протоколе участвуют Трент, обладающий надежной службой меток времени, и Алиса, которая хочет задать метку времени для документа.

(1) Алиса передает копию документа Тренту.

(2) Трент записывает время и дату получения документа, оставляя у себя копию для безопасного хранения.

Теперь, если кто-нибудь усомнится в заявленном Алисой времени создания документа, то Алисе просто нужно обратиться к Тренту. Он предоставит свою копию документа и подтвердит, что он получил документ в указанный день и час.

Этот протокол работает, но есть ряд очевидных проблем. Во первых, невозможно сохранить тайну - Алиса должна предоставить копию документа Тренту. Кто-то, подслушивающий линию связи, сможет прочесть док умент. Она может зашифровать документ при передаче, но ведь он должен будет храниться в базе данных Тре нта. Насколько эта база безопасна?

Во вторых, самой базе данных придется быть очень большой. Велики будут требования и к пропускной сп особности линии связи.

Третья проблема связана с возможными ошибками. Ошибки при передаче или электромагнитная бомба, взорванная где-то в центральном компьютере Трента могут полностью свести на нет заявление Алисы о метке времени.

И в четвертых, может оказаться невозможным найти такого честного Трента для ведения службы меток времени. Может быть, Алиса использует метку времени Боба. Ничто не остановит Алису и Боба от сговора и пометки документа тем временем, которое им нужно.

Улучшенное решение с посредником

Большинство этих проблем легко снимаются при использовании однонаправленной хэш-функции и цифр овых подписей:

(1) Алиса вычисляет значение однонаправленной хэш-функции для документа.

(2) Алиса передает это значение Тренту.

(3) Трент добавляет время и дату получения этого значения и затем подписывает результат цифровой подп исью.

(4) Трент отправляет подписанное значение хэш-функции вместе с меткой времени Алисе.

Это решает все проблемы, кроме последней. Алисе больше не нужно беспокоиться о раскрытии содержания

документа, использование значения хэш-функции вполне достаточно. Тренту больше не нужно хранить копии документов (и даже значения хэш-функции), поэтому снимаются проблемы безопасности и объема сохраняемых данных (помните, у однонаправленных хэш-функций нет ключа). Алиса может немедленно проверить подписанную метку времени, полученную на этапе (4), и немедленно обнаружить любые ошибки передачи. Единственной оставшейся проблемой остается сговор Алисы и Трента с целью создания поддельной метки времени.

Протокол связи

Одним из путей решения этой проблемы является установление связи между меткой времени Алисы и метками времени, ранее созданными Трентом. Весьма вероятно, что эти метки были созданы не для Алисы, а для других людей. Так как порядок, в котором Трент получает различные запросы о метках времени не может быть известен заранее, перед меткой времени для Алисы должна была появиться другая метка времени. И так как запрос, пришедший позже, связан с меткой времени Алисы, то ее метка должна была появиться раньше. Эти две метки содержат между собой запрос Алисы как будто в сэндвиче.

Если A - это имя Алисы, H_n - значение хэш-функции, для которого Алиса хочет зафиксировать время, а T_{n-1} - предыдущая метка времени, то протокол имеет следующий вид:

(1) Алиса посылает Тренту H_n и A.

(2) Трент посылает Алисе обратно:

$T_n=S_K(n,A,H_n,T_n,I_{n-1},H_{n-1},T_{n-1},L_n)$

где состоит L_n - это информация о следующей хэшированной связи:

$L_n =H(I_{n-1},H_{n-1},T_{n-1},L_{n-1})$

S_K указывает, что сообщение подписано открытым ключом Трента. Имя Алисы определяет ее как отправителя запроса. Параметр n указывает последовательность запросов. Это n-ая метка времени, которую создал Трент. Параметр T_n - это время. Дополнительно используется информация об идентификаторе, оригинального значения хэш-функции, времени и хэшированной метка предыдущего документа, помеченного Трентом.

(3) Когда Трент помечает следующий документ, он посылает Алисе идентификатор отправителя этого документа: I_{n-1}.

Если кто-то оспаривает метку времени Алисы, ей надо только связаться с отправителями предыдущего и следующего документов: I_{n-1} и I_{n+1}. Если и их свидетельство под вопросом, можно обратиться к авторам документов I_{n-1} и I_{n+1} и т.д. Любой может показать, что его документ был помечен после одного документа и перед другим.

Этот протокол мешает Алисе и Тренту договориться и создать документ с временем создания, отличным от времени создания настоящего документа. Трент не может изменить дату документа Алисы на более раннюю, так как для этого нужно знать заранее, для какого документа перед данным будет проставляться метка времени. Даже если он сможет подделать предыдущий документ, ему придется знать, какой документ предшествовал предыдущему и так далее. Трент не может изменить дату документа Алисы и на более позднюю, потому что метка времени должна быть вставлена перед меткой времени документа, заверяемого сразу же после данного, а этот документ уже существует. Единственный возможный способ сломать эту схему - это ввести фиктивную цепочку документов перед и после документа Алисы, достаточно длинную, чтобы лишить терпения того, кто проверяет метку времени документа Алисы.

Распределенный протокол

Люди умирают, метки времени теряются. Между пометкой документа и его оспариванием может произойти многое, что помешает Алисе получить копию метки времени I_{n-1}. Эта проблема может быть частично снята вставкой меток времени предыдущих 10 человек в метку Алисы и последующей передаче Алисе имен следующих 10 человек. Так у Алисы появится гораздо больше возможностей найти людей, все еще хранящих свои метки времени.

Развивая эту идею, следующий протокол позволяет обойтись и без Трента:

(1) Используя в качестве входа H_n, Алиса генерирует последовательность случайных чисел с помощью криптографически безопасного генератора случайных чисел.

$V_1, V_2, V_3, \ldots V_k$

(2) Алиса рассматривает каждое из этих чисел как идентификатор, I, другого человека и посылает каждому из этих людей H_n.

(3) Каждый из них добавляет время и дату к значению хэш-функции, подписывает результат и отправляет его

обратно Алисе.

(4) Алиса собирает и хранит все подписи как метку времени.

Криптографически безопасный генератор случайных чисел, используемый на этапе (1), позволяет Алисе избежать преднамеренного выбора коррумпированных I в качестве свидетелей. Даже если она сделает простейшие изменения в своем документе, пытаясь создать набор коррумпированных I, ее шансы добиться этого пренебрежимо малы. Хэш-функция рандомизирует значения, и Алиса не может на них воздействовать.

Этот протокол работает, потому что подделать метку времени Алиса может, только договорившись о сотрудничестве со всеми k людьми. Так как на этапе (1) она выбирала их случайным образом, вероятность этого очень низка. Чем коррумпированнее общество, тем больше должно быть число k.

Кроме того, должен использоваться некоторый механизм, учитывающий то, что ряд людей не смогут вовремя возвратить метку времени. Все, что нужно для правильной метки времени - это некоторое подмножество k. Детали зависят от реализации.

Дальнейшая работа

Дальнейшие улучшения протоколов метки времени описаны в [92]. Авторы используют двоичные деревья для увеличения количества меток времени, зависящих от данной метки, уменьшая вероятность создания цепочки фальшивых меток времени. Они также рекомендуют публиковать список значений хэш-функций за прошедший день в некотором общедоступном источнике, например газете. Это работает как отправка значения хэш-функции случайным людям в распределенном протоколе. Действительно, метка времени появляется в каждом номере воскресной *Нью-Йорк Таймс* с 1992 года.

Эти протоколы меток времени запатентованы [684, 685, 686]. Патенты принадлежат дочерней компании Bellcore, названной Surety Technologies, которая продает Систему цифрового нотариата, поддерживающую эти протоколы. В первой версии клиенты посылали запросы о "заверении" на центральный координирующий центр. Следуя методике Меркла по использованию хэш-функций для построения деревьев [1066], сервер строит дерево значений хэш-функции, листья которого представляют собой все запросы, полученные в течение данной секунды, и посылает каждому автору запроса список значений хэш-функции, описывающий путь от его листа до корня. Клиентская часть программного обеспечения сохраняет этот список и может выдать "сертификат" Цифрового нотариата для любого файла, который был сертифицирован. Последовательность корней этих деревьев образует "Запись универсального подтверждения" ("Universal Validation Record"), которая будет доступна в электронном виде во многих хранилищах (и также выпущена на CD-ROM). Клиентская часть также содержит функцию "подтверждения", позволяющую пользователю проверить, был ли заверена именно текущая форма файла (запросив из хранилища корень соответствующего дерева и сравнив его со значением хэш-функции, соответствующим образом рассчитанным для файла, и сертификатом). За дальнейшей информацией обращайтесь в Surety Technologies, 1 Main St., Chatham, NJ, 07928; (201) 701-0600; Fax: (201) 701-0601.

4.2 Подсознательный канал

Алиса и Боб были арестованы и отправлены в тюрьму, он - в мужскую, а она - в женскую. Уолтер, надзиратель, разрешает Алисе и Бобу обмениваться сообщениями, но он не разрешает шифровать сообщения. Уолтер считает, что они планируют бегство, поэтому он хочет читать все, что они пишут.

Уолтер надеется также суметь обмануть Алису или Боба. Он хочет, чтобы один из них посчитал принятое им ложное сообщение настоящим. Алиса и Боб мирятся с риском возможного обмана, иначе они вообще не смогут общаться, но им нужно согласовать свои планы. Для этого им необходимо обмануть надзирателя и найти способ передавать секретную информацию. Им нужно создать подсознательный канал, скрытый канал связи в открытых сообщениях, хотя сообщения сами по себе не содержат секретной информации. С помощью обмена совершенно безобидными подписанными сообщениями они обменяются секретной информацией и одурачат Уолтера, даже если он просматривает все сообщения.

Простым подсознательным каналом может быть число слов в предложении. Нечетное число слов в предложении может соответствовать "1", а четное число слов -"0". Так, пока вы читаете этот самый обычный абзац, я передал вам сообщение "110". Проблематичность этого метода в том, что он является обычной стеганографией (см. раздел 1.2), ключ не используется и безопасность зависит от секретности алгоритма.

Густавус Симмонс придумал идею организации подсознательного канала с помощью обычного алгоритма цифровой подписи [1458, 1473]. Так как подсознательные сообщения спрятаны в том, что выглядит нормальными цифровыми подписями, это форма маскировки. Уолтер видит, как подписанные безобидные сообщения передаются туда и обратно, но реальная передаваемая информация проходит незаметно для него по подсознательному каналу. В действительности, алгоритм подсознательного канала в подписях не отличим от нормального алгоритма в подписях, по крайней мере для Уолтера. Ер не только не может прочитать сообщение, передаваемое по подсознательному каналу, но у него вообще нет ни малейшего представления о существовании такого

сообщения. В общем случае протокол выглядит примерно так:

(1) Алиса создает безобидное сообщение, все равно какое.

(2) Используя общий с Бобом ключ, Алиса подписывает безобидное сообщение, пряча свое подсознательное сообщение в подписи. (Это суть подсознательного протокола, см. раздел 23.3).

(3) Алиса посылает подписанное сообщение Бобу через Уолтера.

(4) Уолтер читает безобидное сообщение и проверяет подпись. Не обнаружив ничего подозрительного, он передает подписанное сообщение Бобу.

(5) Боб проверяет подпись под безобидным сообщением, убеждаясь, что сообщение получено от Алисы.

(6) Боб игнорирует безобидное сообщение и, используя общий с Алисой секретный ключ, извлекает подсознательное сообщение.

А мошенничество? Уолтер не верит никому, и никто не верит Уолтеру. Он всегда может помешать передаче сообщений, но у него нет возможности подделать сообщение. Так как Уолтер не может создать правильной подписи, Боб обнаружит подделку на этапе (5). Уолтер не может читать подсознательные сообщения - у него нет нужного ключа. Что еще важнее, у него нет ни малейшего представления, что подсознательные сообщения существуют. Подписанные сообщения, использующие алгоритм цифровой подписи на вид ничем не отличаются от подписанных сообщений, содержащих подсознательные сообщения в подписи.

Более проблематичен обман своего партнера Алисой или Бобом. В некоторых реализациях подсознательного канала секретная информация, нужная Бобу для чтения подсознательного сообщения, совпадает с информацией, нужной Алисе для подписи безобидного сообщения. Если это так, Боб может выдать себя за Алису. Он может подписать сообщения, выдав их за посланные Алисой, и Алиса ничего не сможет с этим поделать. Если ей необходимо отправить ему подсознательное сообщение, она должна верить, что он не будет мошенничать с ее закрытым ключом.

В других реализациях подсознательного канала такой проблемы нет. Секретный ключ, общий для Алисы и Боба, позволяет Алисе отправлять Бобу подсознательные сообщения, но закрытый ключ Алисы не передается, и Боб не может подписывать сообщения ее подписью. Алисе не нужно верить, что Боб не будет мошенничать с ее закрытым ключом.

Применения подсознательного канала

Наиболее очевидным применением подсознательного канала является шпионская сеть. Если кто-то посылает и принимает сообщения, то передача сообщений по подсознательному каналу в подписанных документах не будет вызывать подозрений. Конечно же, вражеские шпионы могут делать то же самое.

Используя подсознательный канал, Алиса может, даже если ей угрожают, безопасно подписать документ. Подписывая документ, она может вставить подсознательное сообщение, написав: "Я арестована ". Иные применения не так бросаются в глаза. Компания может подписать документы и вставить подсознательные сообщения для отслеживания времени действия документов. Правительство может "пометить" электронные деньги. Мошенническая программа для подписи документов может использовать подсознательные сообщения в создаваемых подписях для организации утечки секретной информации. Возможности бесконечны.

Подписи, свободные от подсознательного канала

Алиса и Боб обмениваются подписанными сообщениями, обговаривая сроки контракта. Они используют протокол цифровой подписи. Однако, эти переговоры на самом деле маскируют шпионскую деятельность Алисы и Боба. Используя алгоритм цифровой подписи, они не волнуются о подписываемых ими сообщениях. Для обмена секретной информацией они используют подсознательный канал в подписях под документами. Контрразведка, однако, не знает, что переговоры о контракте и используемые подписанные сообщения являются только прикрытием. Для противодействия подобной схеме были разработаны схемы подписи, свободной от подсознательного канала. Используемые в этих схемах цифровые подписи невозможно изменить для организации подсознательного канала. Подробности см. в [480, 481].

4.3 Неотрицаемые цифровые подписи

Обычные цифровые подписи могут быть точно скопированы. Иногда это свойство полезно, например, при распространении публичных заявлений. В другой раз это может оказаться проблемой. Вообразите личное или деловое письмо, подписанное цифровой подписью. Если распространяется множество копий этого документа, каждая из которых может быть проверена кем угодно, то это может привести к замешательству или шантажу. Лучшим решением является цифровая подпись, правильность которой может быть доказана получателю, но которая не позволит получателю показать третьей стороне полученное сообщение без согласия разрешения ли-

ца, подписавшего сообщение.

Alice Software Company (Компания программного обеспечения Алисы) распространяет продукт DEW (Do-Everything-Word, Делая со словом что угодно). Для гарантии отсутствия вирусов каждая копия содержит цифровую подпись. Однако, создатели хотят, чтобы только легальные покупатели продукта, а не компьютерные пираты могли проверить подпись. В то же время, если обнаруживаются копии DEW, содержащие вирус, у Alice Software Company не должно быть возможности отрицать правильную подпись.

Неотрицаемые подписи [343,327] удобны для решения подобных задач. Как и обычная цифровая подпись, неотрицаемая цифровая подпись зависит от подписанного документа и закрытого ключа человека, подписавшего документ. Но, в отличие от обычных цифровых подписей, неотрицаемая подпись не может быть проверена без разрешения подписавшего. Хотя для этих подписей можно было бы подобрать название получше, например, "непередаваемые подписи", существующее название обусловлено тем обстоятельством, что если Алисе придется либо подтвердить, либо отрицать подпись - может быть в суде - она не сможет ложно отрицать свою настоящую подпись. Несмотря на сложность математики основная идея проста:

(1) Алиса предъявляет Бобу подпись.

(2) Боб создает случайное число и посылает его Алисе.

(3) Алиса выполняет вычисления, используя случайное число и свой закрытый ключ, и посылает Бобу результат. Алиса может выполнить эти вычисления только, если подпись правильна.

(4) Боб проверяет это.

Также существует дополнительный протокол, позволяющий Алисе доказать, что она не подписывала документ, и не допускающий возможности ложно отказаться от подписи.

Боб не может повернуться и убедить Кэрол, что подпись Алисы правильна, потому что Кэрол не знает, что числа Боба случайны. Он может легко без помощи Алисы изложить протокол на бумаге и послать результат Кэрол. Кэрол может удостовериться в правильности подписи Алисы только, если она сама выполнит этот протокол с Алисой. Сейчас кажется, что в этом немного смысла, но он появится, когда вы взглянете на математику раздела 23.4.

Это решение не совершенно. Иво Десмедт и Моти Юнг (Moti Yung) показали, что в некоторых случаях Боб может убедить Кэрол в правильности подписи Алисы [489].

Например, Боб покупает легальную копию DEW. Он может подтвердить подпись под программным продуктом, когда захочет. Тогда, Боб может убедить Кэрол, что он работает на Alice Software Company, и продать ей пиратскую копию DEW. Когда Кэрол попытается подтвердить подпись Боба, он одновременно подтверждает подпись у Алисы. Когда Кэрол посылает ему случайное число, он отправляет его Алисе. Ответ Алисы он пересылает Кэрол. Кэрол убеждается в том, что она - легальный покупатель, хотя она таковым не является. Такое вскрытие является прмером проблемы великого гроссмейстера и подробно рассматривается в разделе 5.2.

Несмотря на это у неотрицаемых подписей множество применений, во многих случаях Алиса не хочет, чтобы кто угодно мог проверить ее подпись. Она может не хотеть, чтобы подпись под ее личной корреспонденцией могла быть проверена журналистами, чтобы ее письма были опубликованы и подтверждены независимо от контекста, или просто, чтобы нельзя было обнаружить изменения в письмах, сделанные ею позже. Если она подписывает информацию, которую она продает, то она не хочет, чтобы кто-то, не заплатив за информацию, мог подтвердить ее достоверность. Защитить свои права Алиса может контролируя тех, кто проверяет ее подпись.

Ряд вариантов неотрицаемых подписей отделяет связь между подписавшим и сообщением от связи между подписавшим и подписью [910]. В одной схеме кто угодно может проверить, что подпись действительно была создана ее автором, а для проверки правильности подписи для данного сообщения требуется сотрудничество подписавшего.

Близким понятием является **доверительная неотрицаемая подпись** [1229]. Представьте, что Алиса работает на Toxins, Inc., и передает обличающие документы в газету, используя протокол неотрицаемой подписи. Алиса может подтвердить свою подпись только репортеру газеты и никому больше. Однако, негодяй Боб подозревает, что источником документов является Алиса. Он требует, чтобы Алиса использовала протокол снятия подписи, чтобы очистить свое имя, а Алиса отказывается. Боб настаивает, что единственной причиной отказа Алисы является ее виновность, и убивает ее.

Доверительные неотрицаемые подписи похожи на обычные неотрицаемые подписи за исключением протокола снятия подписи, который может быть запущен только Трентом. Только Трент, а не Боб может потребовать от Алисы использовать протокол снятия. И если Трент представляет судебную систему, то он использует этот протокол только для разрешения формального спора.

4.4 Подписи уполномоченного свидетеля

Alice Software Company добилась бурного роста продаж, продавая DEW - такого, что Алиса большую часть времени посвящает подтверждению неотрицаемых подписей, а не работе над новыми возможностями.

Алисе хотелось бы назначить некоего человека в компании ответственным за подтверждение подписи. Ал иса, ил любой другой программист, сможет подписывать документы с помощью неотрицаемого протокола. Но все подтверждения будут проводиться только Кэрол.

Оказывается, это возможно с использованием **подписи уполномоченного свидетеля** [333,1213]. Алиса может подписать документ, так что Боб убедится, что подпись правильна, но не сможет убедить в этом третье л ицо. В то же время Алиса назначает Кэрол на должность будущего свидетеля своей . Алисе даже не нужно заранее просить разрешения у Кэрол, ей только нужно использовать открытый ключ Кэрол. И Кэрол сможет подтвердить подпись Алисы, если Алиса уехала из города, уволилась, была повышена или умерла .

Подписи уполномоченного свидетеля представляют собой некий компромисс между обычными цифров ыми подписями и неотрицаемыми подписями . Определенно существуют случаи, когда Алиса захочет ограничить число тех, кто может подтвердить ее подпись . С другой стороны, предоставление Алисе полного контроля по дрывает сам институт подписей - Алиса может отказаться сотрудничать и в подтверждении, и в отрицании, она может заявить о потере ключей для подтверждения или отрицания, наконец, она может быть просто недоступна . Подписи уполномоченного свидетеля могут предоставить Алисе защиту, создаваемую неотрицаемой подписью, одновременно не позволяя ей злоупотреблять этой защитой . Алиса даже может предпочесть этот способ : подписи уполномоченного свидетеля могут помешать ложным применениям, защитить ее, если она действительно потеряла свой ключ, выручить, если она в отпуске, в больнице или даже умерла .

Эта идея может иметь различные применения. Например, Кэрол может сделаться государственным нотари усом. Она опубликует в каком-то каталоге свой открытый ключ, и люди получают возможность назначать ее свидетелем своих подписей. Получая небольшую плату за подтверждение подписей, она может жить припева ючи.

Кэрол может быть агентством по охране авторских прав, правительственным агентством или еще какой-нибудь организацией . Этот протокол позволяет отделить людей, подписывающих документы, от людей, которые помогают подтверждать по дписи.

4.5 Подписи по доверенности

Подписи уполномоченного свидетеля позволяют подписавшему назначить кого-то другого для подтвержд ения подписи . Пусть Алиса хочет поехать в деловую поездку в некое место, где нет хорошей компьютерной сети - в африканские джунгли, например. Или она не дееспособна после тяжелой операции . Она ожидает получения важной электронной почты и инструктирует своего секретаря Боба ответить соответствующим образом . Как Алиса может передать Бобу полномочия подписывать сообщения за нее, не передавая ему своего закрытого ключа?

Решением этого являются **подписи по доверенности** [1001]. Алиса передает Бобу полномочия так, чтобы имели место следующие свойства :

— ***Различимость***. Кто угодно может отличить подписи по доверенности от обычных подписей.

— ***Неподдельность***. Только сам подписывающий и назначенный им подписывающий по доверенности м ожет создать правильную подпись по доверенности.

— ***Отличие подписи*** по доверенности. подписывающий по доверенности не может создать правильную подпись по доверенности, которую можно выдать за оригинальную подпись.

— ***Подтверждаемость***. По подписи по доверенности контролер должен убедиться в согласии первон ачального подписывающего с подписанным сообщением.

— ***Идентифицируемость***. Первоначальный подписывающий может определить личность подписывающ его по доверенности по подписи по доверенности.

— ***Неотрицаемость***. Подписывающий по доверенности не может снять им подпись по доверенности, п олученную пользователем

В некоторых случаях требуется строгая форма идентифицируемости - кто угодно должен иметь возможность определить личность подписывающего по доверенности по подписи по доверенности . Схемы подписи по доверенности, основанные на различных схемах цифровой подписи приведены в [1001].

4.6 Групповые подписи

Эта проблема была введена Дэвидом Чаумом (David Chaum) в [330]:

> У компании есть несколько компьютеров, подсоединенных к локальной сети . В каждом отделе компании есть свой принтер (также присоединенный к сети), и только один человек в отделе имеет право печатать на принтере своего отдела. Перед печатью, следовательно, принтер должен проверять, что данный сотрудник работает в этом отделе В то же время, компания хочет обеспечить тайну, имя пользователя не должно раскрываться. Если, однако, кто-то в конце дня обнаружит, что принтер используется слишком часто, у директора должна быть возможность найти того, кто использует принтер не по назначению и послать ему чек.

Решение этой проблемы называется групповой подписью. Групповые подписи обладают следующими свойствами:

— Только члены группы могут подписывать сообщения.

— Получатель подписи может убедиться, что это - правильная подпись группы.

— Получатель подписи не может определить, кто именно из членов группы подписал документ.

— при споре подпись будет раскрыта для определения личности подписавшего.

Групповые подписи с надежным посредником

Следующий протокол использует заслуживающего посредника:

(1) Трент создает большую кучу пар открытый ключ/закрытый ключ и выдает каждому члену группы индивидуальный список уникальных закрытых ключей. Одинаковых ключей в списках нет. (Если в группе n членов, и каждый из них получает m пар ключей, то общее число пар ключей составит $n*m$.)

(2) Трент публикует главный список всех открытых ключей для группы в случайном порядке, сохраняя в секрете, какой ключ кому принадлежит.

(3) Когда член группы хочет подписать документ, он случайным образом выбирает ключ из своего списка.

(4) Когда кто-то хочет убедиться, что подпись принадлежит члену данной группы, он перебирает главный список в поисках подходящего открытого ключа и проверяет подпись.

(5) В случае споров обращаются к Тренту, который знает, какие ключи использует каждый член группы.

Проблема протокола состоит в том, что для него необходим надежный посредник. Трент знает закрытые ключи каждого и может подделывать подписи. Кроме того, должно быть достаточно велико, чтобы помешать попыткам анализа с целью поиска владельца каждого ключа.

Чаум [330] перечислил ряд других протоколов, в некоторых из них Трент не может подделать подписи, а в других от и не нужен вовсе. Еще один протокол [348] не только прячет личность подписывающего, но и позволяет добавлять новых членов в группу . И еще один протокол можно найти в [1230].

4.7 Подписи с обнаружением подделки

Пусть Ева является могучим противником. У нее есть обширные компьютерные сети и залы, набитые компьютерами Крэй, на много порядков более мощных, чем доступные Алисе . Все эти компьютеры днем и ночью пыхтят, пытаясь взломать закрытый ключ Алисы. Наконец - успех. Теперь Ева может выдавать себя за Алису, при желании подделывая ее подпись под документами.

Подписи с обнаружением подделки, введенные Биржитом Пфицманом (Birgit Pfitzmann) и Майклом Уэйднером (Michael Waidner) [1240] предотвращают подобное мошенничество. Если после грубого взлома Ева подделывает подписи Алисы, Алиса сможет доказать подлог . Если Алиса подпишет документ, а потом объявит свою подпись подложной, правда может быть доказана судом .

Основная идея, стоящая за подписями с обнаружением подделки, состоит в том, что для каждому возможному открытому ключу соответствует множество возможных закрытых ключей . Каждый из этих закрытых ключей дает множество различных цифровых подписей . Однако, у Алисы есть только один закрытый ключ, и она может рассчитать только одну подпись. Другие закрытые ключи ей неизвестны.

Ева хочет взломать закрытый ключ Алисы. (Ева также сможет быть Алисой, вычислив для себя второй закрытый ключ.) Она собирает подписанные сообщения и, используя множество своих суперкомпьютеров, пытается раскрыть ключ Алисы. Даже если ей удастся раскрыть подходящий закрытый ключ, таких ключей настолько много, что, скорее всего, она получит иной, чем у Алисы, ключ . Вероятность раскрытия ключа, принадлежащего именно Алисе, настолько мала, что ею можно пренебречь.

Теперь, когда Ева подделает подпись под документом, используя найденный закрытый ключ, подделанная подпись будет отличаться от той подписи, которую поставила бы сама Алиса. При обращении в суд Алиса

предъявит две различных подписи под одним и тем же сообщением и открытый ключ (соответствующий ее закрытому ключу и закрытому ключу, найденному Евой), чтобы доказать подлог. С другой стороны, если Алиса не может предъявить две различные подписи, то подлога не было и Алиса должна отвечать за свою подпись.

Эта схема подписей противостоит взлому Евой подписи Алисы с помощью необычайно мощных вычислительных средств. Она ничего не сможет сделать с более вероятной попыткой Мэллори вломиться в дом Алисы и стащить ее закрытый ключ или с попыткой Алисы подписать документ, а затем "случайно" потерять свой закрытый ключ. Чтобы защититься от упомянутой попытки Мэллори, Алисе стоит купить себе хорошую сторожевую собаку, но подобные рекомендации выходят за рамки криптографии.

Дополнительную теорию и применения подписей с обнаружением подделки можно найти в [1239, 1241, 730, 731].

4.8 Вычисления с зашифрованными данными

Алиса хочет знать решение для некоторой функции $f(x)$ для некоторого конкретного значения x. К несчастью, ее компьютер сломан. Боб хочет вычислить для нее значение $f(x)$, но Алиса не хочет, чтобы Боб знал ее x. Как Алисе позволить Бобу провести вычисление $f(x)$и не сообщить ему x?

Это обычная проблема **вычислений с зашифрованными данными**, также известных как **тайная информация прорицателя**. (Прорицателем является Боб - он отвечает на вопрос.) Для некоторых функций существуют способы решить эту задачу, они обсуждаются в разделе 23.6.

4.9 Вручение битов

Алиса Великолепная, выдающаяся волшебница, сейчас продемонстрирует мощь своего искусства. Она угадает карту, которую выберет Боб до того, как он ее выберет! Следите за тем, как Алиса записывает свое предсказание на кусочке бумаги. Восхищайтесь тем, как Алиса кладет этот кусочек бумаги в конверт и запечатывает его. Дрожите от того, как Алиса отдает запечатанный конверт случайному зрителю. "Выбери карту, Боб, любую карту." Он глядит на нее и показывает карту Алисе и зрителям. Это семерка бубен. Теперь Алиса забирает конверт у зрителя и открывает его. Предсказание, записанное до того, как Боб выбрал карту, сообщает "семерка бубен"! Аплодисменты.

Для успеха этого трюка Алисе нужно подменить конверт в конце фокуса. Однако, криптографические протоколы могут обеспечить защиту от любой ловкости рук. А какая в этом польза? Вот более приземленная история.

Биржевой брокер Алиса хочет убедить инвестора Боба, что ее метод определять перспективные акции заслуживает внимания.

Боб: "Подберите-ка для меня пяток акций. Если на них удастся заработать, я передам свой бизнес вам."

Алиса: "Если я подберу пять акций для вас, вы сможете вложить в них деньги, не заплатив мне. Почему бы мне не показать вам пять акций, которые я подобрал в прошлом месяце?"

Боб: "Откуда я знаю, что вы не подменили результаты вашего выбора, узнав настоящие. Если вы сообщите мне о своем выборе сейчас, я буду уверен, что вы не подмените результат. Я не буду вкладывать деньги в эти акции, пока я не оплачу ваши услуги. Поверьте мне."

Алиса: "Я лучше покажу вам свою подборку акций за прошлый месяц. Я не подменяла их. Поверьте мне."

Алиса хочет передать свое предсказание (т.е., бит или последовательность битов), но не хочет раскрывать свое предсказание до некоторого времени. Боб, с другой стороны, хочет удостовериться, что Алиса не сможет изменить свое мнение после того, как она сделала предсказание.

Вручение битов с помощью симметричной криптографии

Этот протокол вручения битов использует симметричную криптографию:

(1) Боб генерирует строку случайных битов, R, и посылает ее Алисе.

(2) Алиса создает сообщение, состоящее из своего бита, который она хочет вручить, b (в действительности, это может быть и несколько битов), и случайную строку Боба. Она шифрует сообщение некоторым случайным ключом, K, и посылает его обратно Бобу.

$E_K(R,b)$

Эта часть протокола представляет собой процедуру вручения. Боб не может расшифровать сообщение, поэтому он не знает, что за бит прислала Алиса.

Когда для Алисы придет время раскрыть свой бит, протокол продолжается:

(3) Алиса передает Бобу ключ.

(4) Боб расшифровывает сообщения, узнавая бит. Он проверяет свою случайную строку, убеждаясь в правильности бита.

Без случайной строки Боба Алиса может тайно расшифровывать сообщение, посланное Бобу, используя множество ключей, подбирая тот, который позволит при дешифрировании отправленного сообщения изменить врученный бит. Так как у бита только два возможных значения, ее поиски наверняка увенчаются успехом после нескольких попыток. Случайная строка Боба не дает ей использовать этот способ вскрытия, ей придется искать новый ключ, который не только инвертирует врученный бит, но и сохранит нетронутой случайную строку Боба. Если используется достаточно хороший алгоритм шифрования, вероятность удачного поиска чрезвычайно мала. Алиса не может изменить свой бит после его вручения.

Вручение бита с помощью однонаправленных функций

Этот протокол использует однонаправленные функции:

(1) Алиса создает две случайных строки битов, R_1 и R_2.

R_1, R_2

(2) Алиса создает сообщение, состоящее из ее случайных строк и бита, который она хочет вручить (в действительности, это может быть и несколько битов).

(R_1, R_2, b)

(3) Алиса вычисляет однонаправленную функцию для сообщения и посылает результат вместе с одной из случайных строк Бобу.

$H(R_1, R_2, b), R_1$

Это сообщение Алисы является доказательством вручения. Использование однонаправленной функции на этапе (3) мешает Бобу, инвертируя функцию, определить бит.

Когда для Алисы придет время раскрыть свой бит, протокол продолжается:

(4) Алиса отправляет Бобу первоначальное сообщение.

(R_1, R_2, b)

(5) Боб вычисляет однонаправленную функцию для сообщения и сравнивает его и R_1 со значением однонаправленной функции и случайной строкой, полученными на этапе (3). Если они совпадают, то бит правилен.

Преимущество этого протокола перед предыдущим в том, что Бобу не нужно посылать никаких сообщений. Алиса посылает Бобу одно сообщение для вручения бита, а другое - для его раскрытия.

Не нужна и случайная строка Боба, так как результат Алисиного вручения - это сообщение, обработанное однонаправленной функцией. Алиса не может смошенничать и подобрать другое сообщение (R_1, R_2, b'), для которого $H(R_1, R_2, b') = H(R_1, R_2, b)$. Посылая Бобу R_1, она вручает значение b. Если Алиса не сохранит в секрете R_2, то Боб получит возможность вычислить $H(R_1, R_2, b')$ и $H(R_1, R_2, b)$, получая возможность увидеть, что же он получил от Алисы.

Вручение бита с помощью генератора псевдослучайной последовательности

Этот протокол даже проще [1137]:

(1) Боб создает строку случайных битов и посылает ее Алисе.

R_B

(2) Алиса создает стартовую последовательность для генератора псевдослучайных битов. Затем для каждого бита в строке случайных битов Боба она посылает Бобу либо:

(a) выход генератора, если бит Боба равен 0, или

(b) XOR выхода генератора и бита Боба, если Бит Боба равен 1.

Когда для Алисы придет время раскрыть свой бит, протокол продолжается:

(3) Алиса посылает Бобу свою стартовую последовательность.

(4) Боб выполняет этап (2), убеждаясь, что Алиса действует честно.

Если строка случайных битов достаточно длинна, а генератор псевдослучайных битов непредсказуем, мошенничество Алисы практически невозможно.

Blob-объекты

Строки, которые Алиса посылает Бобу для вручения бита, иногда называют **blob-объектами**. Blob-объект - это последовательность битов, хотя протоколы этого и не требуют. Как сказал Жиль Брассар (Gilles Brassard), "Они могли бы быть сделаны и из волшебной пыли, если бы это было полезным " [236]. Blob-объекты обладают следующими четырьмя свойствами:

1. Алиса может вручить blob-объекты. Вручая blob-объект, она вручает бит.
2. Алиса может открыть любой blob-объект, который она вручила. Когда она открывает blob-объект, она может убедить Боба в значении бита, который был вручен вместе с blob-объектом. Следовательно, она не может открыть произвольный blob-объект, например, ноль или единицу.
3. Боб не может знать, каким образом Алиса может открыть blob-объект, который она вручила. Это остается справедливым, даже когда Алиса откроет другие blob-объекты.
4. Blob-объекты не несут никакой информации, кроме вручаемого Алисой бита. Сами по себе blob-объекты, также как и процесс, с помощью которого Алиса вручает и открывает их, не связаны не с чем другим, что Алиса хотела бы сохранить в секрете от Боба.

4.10 Подбрасывание "честной" монеты

Настало время процитировать Джо Килиана (Joe Kilian) [831]:

> Алиса и Боб хотели сыграть в "орла и решку", но монеты у них не было. Алиса предложила простой способ подбрасывать монетку мысленно.
>
> "Сначала вы задумываете случайный бит, затем я задумаю случайный бит. Затем мы выполняем над битами "исключающее или", - предложила она.
>
> "Но если один из нас не будет задумывать биты случайным образом?", - спросил Боб.
>
> "Это не важно. Если хотя бы один из битов действительно случаен, то и "исключающее или" битов должно быть действительно случайным", - ответила Алиса, и после минутного раздумья Боб согласился.
>
> Немного спустя Алиса и Боб наткнулись на книгу по искусственному интеллекту, лежащую на обочине дороги. Алиса, добропорядочная гражданка, сказала: "Один из нас должен подобрать эту книгу и сдать ее в бюро находок". Боб согласился, и предложил использовать их протокол подбрасывания монетки, что бы определить, кто унесет книгу.
>
> "Если полученный бит будет 0, то ты возьмешь книгу, а если 1 - то я", - сказала Алиса. " Какой у тебя бит?"
>
> Боб ответил: "1".
>
> "Ну вот, и у меня такой же", - лукаво заметила Алиса. - "Я думаю, у тебя сегодня неудачный день".
>
> Очевидно, у протокола подбрасывания монетки есть серьезный дефект. Хотя это правда, что "исключающее или" действительно случайного бита, x, и любого независимо распределенного бита, y, дает в результате действительно случайный бит, протокол Алисы не гарантирует, что два бита будут распределены независимо. На самом деле нетрудно убедиться, что не существует мысленного протокола, который позволит двум независимым сторонам подбрасывать "честную" монетку. Алиса и Боб горевали, пока не получили письмо от неизвестного студента с дипломом по криптографии. Информация в письме была слишком теоретической, чтобы ее можно было применить для чего-то земного, но конверт, в котором пришло письмо, оказался чрезвычайно полезным.
>
> Когда Алиса и Боб в следующий раз захотели подбросить монетку, они изменили первоначальный протокол. Сначала бит задумал Боб, но вместо того, чтобы открыть его немедленно, он записывает свой бит на листке бумаги и кладет листок в конверт. Затем Алиса объявляет свой бит. Наконец, Алиса и Боб достают бит Боба из конверта и вычисляют случайный бит. Этот бит уже действительно случаен, независимо от честности играющих. Алиса и Боб получили работающий протокол, с социально значимая мечта криптографов осуществилась, и все они жили долго и счастливо.

Эти конверты выглядят весьма похожими на blob-объекты вручения бита. Когда Мануэль Блам (Manuel Blum) столкнулся с проблемой подбрасывания "честной" монеты по модему [194], он решил ее, используя протокол вручения бита:

(1) Алиса вручает случайный бит, используя любую из схем вручения бита, описанную в разделе 4.9.

(2) Боб загадывает свой бит.

(3) Алиса раскрывает бит Бобу. Боб выигрывает бросок, если он правильно загадал бит.

В общем случае, нам нужен протокол со следующими свойствами:

— Алиса должна "бросить монету" до того, как Боб загадает свой бит.

— Алиса не должна иметь возможности изменить результаты своего броска, узнав бит Боба.

— У Боба не должно быть возможности узнать результат броска перед тем, как он сделает свое предположение.

Существует несколько возможностей выполнить это.

Бросок монеты с помощью однонаправленных функций

Если Алиса и Боб договорятся об однонаправленной функции, протокол прост:

(1) Алиса выбирает случайное число, x. Она вычисляет $y=f(x)$, где $f(x)$ - однонаправленная функция.

(2) Алиса посылает *y* Бобу.

(3) Боб предполагает, что *x* четно или нечетно, и посылает свое предположение Алисе.

(4) Если предположение Боба правильно, результатом броска является "орел", если неправильно - то "решка". Алиса объявляет результат броска монеты и посылает *x* Бобу.

(5) Боб проверяет, что *y=f(x)*.

Безопасность этого протокола обеспечивается однонаправленной функцией . Если Алиса сможет найти *x* и *x′*, такие что *x* - четно, а *x′* - нечетно, и *y=f(x)= f(x′)*, то она каждый раз сможет обманывать Боба. Кроме того, наименьший значащий бит *f(x)* должен быть некоррелирован с *x*. В противном случае Боб сможет обманывать Алису, по крайней мере иногда. Например, если *f(x)* в 75 процентах случаев четна, если *x*, у Боба будет преимущество. (Иногда наименьший значащий бит не является лучшим выбором для использования в приложении, потому что его вычисление может оказаться слишком простым .)

Бросок монеты с помощью криптографии с открытыми ключами

Этот протокол работает как с криптографией с открытыми ключами, так и с симметричной криптографией . Единственное условие - переключение алгоритма. То есть :

$$D_{K_1}(E_{K_2}(E_{K_1}(M))) = E_{K_2}(M)$$

В общем случае это свойство не выполняется для симметричных алгоритмов, но справедливо для некоторых алгоритмов с открытыми ключами (например, RSA с идентичными модулями). Этот протокол:

(1) И Алиса, и Боб создают пары открытый ключ/закрытый ключ.

(2) Алиса создает два сообщения, одно для "орла", а второе - для "решки". Эти сообщения должны включать некоторую случайную строку, чтобы она могла подтвердить их подлинность на последующих этапах протокола. Алиса шифрует оба сообщения своим открытым ключом и посылает их Бобу в произвольном порядке.

$E_A(M_1), E_A(M_2)$

(3) Боб, которые не может прочитать не одно сообщение, случайным образом выбирает одно из них. (Он может посчитать их с помощью "Эники-беники ели вареники", воспользоваться компьютером для взлома протоколы или обратиться к цыганке.) Он шифрует выбранное сообщение своим открытым ключом и посылает его обратно Алисе.

$E_B(E_A(M))$

где M - M_1 или M_2.

(4) Алиса, которая не может прочитать полученное сообщение, расшифровывает его своим закрытым ключом и посылает обратно Бобу.

$D_A(E_B(E_A(M)))= E_B(M_1),$ если $M = M_1$, или $E_B(M_2)$, если $M = M_2$.

(5) Боб расшифровывает сообщение своим закрытым ключом, раскрывая результат броска монеты, и посылает расшифрованное сообщение Алисе.

$D_B(E_B(M_1))$ или $D_B(E_B(M_2))$

(6) Алиса читает результат броска монеты и проверяет, что случайная строка правильна.

(7) Алиса и Боб раскрывают пары своих ключей, чтобы каждый из сторон могла убедиться в отсутствии мошенничества.

Этот протокол самодостаточен. Любая сторона может немедленно обнаружить мошенничество другой, и не требуется третья сторона ни для участия в протоколе, ни в качестве арбитра после завершения протокола . Чтобы посмотреть, как это работает, давайте попытаемся смошенничать .

Если выиграть, смошенничав, хочет Алиса, у нее есть три возможных пути повлиять на результат. Во первых, она может зашифровать два сообщения для "орла" на этапе (2). Боб обнаружит это, когда Алиса раскроет свои ключи на этапе (7). Во вторых, она может использовать какой-то другой ключ для расшифровывания сообщения на этапе (4). Это приведет к бессмыслице, которую Боб и обнаружит на этапе (5). В третьих, она может объявить неправильным сообщение на этапе (6). Боб также обнаружит это на этапе (7), когда Алиса не сможет доказать, неправильность сообщения. Конечно, Алиса может отказаться от участия в протоколе на любом этапе, когда жульничество Алисы станет для Боба очевидным.

Если Боб захочет мошеннически выиграть, его положение ничуть не лучше. Он может неправильно зашифровать сообщение на этапе (3), но Алиса обнаружит обман, взглянув на заключительное сообщение на этапе (6).

Он может заявить, что неправильно выполнил этап (5) из-за какого-то мошенничества со стороны Алисы, но эта форма жульничества вскроется на этапе (7). Наконец, он может послать Алиса сообщение о "решке" на этапе (5), независимо от расшифрованного сообщения, но Алиса сможет немедленно проверить достоверность сообщения на этапе (6).

Бросок монеты в колодец

Интересно отметить, что во всех этих протоколах Алиса и Боб узнают результат броска не одновременно. В каждом протоколе есть момент, когда одна из сторон (Алиса в первых двух протоколах и Боб в последнем) узнает результат броска, но не может изменить его. Эта сторона может, однако, задержать раскрытие результата для второй стороны. Это называется **броском монет в колодец**. Представьте себе высохший колодец. Алиса стоит рядом с колодцем, А Боб - немного подальше. Боб бросает монету, и она падает в колодец. Алиса может теперь заглянуть в колодец и увидеть результат, но она не может спуститься вниз и изменить его. Боб не сможет увидеть результат, пока Алиса не позволит ему подойти и заглянуть в колодец.

Генерация ключей с помощью броска монеты

Реальным применением этого протокола служит генерация сеансового ключа. Протоколы броска монеты позволяют Алисе и Бобу создать случайный сеансовый ключ так, что никто из них не сможет повлиять на то, каким будет этот ключ. Если Алиса и Боб зашифруют свои сообщения, процедура генерации ключа к тому же станет безопасной от злоумышленника.

4.11 Мысленный покер

Протокол, аналогичный протоколу броска монеты с помощью открытых ключей, позволяет Алисе и Бобу играть друг с другом в покер по электронной почте. Алиса вместо создания и шифрования двух сообщений, одного для "орла", а другого - для "решки", создает 52 сообщения M_1, M_2, ..., M_{52}, по числу карт в колоде. Боб случайным образом выбирает пять из них, шифрует своим открытым ключом и посылает обратно Алисе. Алиса расшифровывает сообщения и посылает их обратно Бобу, который расшифровывает их для определения своей "руки". Затем он случайным образом выбирает еще пять сообщений и, не изменяя их, посылает Алисе. Она расшифровывает их, и эти соответствующие карты становятся ее "рукой". В течение игры эта же процедура применяется для сдачи игрокам дополнительных карт. В конце игры Алиса и Боб раскрывают свои карты и пары ключей, чтобы каждый мог убедиться в отсутствии мошенничества.

Мысленный покер с тремя игроками

Покер интереснее, если в игре участвуют несколько человек. Базовый протокол мысленного покера легко может быть распространен на трех и более игроков. В этом случае криптографический алгоритм также должен быть коммутативным.

(1) Алиса, Боб и и Кэрол создают пары открытый ключ/закрытый ключ.

(2) Алиса создает 52 сообщения, по одному для каждой карты колоды. Эти сообщения должны включать некоторую уникальную случайную строку, чтобы Алиса могла проверить их подлинность на последующих этапах протокола. Алиса шифрует все сообщения своим открытым ключом и посылает их Бобу в произвольном порядке.

$E_A(M_n)$

(3) Боб, который не может прочитать не одно сообщение, случайным образом выбирает пять из них. Он шифрует их своим открытым ключом и посылает обратно Алисе.

$E_B(E_A(M_n))$

(4) Боб отправляет Кэрол оставшиеся 47 сообщений.

$E_A(M_n)$

(5) Кэрол, которая не может прочитать не одно сообщение, случайным образом выбирает пять из них. Она шифрует их своим открытым ключом и посылает Алисе.

$E_C(E_A(M_n))$

(6) Алиса, которая не может прочитать ни одно из полученных сообщений, расшифровывает их своим закрытым ключом и посылает обратно Бобу или Кэрол (в соответствии с тем, от кого она их получила).

$D_A(E_B(E_A(M_n)))= E_B(M_n)$

$D_A(E_C(E_A(M_n)))= E_C(M_n)$

(7) Боб и Кэрол расшифровывают сообщения своими ключами, чтобы узнать свои карты

$D_B(E_B(M_n))$

$D_C(E_C(M_n))$

(8) Кэрол случайным образом выбирает пять из оставшихся 42 сообщений и п осылает Алисе.

$E_A(M_n)$

(9) Алиса расшифровывает сообщения, чтобы узнать свои карты.

$D_A(E_A(M_n))$

(10) В конце игры Алиса, Боб и Кэрол раскрывают свои карты и пары ключей, чтобы каждый мог убедиться в отсутствии мошенничества.

Дополнительные карты раздаются подобным же образом. Если карта нужна Бобу или Кэрол, любой из них берет зашифрованную колоду и повторяет протокол с Алисой, Если карта нужна Алисе, то тот, у кого сейчас находится зашифрованная колода, посылает ей случайную карту.

В идеале, этап (10) является обязательным. Свои "руки" в конце протокола должны открывать не все игроки, а только те, которые не спасовали. Так как этап (10) в протокол только для контроля мошенничества, возможны какие-нибудь улучшения.

В покере интересно только, не смошенничал ли победитель. Все остальные могут мошенничать сколько влезет, раз уж они все равно проигрывают. (В действительности это не совсем верно. Кто-то, проигрывая, может собирать данные о стиле игры в покер других игроков.) Итак, взглянем на случаи выигрыша различных игроков.

Если выигрывает Алиса, она раскрывает свою "руку" и свои ключи. Боб может использовать закрытый ключ Алисы для проверки правильности действий Алисы на этапе (2), то есть проверить, что каждое из 52 сообщений соответствует отдельной карте. Кэрол может проверить, что Алиса не лжет о своей "руке", шифруя карты открытым ключом Алисы и проверяя, что они соответствуют шифрованным сообщениям, которые она послала Алисе на этапе (8).

Если выигрывают Боб или Кэрол, победитель раскрывает свои карты и ключи. Алиса может убедиться в правильности карт, проверив свои случайные строки. Она может также убедиться, что сданы были именно эти карты, шифруя их открытым ключом победителя и проверяя, что они совпадают с зашифрованными сообщениями, полученными на этапах (3) или (5).

Этот протокол не защищен от сговора игроков-мошенников. Алиса и другой игрок могут объединиться и безнаказанно вместе надувать третьего игрока. Следовательно, важно проверять все ключи и случайные строки каждый раз, когда игроки раскрывают свои карты. И если вы сидите за виртуальным столом с двумя игроками, которые никогда одновременно не раскрывают свои карты, причем один из них сдает (в предыдущем протоколе это Алиса) кончайте игру.

Все это красиво в теории, но реализовать все это на компьютере весьма непросто. В реализации для трех игроков на трех различных Sparc-станциях восемь часов требуется только для тасования колоды, так что лучше поиграть в настоящий покер [513].

Вскрытия мысленного покера

Криптографы показали, что при использовании этими протоколами покера алгоритма с открытыми ключами RSA происходит небольшая утечка информации [453, 573]. Конкретно, если двоичное представление карт является квадратичным остатком (см раздел 11.3), то зашифрованные карты также являются квадратичным остатком. Это свойство может быть использовано для "крапления" некоторых карт - например, всех тузов. Это даст не много информации о сдачах, но в такой игре как покер даже чуть-чуть информации даст преимущество при длительной игре.

Шафи Голдвассер (Shafi Goldwasser) и Сильвия Микали (Silvia Micali) [624] разработали протокол умственного покера для двух игроков, который решает эту проблему, хотя из-за своей сложности он скорее имеет только теоретическое значение. Обобщенный протокол покера для n игроков, устраняющий проблему утечки информации, был разработан в [389].

Результаты других исследований протоколов игры в покер можно найти в [573, 1634, 389]. Усложненный протокол, позволяющий игрокам не раскрывать своих "рук", приведен в [390]. Дон Копперсмит (Don Coppersmith) рассматривает два способа мошенничества в умственном покере, использующем алгоритм RSA [370].

Анонимное распределение ключей

Хотя непохоже, чтобы кто-нибудь собирался использовать этот протокол для игры в покер по модему, Чарльз Пфлегер (Charles Pfleeger) рассматривает ситуацию, в которой этот тип протокола может оказаться полезным [1244].

Рассмотрим проблему распределения ключей. Если предположить, что люди не могут сами генерировать свои ключи(ключи должны иметь определенную форму, или должны быть подписаны некоторой организацией, или еще что-нибудь подобное), то для генерации и рассылки ключей придется создать Центр распределения ключей (Key Distribution Center, KDC). Проблема в том, что нужно найти такой способ распределения ключей, что никто, включая сервер, не сможет понять, кому какой ключ достался. Следующий протокол решает эту проблему:

(1) Алиса создает пару открытый ключ/закрытый ключ. В этом протоколе она сохраняет в секрете оба ключа.

(2) KDC генерирует непрерывный поток ключей.

(3) KDC шифрует ключи, один за другим, своим открытым ключом.

(4) KDC передает зашифрованные ключи, один за другим, по сети.

(5) Алиса случайным образом выбирает ключ.

(6) Алиса шифрует выбранный ключ своим открытым ключом.

(7) Алиса ждет какое-то время (достаточно большое, чтобы сервер не мог определить, какой ключ она выбрала) и посылает дважды зашифрованный ключ в KDC.

(8) KDC расшифровывает дважды зашифрованный ключ с помощью своего закрытого ключа, получая ключ, зашифрованный открытым ключом Алисы.

(9) Сервер посылает шифрованный ключ обратно Алисе.

(10) Алиса расшифровывает ключ с помощью своего закрытого ключа.

У находящейся где-то в середине протокола Евы нет ни малейшего представления о выбранном Алисой ключе. Она видит непрерывный поток ключей, создаваемых на этапе (4). Когда Алиса посылает ключ серверу на этапе (7), он шифруется ее открытым ключом, который также для этого протокола хранится в секрете. Способа связать это сообщение с потоком ключей у Евы нет. Когда ключ возвращается Алисе сервером на этапе (9), он также зашифрован открытым ключом Алисы. Ключ становится известным, только когда Алиса расшифровывает его на этапе (10).

Если вы используете RSA, в этом протоколе происходит утечка информации со скоростью, по меньшей мере, один бит на сообщение. Причиной этого снова являются квадратичные остатки. Если вы собираетесь использовать этот способ для распределения ключей, убедитесь, что эта утечка не приведет к каким-либо последствиям. Кроме того, поток ключей, создаваемый KDC должен быть достаточно большим, чтобы противостоять вскрытию грубым взломом. Конечно же, если Алиса не может верить KDC, то она не должна пользоваться его ключами. Мошенничающий KDC может предусмотрительно записывать все создаваемые им ключи. Тогда он сможет найти среди них ключ, выбранный Алисой.

Этот протокол также предполагает, что Алиса будет действовать честно. При использовании RSA существует ряд действий, которые может предпринять Алиса, чтобы получить больше информации, чем ей удалось бы при другом методе шифрования. В нашем сценарии эта проблема не существенна, но при других обстоятельствах она может стать важной.

4.12 Однонаправленные сумматоры

Алиса является членом организации "Заговорщики". Иногда ей приходится встречаться с другими членами в плохо освещенных ресторанах и шептать секреты налево и направо. Беда в том, что рестораны настолько плохо освещены, что она не может быть уверена, что человек, сидящий напротив нее за столом, тоже член организации.

"Заговорщики" могут выбирать из нескольких решений. Каждый может носить с собой список членов организации. Это влечет за собой две следующих проблемы. Во первых, теперь каждый должен носить с собой большую базу данных, и, во вторых, им придется как следует охранять этот список членов. Другим способом является использование идентификационных карт, выпущенных надежным секретарем. Дополнительным преимуществом этого способа является то, что и посторонние смогут проверять членов организации (всякие скидки в местной бакалейной лавке), до для этого нужен надежный секретарь. Никому из "заговорщиков" нельзя доверять до такой степени.

Новым решением является использование **однонаправленного сумматора** [116]. Это что-то похожее на од-

нонаправленные хэш-функции, для которых выполняется требование коммутативности. То есть, можно хэш ировать базы данных членов организации в произвольном порядке и получать одно и то же значение . Более того, можно добавлять новых членов в хэш-таблицу и получать новое хэш-значение, снова не зависящее от порядка .

Итак, вот что делает Алиса. Она выполняет расчет, используя множество всех имен членов организации, о тличных от нее. Затем она сохраняет это полученное значение вместе со своим именем. Боб и другие члены д елают то же самое. Теперь, когда Алиса и Боб встречаются в плохо освещенном ресторане, они просто обмен иваются друг с другом вычисленными значениями и именами. Алиса убеждается, что результат, получаемый при добавлении имени Боба к значению Алисы, совпадает с результатом, получаемым при добавлении имени Ал исы к значению равно значению Боба. Боб делает то же самое. Теперь они оба знают, что собеседник - также член организации. И в то же время никто не сможет определить личности других членов организации .

Более того, рассчитанные значения каждого члена могут быть выданы посторонним. Тогда Алиса сможет подтвердить свое членство постороннему (возможно, для членской скидки в буфете местной контрразведки), не показывая ему весь список членов.

Новых членов можно добавить просто послав по кругу новые имена. К несчастью, удалить члена можно только единственным путем: всем членам рассылается новый список и они пересчитывают свои значения . Но "заговорщикам" придется выполнять это действие только при отставке кого-то из членов, мертвые члены могут остаться в списке. (Странно, но это не создает проблемы.)

Это разумная идея применяется в ряде приложений, когда вы хотите достичь эффекта цифровой подписи без использования централизованной системы подписей.

4.13 Раскрытие секретов "все или ничего"

Представьте себе, что Алиса - бывший агент бывшего Советского Союза, а теперь безработная . Чтобы заработать, она продает секреты. Каждый, готовый заплатить названную цену, может купить секрет . У нее даже есть каталог. Все ее секреты с аппетитными названиями упорядочены по номерам: "Где Джимми Хоффа?", "Кто тайно контролирует Трехстороннюю комиссию?", "Почему Борис Ельцин всегда выглядит, как будто он прогл отил живую лягушку?", и т.д.

Алиса не хочет отдавать два секрета по цене одного и не показывает даже части информации, касающейся любого из секретов. Боб, потенциальный покупатель, не хочет платить за кота в мешке . Он также не хочет сообщать Алисе, какие из секретов ему нужны. Это не ее дело, и, кроме того, тогда Алиса сможет добавить в свой каталог пункт "Секреты, которыми интересуется Боб".

Протокол покера не работает в этом случае, так как в конце этого протокола Алиса и Боб должны раскрыть свои карты друг другу. К тому же, существуют трюки, с помощью которых Боб может узнать сразу несколько секретов.

Решение называется **раскрытием секретов "все или ничего"** (all-or-nothing disclosure of secrets, **ANDOS**) [246], потому что если Боб получил любую информацию о любом из секретов Алисы, то он потерял возможность узнать что-либо еще о других ее секретах.

В криптографической литературе можно найти различные протоколы ANDOS. Некоторые из них обсуждаются в разделе 23.9.

4.14 Условное вручение ключей

Вот отрывок из введения в тему Сильвио Микали [1084]:

> Сегодня подслушивание с разрешения суда является эффективным методом отдавать преступников в руки правосудия . По нашему мнению еще более важно, что это также предотвращает дальнейшее распространение преступления, удерживая от использования обычных сетей связи с незаконными целями. Следовательно, существует обоснованное беспокойство, что ра спространение криптографии с открытыми ключами может быть на руку преступным и террористическим организациям . Действительно, во многих законах предполагается, что соответствующие правительственные ведомства при определенных усл овиях, оговоренных законом, должны иметь возможность получить открытый текст любого обмена информацией по общедо ступным сетям. В настоящее время many это может быть трактоваться, как требование к законопослушным гражданам либо (1) *использовать слабые криптосистемы* - т.е., криптосистемы, которые соответствующие власти (а также кто угодно!) см огут вскрыть с помощью умеренных усилий, или (2) заранее *сообщать свои секреты* властям. Не удивительно, что такая альтернатива законно встревожила многих заинтересованных граждан, создавая в результате мнение, что тайна личности дол жна стоять над национальной безопасностью и отправлением закона .

Условное вручение ключей является сутью продвигаемых правительством США программы Clipper и Стандарта условного шифрования (Escrowed Encryption Standard). Проблема в том, чтобы и обеспечить тайну личн ости, и в то же время позволить разрешенное судом подслушивание .

Escrowed Encryption Standard обеспечивает безопасность с помощью защищенного оборудования. У каждой микросхемы шифрования уникальный идентификационный номер (ID) и секретный ключ. Этот ключ делится на

две части и хранится, вместе с ID, двумя различными организациями условного вручения. Всякий раз, когда микросхема шифрует файл данных, она сначала шифрует сеансовый ключ уникальным секретным ключом. Затем она передает зашифрованный сеансовый ключ и свой ID по каналу связи. Когда правоохранительные органы хотят расшифровать поток информации, зашифрованной одной из этих микросхем, они извлекают из потока ID, получают соответствующие ключи из организаций условного вручения, объединяют их с помощью операции XOR, расшифровывают сеансовый ключ и затем используют его для дешифрирования потока сообщений. Для защиты от мошенников в эту схему введены дополнительные усложнения, подробно описанные в разделе 24.16. Аналогичная схема может быть реализована и программно с использованием криптографии с открытыми ключами [77, 1579, 1580, 1581].

Микали называет свою идею **честной криптосистемой** [1084,10851. (Говорят, что правительство США заплатило Микали $1000000 за использование его патентов [1086, 1087] в своем стандарте Escrowed Encryption Standard, затем патент Микали купил Банковский трест.) В таких криптосистемах закрытый ключ делится на части и распределяется среди различных организаций. Как и схема с совместным использованием секрета, эти организации могут объединиться и восстановить закрытый ключ. Однако, части ключа обладают дополнительным свойством - их правильность может быть проверена независимо без восстановления закрытого ключа.

Алиса может создать свой собственный закрытый ключ и распределить его части среди n доверительных собственников. Ни один из них не может восстановить закрытый ключ Алисы. Однако каждый может проверить, что его часть - это правильная часть закрытого ключа. Алиса не может послать кому-то из доверительных собственников строку случайных битов и надеяться улизнуть. Если судебные власти разрешат подслушивание, соответствующие правоохранительные органы смогут воспользоваться постановлением суда для того, чтобы n доверительных собственников выдали свои части. Собрав все n частей, власти восстановят закрытый ключ и смогут подслушивать линии связи Алисы. С другой стороны, чтобы получить возможность восстановить ключ Алисы и нарушить ее тайну личности, Мэллори придется купить всех n доверительных собственников.

Вот как работает этот протокол:

(1) Алиса создает пару закрытый ключ/открытый ключ. Она разбивает закрытый ключ на несколько открытых и закрытых частей.

(2) Алиса посылает открытую часть и соответствующую закрытую часть каждому из доверительных собственников. Эти сообщения должны быть зашифрованы. Она также посылает открытый ключ в KDC.

(3) Каждый из доверительных собственников независимо выполняет вычисления над своими закрытой и открытой частями, чтобы убедиться в их правильности. Каждый доверительный собственник хранит закрытую часть в каком-нибудь надежном месте и отправляет открытую часть в KDC.

(4) KDC выполняет иное вычисление для открытых частей и открытого ключа. Убедившись, что все правильно, он подписывает публичный ключ и отправляет его обратно Алисе или помещает в какую-нибудь базу данных.

При наличии постановления суда о подслушивании каждый из доверительных собственников передает свою часть в KDC, и KDC получает возможность восстановить закрытый ключ. До этой передачи ни KDC, ни кто-либо из доверительных собственников не может самостоятельно восстановить закрытый ключ, для восстановления ключа нужны все доверительные собственники.

Любой алгоритм с открытыми ключами можно сделать "честным" подобным образом. Ряд конкретных алгоритмов рассматривается в разделе 23.10. В работах Микали [1084, 1085] обсуждаются пути объединения описанного с пороговой схемой, чтобы для восстановления закрытого ключа требовалось некоторое подмножество доверительных собственников (например, трое из пяти). Он также показывает, как объединить это с рассеянной передачей (см. раздел 5.5) так, чтобы доверительные собственники не знали, чей закрытый ключ восстанавливается.

"Честные" криптосистемы несовершенны. Преступник может использовать такую систему, применяя подсознательный канал (см. раздел 4.2.), чтобы вставить другой секретный ключ в свою информацию. Таким образом он может безопасно обмениваться информацией с кем-нибудь еще, используя подсознательный ключ и совершенно не волнуясь по поводу разрешенного судом подслушивания. Данная проблема решается другим протоколом, который называется **отказоустойчивым условным вручением ключей** [946, 833]. Этот алгоритм и протокол описывается в разделе 23.10.

Политика условного вручения ключей

Помимо правительственных планов относительно условного вручения ключей распространяются и коммерческие системы с условным вручением ключей. Возникает очевидный вопрос: какое преимущество от условного вручения ключей получает пользователь?

Ну, на самом деле никакого. Пользователь не получает от условного вручения ключей ничего такого, чего он

и сам не смог бы обеспечить. Он и сам может создать резервную копию ключей, если захочет (см. раздел 8.8). Условное вручение ключей гарантирует, что полиция сможет подслушивать его разговоры или читать файлы данных, даже когда они шифрованы. Оно гарантирует, что NSA сможет подслушивать его международные звонки - без всякого ордера - хотя они и шифрованы. Может ему будет разрешено использовать такую криптографию с теми странами, для которых сейчас установлены запреты, но это сомнительное преимущество.

Недостатки условного вручения ключей весьма ощутимы. Пользователю приходится верить в безопасность действия организаций, занятых условным вручением ключей также, как и в честность занятых этим людей. Ему придется верить, что политика соответствующих организаций останется неизменной, правительство не поменяет законы, и те, кто имеет полномочия вскрыть его ключ, будут делать это по закону и с полной ответственностью. Вообразите нападение террористов на Нью-Йорк, какие бы ограничения не были бы сметены полицией, чтобы остановить последствия?

Трудно представить себе, что эти условные схемы шифрования, как говорят их защитники, будут использоваться без принуждения извне. Следующим очевидным шагом будет запрет на использование всех других способов шифрования. Это, вероятно, единственный способ добиться коммерческого успеха этой системы, и это, определенно, единственный способ заставить технически грамотных преступников и террористов использовать ее. Пока не ясно, насколько трудно будет объявить не-условную криптографию вне закона, или как это повлияет на криптографию как на академическую дисциплину. Как я могу исследовать программно ориентированные алгоритмы криптографии, не имея доступа к программному обеспечению устройств не-условного шифрования, нужна ли мне будет специальная лицензия?

И другие законные вопросы. Как условно врученные ключи повлияют на ответственность пользователей, должна ли становиться известной зашифрованная информация? Если правительство США пытается защитить органы условного вручения, не будет ли это косвенным свидетельством того, что если секрет скомпрометирован либо пользователем, либо органами условного вручения, то виновником будет признан пользователь?

Что если база данных главной службы условного вручения ключей, все равно государственной или коммерческой, будет украдена? Что, если правительство США попытается ненадолго скрыть этот факт? Ясно, что все эти вопросы повлияют на желание пользователей пользоваться условным вручением ключей. Если использование не будет добровольным, то пара скандалов вызовет рост политического давления с целью либо сделать использование подобных систем добровольным, либо ввести новые сложные правила в этой отрасли.

Еще более опасным будет скандал, когда выяснится, что годами под наблюдением находился политический оппонент текущей администрации или некий громкоголосый критик спецслужб и полицейских ведомств. Это сильно настроит общественное мнение против условного шифрования.

Если ключи подписей будут шифроваться тем же способом, что и ключи шифрования, возникнут дополнительные моменты. Допустимо ли для властей использовать ключи подписей для проведения операции против подозреваемого преступника? Будет ли признана судом подлинность подписей, основанных на ключах с условным вручением? Чем в действительности будут владеть пользователи, если власти действительно используют их ключи пользователей для подписи какого-то невыгодного контракта, для поддержки определенных отраслей промышленности, или просто, чтобы украсть деньги?

Глобальное распространение криптографии рождает дополнительные вопросы. Будут ли схемы условного вручения ключей совместимы в различных странах? Захотят ли транснациональные корпорации смириться с существованием в каждой стране своих условно врученных ключей, совместимых с различным местным законодательством? Без обеспечения совместимости исчезает одно из пропагандируемых преимуществ схемы с условным вручением ключей (международное использование мощных средств криптографии).

Что если ряд стран не примет на веру надежность организаций, связанных с условным вручением ключей? Как будут пользователи вести свои дела в этих странах? Будут ли признаны судами их электронные контракты, или тот факт, что ключи их подписей условно хранятся в США, позволит им утверждать где-нибудь в Швейцарии, что этот электронный контракт мог подписать кто-то другой? Или для людей, которые ведут дела в подобных странах, будут специальные исключения?

А что делать с промышленным шпионажем? Где гарантии, что страны, занимающиеся сейчас промышленным шпионажем для своих важнейших или государственных предприятия, не воспользуются для этого системами с условным вручением ключей? В самом деле, так как ни одна страна не собирается позволять другим странам следить за своими разведывательными операциями, распространение условного шифрования возможно приведет к увеличению подслушивания.

Даже если страны, в которых соблюдаются гражданские права, будут использовать условность такого шифрования только для законного преследования преступников и террористов, где-нибудь этим обязательно воспользуются для выслеживания диссидентов, шантажа политических оппонентов, и т.п. Цифровые линии связи предоставляют возможность гораздо более тщательно, чем это было возможно в аналоговом мире, контролировать действия граждан, их мнения, Digital communications offer the opportunity to do a much more thorough lob of

monitoring citizens' actions, opinions, доходы и объединения.

Не ясно, не будет ли через 20 лет продажа системы с условным вручением ключей Турции или Китаю пох одить на продажу электрических дубинок Южной Африке в 1970 году или на строительство химического завода в Ираке в 1980 году. Даже хуже, легкое и незаметное подслушивание линий связи может искусить многие пр авительства, которые раньше, возможно, этим и не занимались, следить за корреспонденцией своих граждан . И нет гарантии, что либеральные демократии устоят перед подобным искушением .

Глава 5
Развитые протоколы

5.1 Доказательства с нулевым знанием

А вот другая история:

Алиса: "Я знаю пароль компьютера Федеральной Резервной Системы, компоненты секретного соуса МакДональдс и оде р-жание 4-го тома Дональда Кнута".

Боб: "Нет, ты не знаешь".

Алиса: "Нет, я знаю".

Боб: "Не знаешь".

Алиса: "Нет, знаю".

Боб: "Докажи".

Алиса: "Хорошо, я скажу тебе". Она шепчет Бобу на ухо.

Боб: "Это интересно. Теперь я тоже это знаю и собираюсь рассказать это все *Вашингтон Пост*".

Алиса: "Ооооой".

К несчастью, обычно Алиса может доказать что-нибудь Бобу, только рассказав ему все. Но тогда он тоже получит все сведения. Затем Боб может выложить полученные сведения кому угодно, и Алиса ничего не сможет с этим поделать. (В литературе для описания этих протоколов часто используются различные персонажи . Пегги обычно доказывает, а Виктор проверяет. Именно эти имена появляются в используемых примерах вместо Ал и-сы и Боба.)

Используя однонаправленные функции, Пегги сможет провести **доказательство с нулевым знанием** [626]. Этот протокол доказывает Виктору, что у Пегги действительно есть информация, но не дает Виктору не мале й-шей возможности узнать, что это за информация.

Эти доказательства принимают форму интерактивного протокола. Виктор задает Пегги ряд вопросов. Если Пегги знает секрет, то она ответит на все вопросы правильно. Если секрет ей неизвестен, у нее есть некоторая вероятность - 50 процентов в следующих примерах - ответить правильно . После примерно 10 вопросов Виктор убедится, что Пегги знает секрет. Но ни один из вопросов или ответов не даст Виктору ни малейших сведений об информации Пегги, но докажет знание Пегги этой информации .

Базовый протокол с нулевым знанием

Жан-Жак Кискатер (Jean-Jacques Quisquater) и Луи Гилу (Louis Guillou) поясняют нулевое знание историей о пещере [1281]. У пещеры, показанной на 4-й, есть секрет. Тот, кто знает волшебные слова может открыть п о-тайную дверь между C и D. Для всех остальных оба прохода ведут в тупик.

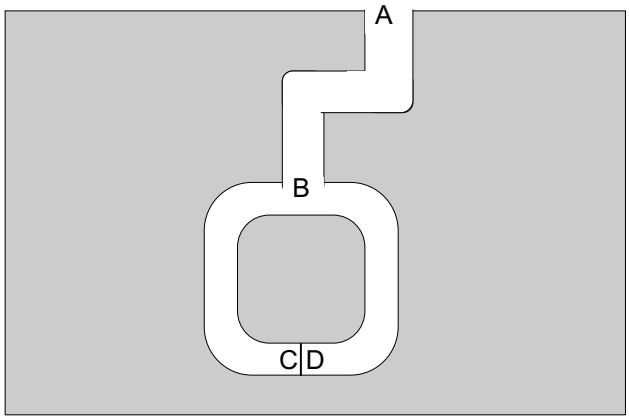

Рис. 5-1. Пещера нулевого знания

Пегги знает секрет пещеры. Она хочет доказать свое знание Виктору, но не хочет раскрывать волшебных слов. Вот как она убеждает его:

(1) Виктор находится в точке A.

(2) Пегги проходит весь путь по пещере, либо до точки C, либо до точки D.

(3) После того, как Пегги исчезнет в пещере, Виктор переходит в точку B.

(4) Виктор кричит Пегги, спрашивая ее либо о:

(a) или выйти из левого прохода

(b) выйти из правого прохода.

(5) Пегги исполняет его просьбу, при необходимости используя волшебные слова, чтобы отпереть дверь.

(6) Пегги и Виктор повторяют этапы (1) - (5) *n* раз.

Предположим, что у Виктора есть видеокамера, и он записывает все, что видит. Он записывает, как Пегги исчезает в пещере, записывает, как он сам кричит, указывая, где Пегги должна появиться, записывает как Пегги появляется. Он записывает все *n* тестов. Если он покажет эту видеозапись Кэрол, поверит ли она, что Пегги знает волшебные слова, отпирающие дверь? Нет. А что если Пегги и Виктор заранее договорились, что Виктор будет кричать, а Пегги будет делать вид, что она прошла весь путь. Тогда она будет каждый раз выходить из указанного Виктором места, не зная волшебных слов. Или они могли сделать по другому. Пегги входит в один из проходов и Виктор случайным образом выкрикивает свои просьбы. Если Виктор угадывает правильно, хорошо, если нет - они вырежут эту попытку из видеозаписи. В любом случае Виктор может получить видеозапись, показывающую в точности ту последовательность, которая получилась бы, если бы Пегги знала волшебные слова.

Этот опыт показывает две вещи. Во первых, Виктор не может убедить третью сторону в правильности доказательства. И во вторых, данный протокол является протоколом с нулевым знанием. Если Пегги не знает волшебных слов, то очевидно, что Виктор не узнает ничего из просмотра видеозаписи. Но так как нет способа отличить реальную видеозапись от подделанной, то Виктор не может ничего узнать из реального доказательства - это и есть нулевое знание.

Методика, используемая в этом протоколе, называется **разрезать и выбрать** из-за того, что она похож на классический протокол честного деления чего-либо:

(1) Алиса делит некую вещь пополам.

(2) Боб выбирает одну из половин себе.

(3) Алиса забирает оставшуюся половину.

В интересах Алисы честно разделить на этапе (1), потому что Боб выберет на этапе (2) ту половину, которая ему больше нравится. Майкл Рабин (Michael Rabin) первым использовал в криптографии технику "разрезать и выбрать" [1282]. Понятия **интерактивного протокола** и нулевого знания были формализованы позже [626, 627].

Протокол "разрезать и выбрать" работает, потому что Пегги не может несколько раз подряд угадывать, откуда Виктор попросит ее выйти. Если Пегги не знает секрета, он может выйти только из того прохода, в который она зашла. В каждом раундепротокола ее вероятность (иногда называемая **аккредитацией**) угадать, с какой стороны Виктор попросит ее выйти, составляет 50 процентов, поэтому ее вероятность обмануть Виктора также равна 50 процентам. Вероятность обмануть его в двух раундах составит 25 процентов, а во всех *n* раундах - один шанс из 2^n. После 16 раундов у Пегги 1 шанс из 65536 обмануть Виктора. Виктор может уверенно предположить, что если все 16 доказательств Пегги правильны, то она действительно знает тайные слова, открывающие дверь между точками C и D. (Аналогия с пещерой несовершенна. Пегги может просто входить с одной стороны и выходить с другой, протокол "разрезать и выбрать" не нужен. Однако, он необходим с для нулевого знания с математической точки зрения.)

Предположим, что Пегги известна некоторая информация, которая является решением трудной проблемы. Базовый протокол нулевого знания состоит из нескольких раундов.

(1) Пегги использует свою информацию и случайное число для преобразования одной трудной проблемы в другую, изоморфную оригинальной проблеме. Затем она использует свою информацию и случайное число для решения новой трудной проблемы.

(2) Пегги вручает решение новой проблемы, используя схему вручения бита.

(3) Пегги раскрывает Виктору новый экземпляр проблемы. Виктор не может использовать эту новую проблему для получения информации о первоначальной проблеме или ее решении.

(4) Виктор просит Пегги либо

(a) доказать ему, что новая и старая проблема изоморфны (т.е., два различных решения для двух связанных проблем), либо

(b) открыть решение, полученное на этапе (2) и доказать, что это решение новой проблемы.

(5) Пегги исполняет его просьбу.

(6) Пегги и Виктор повторяют этапы (1) - (5) n раз.

Помните видеокамеру в протоколе для пещеры? Здесь вы можете сделать то же самое . Виктор может записать обмен между ним и Пегги. Он не сможет использовать эту запись для убеждения Кэрол, но он всегда может сговориться с Пегги с целью создать имитатор, который подделывает информацию Пегги . Этот аргумент может быть использован, чтобы доказать, что используется доказательство с нулевым знанием .

Математическая основа доказательства этого типа сложна. Проблемы и случайное преобразование должны выбираться осторожно, чтобы Виктор не получил никакой информации о решении оригинальной проблемы, даже после многих повторений протокола . Не все трудные проблемы можно использовать для доказательств с нулевым знанием, но большинство из них .

Изоморфизм графа

Объяснение этого понятия, пришедшего из теории графов [619, 622], может занять некоторое время. Граф представляет собой сеть линий ,соединяющих различные точки. Если два графа идентичны во всем, кроме имен точек, они называются **изоморфными**. Для очень больших графов доказательство их изоморфности может потребовать веков компьютерного времени , это одна из так называемых **NP-полных** проблем, рассматриваемых в разделе 11.1.

Предположим, что Пегги знает об изоморфности двух графов , G_1 и G_2. Следующий протокол докажет Виктору знание Пегги:

(1) Пегги случайным образом тасует G_1, получая другой граф, H, который изоморфен G_1. Так как Пегги знает об изоморфизме H и G_1, то ей также известен изоморфизм между H и G_2. Для любого другого поиск изоморфизма между H и G_1 или H и G_2 является такой же трудной задачей, как и поиск изоморфизма между G_1 и G_2.

(2) Пегги посылает H Виктору.

(3) Виктор просит Пегги либо

(a) доказать, что H и G_1 изоморфны, либо

(b) доказать, что H и G_2 изоморфны.

(4) Пегги исполняет его просьбу. Она либо:

(a) доказывает, что H и G_1 изоморфны, не доказывая, что H и G_2 изоморфны, либо

(b) доказывает, что H и G_2 изоморфны, не доказывая, что H и G_1 изоморфны.

(5) Пегги и Виктор повторяют этапы (1) - (4) n раз.

Если Пегги не знает об изоморфизме между G_1 и G_2, она не сможет создать граф H, изоморфный обоим графам. Она может создать либо граф, который изоморфен G_1, либо граф, который изоморфен G_2. Как и в предыдущем примере у нее только 50 шансов из 100 угадать, какое доказательство потребует от нее Виктор на этапе (3).

Этот протокол не дает Виктору никакой полезной информации, помогающей ему из ответов Пегги установить изоморфизм между G_1 и G_2. Так как Пегги для каждого нового раунда протокола генерирует новый граф H, Виктор не сможет получить информацию независимо от того, из скольких раундов будет состоять их протокол . Он не сможет из ответов Пегги установить изоморфизм между G_1 и G_2.

В каждом раунде Виктор получает новое случайное преобразование H, вместе с изоморфизмом между H и G_1 или G_2. Виктор может также создать что-то подобное самостоятельно. Так как Виктор может создать имитацию протокола, это действительно доказательство с нулевым знанием .

Гамильтоновы циклы

Вариант этого примера был впервые представлен Мануэлем Блюмом (Manuel Blum) [196]. Пегги знает кружной, продолжительный путь вдоль линий графа, который проходит через каждую точку только один раз . Этот путь называется **гамильтоновым циклом**. Поиск гамильтонова цикла является другой тяжелой задачей . У Пегги есть эта информация - она, возможно, получила ее создав граф с конкретным гамильтоновым циклом - и она хочет доказать Виктору, что эта информация ей известна .

Пегги знает гамильтонов цикл графа , G. Виктору известен G, но не его гамильтонов цикл . Пегги хочет доказать Виктору, что она знает гамильтонов цикл, не раскрывая самого цикла . Вот как она должна действовать:

(1) Пегги случайным образом преобразовывает G. Она передвигает точки и изменяет их метки, создавая но-

вый граф, *H*. Поскольку *G* и *H* топологически изоморфны (т.е., это один и тот же граф), если ей известен гамильтонов цикл *G*, то она легко может найти гамильтонов цикл *H*. Если она не сама создает *H*, определение изоморфизма между двумя графами будет являться другой сложной проблемой, решение которой также потребует веков компьютерного времени. Затем она шифрует *H*, получая *H'*. (Должно использоваться вероятностное шифрование каждой строчки *H*, то есть, шифрованный 0 или шифрованная 1 для каждой линии *H*.)

(2) Пегги передает Виктору копию *H*.

(3) Виктор просит Пегги либо:

(a) доказать ему, что *H'* - это зашифрованная изоморфная копия *G*, либо

(b) показать ему гамильтонов цикл для *H*.

(4) Пегги исполняет его просьбу. Она либо:

(a) доказывает, что *H'* - это зашифрованная изоморфная копия *G*, раскрывая преобразования и расшифровывая все, не показывая гамильтонов цикл для *G* или *H*, либо

(b) показывает гамильтонов цикл для *H*, расшифровывая только те строки, которые образуют гамильтонов цикл, не доказывая, что *H* и *G* топологически изоморфны.

(5) Пегги и Виктор повторяют этапы (1) - (4) *n* раз.

Если Пегги не обманывает, она сможет предъявить Виктору одно из доказательств на этапе (3). Однако, если гамильтонов цикл для *G* ей неизвестен, она не сможет создать зашифрованный граф *H'*, который удовлетворяет обоим доказательствам. Лучшее, что она может сделать - это создать или граф, изоморфный *G*, или граф с таким же числом точек и линий и правильным гамильтоновым циклом. Хотя ее шансы угадать, какое доказательство потребует Виктор на этапе (3), составляют 50 процентов, Виктор может повторить протокол достаточное число раз, убеждаясь, что Пегги знает гамильтонов цикл для *G*.

Параллельные доказательства с нулевым знанием

В базовом протоколе с нулевым знанием используется *n* обменов информацией между Пегги и Виктором. Почему бы не выполнить их параллельно:

(1) Пегги использует свою информацию и *n* случайных чисел для преобразования трудной проблемы в *n* различных изоморфных проблем. Затем она с помощью своей информации и случайных чисел решает *n* новых трудных проблем.

(2) Пегги вручает решение *n* новых трудных проблем.

(3) Пегги раскрывает Виктору эти *n* новых трудных проблем. Виктор не может воспользоваться этими новыми проблемами для получения информации об оригинальных проблемах или их решении.

(4) Для каждой новой трудной проблемы Виктор просит Пегги либо

(a) доказать ему, что старая и новая проблемы изоморфны, либо

(b) раскрыть решение, врученное на этапе (2), и доказать, что оно является решением данной новой проблемы.

(5) Пегги исполняет его просьбу для каждой новой проблемы.

К несчастью, все не так просто. Этот протокол, в отличие от предыдущего, не обладает такими же свойствами нулевого знания. На этапе (4) Виктор может потребовать, чтобы доказательство было представлено в виде значения однонаправленной хэш-функции всех значений, врученных на первом этапе, делая невозможным имитацию записи протокола. Это тоже нулевое знание, но другого рода. На практике оно представляется безопасным, но никто не знает, как это доказать. Мы действительно знаем только то, что при определенных условиях определенные протоколы для определенных проблем могут быть выполнены параллельно без потери свойства нулевого знания [247, 106, 546, 616].

Неинтерактивные доказательства с нулевым знанием

Кэрол невозможно убедить, потому что она не участвует в интерактивном процессе протокол. Для убеждения Кэрол и других заинтересованных лиц нам нужен неинтерактивный протокол.

Для неинтерактивных доказательств с нулевым знанием был придуман ряд протоколов [477, 198, 478, 197], которые не требуют непосредственного взаимодействия. Пегги может опубликовать их и, таким образом, доказать свое знание всем, у кого найдется время это проверить

Базовый протокол похож на параллельное доказательство с нулевым знанием, но место Виктора занимает

однонаправленная хэш-функция:

(1) Пегги использует свою информацию и n случайных чисел для преобразования трудной проблемы в n различных изоморфных проблем. Затем она с помощью своей информации и случайных чисел решает n новых трудных проблем.

(2) Пегги вручает решение n новых трудных проблем.

(3) Пегги использует все эти вручения в качестве входа для однонаправленной хэш-функции. (В конце концов эти вручения - не что иное, как строки битов.) Затем она сохраняет первые n битов полученного значения однонаправленной хэш-функции.

(4) Пегги берет n битов, полученных на этапе (3). По очереди для каждой n-ой трудной проблемы она берет n-ый бит и

(a) если бит равен 0, доказывает, что старая и новая проблемы изоморфны, либо

(b) если бит равен 1, раскрывает решение, врученное на этапе (2), и доказывает, что оно является решением данной новой проблемы.

(5) Пегги опубликовывает все решения, врученные на этапе (2), и все доказательства, полученные на этапе (4).

(6) Виктор, Кэрол и все остальные заинтересованные лица проверяют, что этапы (1)-(5) выполнены правильно.

Это впечатляет: Пегги может опубликовать некоторые данные, которые не содержат никакой информации о ее секрете, но могут кого угодно убедить в существовании самого секрета. Этот протокол может быть использован проверка определена как вычисление однонаправленной хэш-функции первоначальных сообщений и подписываемого сообщения.

Эта схема работает, потому что однонаправленная хэш-функция действует как беспристрастный генератор случайных битов. Чтобы мошенничать, Пегги нужно уметь предсказывать результат однонаправленной хэш-функции. (Помните, если решение трудной проблемы ей неизвестно, она может сделать на этапе (4) либо (a), либо (b), но не оба действия одновременно.) Если она каким-то образом узнает, выполнение какого действия потребует от нее однонаправленная хэш-функция, то она сможет смошенничать. Однако, Пегги не сможет заставить однонаправленную хэш-функцию выдать определенный бит или догадаться, какой бит будет получен. Однонаправленная хэш-функция по сути является заменителем Виктора в случайном выборе одного из двух доказательств на этапе (4).

В неинтерактивном протоколе должно быть гораздо больше итераций в последовательности запрос/ответ. Пегги, а не Виктор, отбирает трудные проблемы с помощью случайных чисел. Она может подбирать различные проблемы, следовательно, и различные векторы вручения, до тех пор, пока хэш-функция не выдаст что-то, нужное Пегги. В интерактивном протоколе 10 итераций - вероятность мошенничества Пегги составит 1 шанс из 2^{10} (1 из 1024) - может быть достаточно. Однако, для неинтерактивных доказательств с нулевым знанием этого не хватит. Помните, что Мэллори всегда может выполнить на этапе (4) либо (a), либо (b). Он может, выполняя этапы (1)-(3), попытаться догадаться, что его попросят сделать, и посмотреть, правильно ли его предположение. Если нет, он попробует снова и снова. Сделать 1024 предположения на компьютере нетрудно. Для предотвращения такого вскрытия грубым взломом для неинтерактивных протоколов нужно 64 или даже 128 итераций.

Главная идея состоит в использовании однонаправленной хэш-функции - Пегги не может предсказать выход хэш-функции, потому что она не может предсказать ее вход. Вручения, используемые на входе, становятся известны только после решения новых проблем.

Общие замечания

Блюм (Blum) доказал, что любая математическая теорема может быть преобразована в граф, такой, что доказательство теоремы будет эквивалентно доказательству существования гамильтонова цикла для этого графа. В общем виде то, что для любого NP-полного утверждения есть доказательство с нулевым знанием, использующее однонаправленные функции и, следовательно, хорошие алгоритмы шифрования, доказано в [620]. Любое математическое доказательство может быть преобразовано в доказательство с нулевым знанием. Используя эту методику, исследователь может доказать миру, что ему известно доказательство конкретной теоремы, не раскрывая самого решения. Блюм мог опубликовать свои результаты, не раскрывая их.

Также существуют **доказательства с минимальным раскрытием** [590]. Для доказательства с минимальным раскрытием выполняются следующие свойства:

1. Пегги не может обмануть Виктора. Если Пегги не знает доказательства, ее шансы убедить Виктора в

том, что доказательство ей известно, пренебрежимо малы.

2. Виктор не может обмануть Пегги. Он не получает ни малейшего намека на доказательство кроме того факта, что доказательство известно Пегги. В частности, Виктор не может продемонстрировать доказательство никому другому, не доказав все сам с самого начала.

У доказательств с нулевым знанием есть дополнительное условие :

3. Виктор не узнает от Пегги ничего такого, чего он не смог бы узнать и самостоятельно кроме того факта, что доказательство известно Пегги.

Существует заметная математическая разница между доказательствами с минимальным раскрытием и доказательствами с нулевым знанием. Это различие находится вне рамок данной книги, но более искушенные читатели могут проштудировать другую литературу. Понятия изложены в in [626, 619, 622]. Дальнейшая проработка этих идей, основанная на различных математических предположениях, выполнена в [240, 319, 239].

Существуют различные типы доказательств с нулевым знанием :

— **Совершенное**. Существует имитатор, который создает стенограммы, полностью соответствующие реальным стенограммам (примеры с гамильтоновым циклом и изоморфизмом графов).

— **Статистическое**. Существует имитатор, который создает стенограммы, полностью соответствующие реальным стенограммам, кроме фиксированного числа исключений.

— **Вычислительное**. Существует имитатор, который создает стенограммы, неотличимые от реальных.

— **Неиспользующее**. Имитатора может и не быть, но мы можем доказать, что Виктор не узнает никакой информации из доказательства (параллельный пример)

Годы тяжелой работы, как теоретической, так и прикладной, присели к появлению доказательств с минимальным раскрытием и нулевым знанием. Майк Берместер (Mike Burmester) и Иво Десмедт изобрели широковещательно интерактивное доказательство, где владелец секрета может широковещательно передавать большой группе контролеров интерактивное доказательство с нулевым знанием [280]. Криптографы доказали, что *все*, что может быть доказано с помощью интерактивного доказательства, может быть доказано и с помощью интерактивного доказательства с нулевым знанием [753, 137].

Хорошей обзорной статьей по данной теме является [548]. Дополнительные математические подробности, варианты, протоколы и приложения ищите в [590, 619, 240, 319, 620, 113,241, 152, 8, 660, 238, 591, 617, 510, 592, 214, 104, 216, 832, 97, 939, 622, 482, 615, 618, 215, 476, 71]. *Много чего* было написано по этому вопросу.

5.2 Использование доказательства с нулевым знанием для идентификации

В реальном мире для доказательств подлинности часто используются физические символы: паспорта, водительские права, кредитные карточки и т.д. Эти символы содержат что-то, связывающее их с конкретным человеком: обычно фотографию или подпись, но с той же вероятностью это может быть отпечаток пальца, снимок сетчатки глаза или рентгеновский снимок челюсти. Как было бы здорово делать что-то подобное цифровым образом?

Использовать доказательства с нулевым знанием для доказательства идентичности было впервые предложено Уриелем Файгом (Uriel Feige), Амосом Фиатом (Amos Fiat) и Ади Шамиром [566, 567]. Закрытый ключ Алисы становится функцией ее "идентичности". Используя доказательство с нулевым знанием, она доказывает, что она знает свой закрытый ключ и, таким образом, свою идентичность. Соответствующие алгоритмы можно найти в разделе 23.11.

Это очень многообещающая идея. Она позволяет человеку доказать свою личность без использования физических символов. Однако, она не совершенна. Вот примеры возможных злоупотреблений.

Проблема гроссмейстера

Вот как Алиса, даже не зная правил шахмат, может обыграть гроссмейстера. (Иногда это называется проблемой гроссмейстера.) Она посылает вызов Гарри Каспарову и Анатолию Карпову, предлагая играть в одно время, в одном и том же месте, но в раздельных комнатах. Она играет белыми против Каспарова и черными против Карпова. Ни один гроссмейстер не знает о другом.

Карпов, играя белыми, делает свой ход первым. Алиса записывает ход и идет в комнату к Каспарову. Играя белыми, она делает тот же ход на доске Каспарова. Каспаров делает свой первый ход черными. Алиса записывает ход, идет в комнату к Карпову и делает тот же ход. Это продолжается, пока она не выигрывает одну из партий, проигрывая другую, или обе партии кончаются вничью.

На самом деле Каспаров играет с Карповым, а Алиса просто посредник, повторяющий ходы одного гросс-

мейстера на доске другого. Однако, если Карпов и Каспаров не знают о присутствии друг друга, каждый из них будет поражен игрой Алисы.

Этот способ мошенничества может быть использовать против доказательства личности с нулевым знанием [485, 120]. Когда Алиса доказывает свою личность Мэллори, Мэллори может одновременно доказать Бобу, что он-то и есть Алиса.

Обман, выполненный мафией

Обсуждая свой протокол идентификации с нулевым знанием, Ади Шамир сказал [1424]: "Я могу ходить в принадлежащий мафии магазин хоть миллион раз подряд, а они все еще не смогут выдать себя за меня."

Вот как мафия сможет это сделать. Алиса ест в ресторанчике Боба, принадлежащем мафии. Кэрол делает покупки в дорогом ювелирном магазине Дэйва. Боб и Кэрол - мафиози, переговаривающиеся по потайному радиоканалу. Алиса и Дэйв не подозревают о мошенничестве.

Когда Алиса поела и собралась платить и доказывать свою личность Бобу, Боб подает сигнал Кэрол, что пора начинать. Кэрол выбирает бриллианты подороже и собирается доказывать свою личность Дэйву. Теперь, пока Алиса доказывает свою личность Бобу, тот подает сигнал Кэрол, и та выполняет тот же протокол с Дэйвом. Когда Дэйв задает вопрос по протоколу, Кэрол сообщает этот вопрос Бобу, а Боб задает его Алисе. Когда Алиса отвечает, Боб передает правильный ответ Кэрол. По сути, Алиса просто доказывает свою личность Дэйву, а Боб и Кэрол просто, находясь внутри протокола, передают сообщения туда-сюда. Когда протокол завершается, Алиса доказала свою личность Дэйву и заплатила за дорогие бриллианты (с которыми Кэрол теперь и исчезнет).

Обман, выполненный террористами

Если Алиса хочет объединиться с Кэрол, то они также могут провести Дэйва. В этом протоколе Кэрол - известная террористка. Алиса помогает ей въехать в страну. Дэйв - офицер-пограничник, Алиса и Кэрол общаются по тайному радиоканалу.

Когда Дэйв задает Кэрол вопросы в соответствии по протоколу с нулевым знанием, Кэрол передает их Алисе, которая и отвечает на вопросы. Кэрол повторяет эти ответы Дэйву. Carol recites these answers to Dave. По сути, Свою личность Дэйву доказывает Алиса, а Кэрол выступает в роли линии связи. Когда протокол завершается, Дэйв считает, что Кэрол - это Алиса, и разрешает ей въехать в страну. Спустя три дня Кэрол всплывает у правительственного здания вместе с микроавтобусом, набитом взрывчаткой.

Предлагаемые решения

Оба описанных мошенничества возможны, так как заговорщики используют тайный радиоканал. Одним из способов предотвратить мошенничество является проведение процедуры идентификации в клетке Фарадея, блокирующей электромагнитное излучение. В примере с террористом это гарантирует, что Кэрол не получит ответов от Алисы. В примере с мафией Боб может построить фальшивую клетку Фарадея в своем ресторане, но у ювелира-то клетка будет работать, и Боб и Кэрол не смогут обмениваться сообщениями. Для решения проблемы гроссмейстера Алиса должна сидеть на своем стуле до конца игры.

Тотас Бот (Thomas Both) и Иво Десмедт предложили другое решение, использующее точные часы [148]. Если каждый этап протокола должен происходить в заданное время, у мошенников не останется времени для обмена информацией. В случае с проблемой гроссмейстера это соответствует предложению ограничить время обдумывания хода одной минутой - у Алисы не останется времени бегать из комнаты в комнату. В истории с мафией у Боб и Кэрол не хватит времени передавать друг другу ответы и вопросы.

Обман с несколькими личностями

Существуют и другие способы злоупотребить доказательствами идентичности с нулевым знанием, также рассматриваемые в [485, 120]. В ряде реализаций проверка при регистрации человеком своего ключа не производится. Следовательно, у Алисы может быть несколько закрытых ключей и, таким образом, несколько личностей. Это может здорово помочь ей, если она захочет мошенничать с налогами. Алиса также может совершить преступление и скрыться. В первых, она создает несколько личностей, одна из которых не используется. Затем, она использует эту личность для совершения преступления так, чтобы свидетель идентифицировал ее как эту личность. Затем, она немедленно прекращает пользоваться этой личностью. Свидетель знает личность преступника, но Алиса никогда больше не будет использовать эту личность - ее невозможно проследить.

Для защиты от этого нужны механизмы, обеспечивающие, чтобы у каждого человека была только одна личность. В [120] авторами предлагается причудливая идея защищенных от воровства детей, которые не могут быть клонированы, и у которых есть уникальный номер, являющийся частью их генетического кода. Они также предложили присваивать каждому ребенку личность при рождении. (Действительно, родителям придется сде-

лать это, так как иначе ребенок может быть украден.) Этим тоже легко злоупотребить - родители могут создать для родившегося ребенка несколько личностей. В конце концов, уникальность личности основана на доверии.

Прокат паспортов

Алиса хочет поехать в Заир, но правительство этой страны не дает ей визы. Кэрол предлагает сдать свою личность Алисе "напрокат". (Первым это предложил Боб, но возник ряд очевидных проблем.) Кэрол продает Алисе свой закрытый ключ и Алиса едет в Заир, выдавая себя за Кэрол.

Кэрол получает не только плату за свою личность, но и идеальное алиби. Она совершает преступление, пока Алиса находится в Заире. "Кэрол" доказала свою личность в Заире, как она могла совершить преступление дома?

Конечно, развязаны руки и у Алисы. Она может совершить преступление либо перед отъездом, либо сразу же после возвращения, около дома Кэрол. Сначала она покажет, что она - Кэрол (имея закрытый ключ Кэрол, ей не составит труда сделать это), затем она совершит преступление и убежит. Полиция будет искать Кэрол. Кэрол будет утверждать, что сдала свою личность напрокат Алисе, но кто поверит в такую невероятную историю?

Проблема в том, что Алиса доказывает не свою личность, а то, что ей известна некоторая секретная информация. Именно связь между этой информацией и личностью и служит предметом злоупотребления. Решение защищенных от воровства детей защитило бы от такого мошенничества, как и создание полицейского государства, в котором все граждане должны очень часто доказывать свою личность (в конце дня, на каждом углу и т.д.). Помочь решить эту проблему могут биометрические методы - отпечатки пальцев, снимки сетчатки глаза, запись голоса и т.п.

Доказательство членства

Алиса хочет доказать Бобу, что она является членом суперсекретной организации, но не хочет раскрывать свою личность. Эта проблема, близкая проблеме доказательства личности, но отличающаяся от нее, была изучена в [887, 906, 907, 1201, 1445]. Ряд решений связан с проблемой групповых подписей (см. раздел 4.6).

5.3 Слепые подписи

Важным свойством протоколов цифровой подписи является знание подписывающим содержания подписываемого документа. Это хорошее свойство, даже когда хочется обратного.

Мы можем пожелать, чтобы люди подписывали документы, даже не зная их содержания. Существуют способы, когда подписывающий может не точно, но *почти* знать, что он подписывает. Но все по порядку.

Полностью слепые подписи

Боб - государственный нотариус. Алиса хочет, чтобы он подписал документ, не имея ни малейшего представления о его содержании. Боб не отвечает за содержание документа, он только заверяет, что нотариально засвидетельствовал его в определенное время. Он собирается действовать по следующему плану:

(1) Алиса берет документ и умножает его на случайное число. Это случайное число называется **маскирующим множителем**.

(2) Алиса посылает замаскированный документ Бобу.

(3) Боб подписывает замаскированный документ.

(4) Алиса удаляет маскирующий множитель, получая оригинальный документ, подписанный Бобом.

Этот протокол работает только, если функция подписи и умножение коммутативны. Если нет, то помимо умножения существуют и другие способы изменить документ. Несколько подходящих алгоритмов приведены в разделе 23.12. А сейчас, для простоты математики остановимся на умножении.

Может ли боб смошенничать? Может ли он получить какую-нибудь информацию о том, что пописывает? Если множитель осторожности действительно случаен и делает замаскированный документ действительно случайным, то нет. Замаскированный документ, подписываемый Бобом на этапе, (2) ничем не похож на оригинальный документ Алисы. Замаскированный документ с подписью Боба на нем на этапе (3) ничем не похож на подписанный документ этапа (4). Даже если Боб заполучит документ со своей подписью после окончания протокола, он не сможет доказать (себе или кому-то другому), что он подписал его в этом конкретном протоколе. Он знает, что его подпись правильна. Он может, как и любой другой, проверить свою подпись. Однако, у него нет никакой возможности связать подписанный документ и любую информацию, полученную при выполнении протокола. Если он подписал, используя этот протокол, миллион документов, у него не будет способа узнать когда какой документ он подписал. Полностью слепые подписи обладают следующими свойствами:

1. Подпись Боба под документом правильна и служит доказательством того, что Боб подписал этот д окумент. Она убедит Боба в том, что он подписал этот документ, когда документ будет впоследствии показан Бобу. Она также обладает всеми свойствами цифровых подписей, обсуждаемых в разделе 2.6.

2. Боб не может связать подписанный документ с процессом подписания документа. Даже если у него хранятся записи обо всех сделанных им слепых подписях, он не сможет определить, когда он подп исал конкретный документ.

У Евы, находящейся между Алисой и Бобом и следящей за протоколом, информации еще меньше, чем у Б оба.

Слепые подписи

С помощью протокола полностью слепых подписей Алиса может заставить Боба подписать что-нибудь вр оде: "Боб должен Алисе миллион долларов", "Боб должен Алисе своего первого ребенка", "Боб должен Алисе ящик шоколада". Возможности бесконечны, и поэтому во многих приложениях этот протокол бесполезен..

Однако, существует способ, с помощью которого Боб может узнать, что он подписывает, вместе с тем сохр аняя полезные свойства слепых подписей. Центральным моментом этого протокола является техника "разрезать и выбрать". Рассмотрим следующий пример. Множество людей каждый день въезжают в некую страну, и Д епартамент иммиграции хочет удостовериться, что они не ввозят кокаин. Служащие могут обыскивать каждого, но вместо этого используется вероятностное решение - обыскивается каждый десятый въезжающий. Подвергается досмотру имущество одного человека из десяти, остальные девять пропускаются беспрепятственно. Постоянные контрабандисты в большинстве случаев проскакивают незамеченными, но с вероятностью 10 процентов их ловят. И если судебная система работает эффективно, наказание за единственную поимку на месте престу пления более чем перевешивает выгоды девяти удачных попыток.

Если Департамент иммиграции захочет повысить вероятность поимки контрабандистов, служащим придется обыскивать больше людей, захочет понизить вероятность - можно будет обыскивать меньше людей. Управляя вероятностями, можно контролировать эффективность протокола при поимке контрабандистов.

Протокол слепой подписи работает аналогичным образом. Боб получает большую пачку различных замаск ированных документов. Он **откроет**, например, все кроме одного и затем подпишет последний.

Посмотрите на замаскированный документ как на лежащий в конверте. Процесс маскировки документа можно рассматривать как помещение документа в конверт, а процесс удаления множителя маскировки - как вскрытие конверта. Когда документ спрятан в конверт, никто не сможет его прочитать. Документ подписывается с помощью кусочка копировальной бумаги, помещенной в конверт: Когда подписывающий ставит свою по дпись на конверте, с помощью кусочка копировальной бумаги эта подпись ставится и под документом.

В следующем сценарии действует группа агентов контрразведки. Их личности засекречены, даже само Управление контрразведки не знает, кто они такие. Директора Управления хочет выдать каждому агенту подп исанный документ следующего содержания: "Податель этого подписанного документа, (вставьте имя, под кот орым действует агент), обладает полной дипломатической неприкосновенностью". У каждого из агентов есть свой список псевдонимов, поэтому Управление не может просто раздать подписанные документы. Агенты не хотят передавать свои псевдонимы в Управление, так как враг мог вскрыть компьютер Управления. С другой стороны, Управление не хочет слепо подписывать документы, предоставленные агентом. Хитрый агент может представить сообщение, подобное следующему: "Агент (имя) вышел в отставку, и ему назначена ежегодная пенсия в миллион долларов. Подписано, Президент". В этом случае могут быть полезны слепые подписи.

Предположим, что у каждого агента по 10 псевдонимов, выбранных ими самими и больше никому неизвес тных. Предположим также, что агентам все равно, под каким именем они получат дипломатическую неприко сновенность. Также предположим, компьютер управления называется Agency's Large Intelligent Computing Engine (Большая Интеллектуальная Вычислительная Машина Управления), или ALICE, а нашим конкретным агентом является Bogota Operations Branch (Сектор операций в Боготе): BOB.

(1) BOB готовит *n* документов, каждый из которых использует различный псевдоним, дающих ему дипломатическую неприкосновенность.

(2) BOB маскирует каждый из документов отличным маскирующим множителем.

(3) BOB отправляет *n* документов ALICE.

(4) ALICE случайным образом выбирает *n-1* документ и просит BOB'а прислать маскирующий множитель для каждого из выбранных документов.

(5) BOB посылает ALICE соответствующие маскирующие множители.

(6) ALICE открывает (т.е., удаляет маскирующий множитель) *n-1* документ и убеждается в том, что они ко р-

ректны - и не являются разрешением на выплату пенсии.

(7) ALICE подписывает оставшийся документ и посылает его BOB'у.

(8) Агент удаляет маскирующий множитель и читает свой новый псевдоним: "Малиновая полоса." Подписанный документ дает ему дипломатическую неприкосновенность под этим именем.

Этот протокол надежно защищен от мошенничества BOB'а. Чтобы смошенничать, он должен точно угадать, какой документ ALICE не будет проверять. Вероятность этого - 1 шанс из *n* - не слишком велика. ALICE знает это и чувствует себя уверенно, подписывая документ, который она не сможет проверить. Для этого документа рассматриваемый протокол полностью совпадает с предыдущим протоколом полностью слепой подписи, сохраняя все свойства анонимности.

Существует трюк, который еще больше уменьшает вероятность мошенничества BOB'а. На этапе (4) ALICE случайным образом выбирает *n/2* документов для проверки, и BOB присылает ей соответствующий маскирующие множители на этапе (5). На этапе (7) ALICE перемножает все непроверенные документы и подписывает получившийся мегадокумент. На этапе (8) BOB удаляет все маскировочные множители. Подпись ALICE будет правильной, только если ею подписано произведение *n/2* идентичных документов. Чтобы смошенничать, BOB'у нужно точно угадать, какое подмножество документов будет проверять ALICE. Вероятность этого гораздо ниже, чем вероятность угадать единственный документ, который ALICE не проверяла.

BOB может смошенничать по другому. Он может создать два различных документа, один из которых ALICE согласна подписать, а другой - нет. Затем он может попытаться найти два различных маскирующих множителя, которые преобразуют указанные документы к одинаковому виду. Таким образом, если ALICE захочет проверить документ, BOB передаст ей маскирующий множитель, преобразующий документ к невинному виду. Если ALICE не захочет просмотреть документ и подпишет его, он применит тот маскирующий множитель, который преобразует замаскированный подписанный документ в документ, являющийся целью мошенничества. Хотя теоретически это и возможно, математика конкретных алгоритмов делает пренебрежимо малой вероятность для BOB'а найти такую пару. Действительно, она может быть столь низкой, как и вероятность Боба создать необходимую подпись под произвольным документом самостоятельно. Этот вопрос обсуждается ниже в разделе 23.12.

Патенты. Владельцем патентов на ряд особенностей слепых подписей является Чаум (Chaum) (см. 4-й).

Табл. 5-1. Патенты Чаума на слепые подписи

№ патента США	Дата	Название
4759063	19.07.88	Blind Signature Systems [323] (Системы слепых подписей)
4759064	19.07.88	Blind Unanticipated Signature Systems [324] (Системы слепых неожиданных подписей)
4914698	03.03.90	One-Show Blind Signature Systems [326] (Системы слепых подписей, показываемых один раз)
4949380	14.08.90	Returned-Value Blind Signature Systems [328] (Системы слепых подписей с возвращаемым значением)
4991210	05.02.91	Unpredictable Blind Signature Systems [331] (Системы непредсказуемых слепых подписей)

5.4 Личностная криптография с открытыми ключами

Алиса хочет отправить Бобу безопасное сообщение. Она не хочет получать свой открытый ключ с сервера ключей, она не хочет проверять подпись некоторой заслуживающей доверия третьей стороны на сертификате своего открытого ключа, и она даже не хочет хранить открытый ключ Боба в своем компьютере. Она хочет просто послать ему безопасное сообщение.

Эту проблему решают **личностные** криптосистемы, иногда называемые системами с Неинтерактивным разделением ключей (Non-Interactive Key Sharing, NIKS) [1422]. Открытый ключ Боба основывается на его имени и сетевом адресе (телефонном номере, почтовом адресе или чем-то подобном). В обычной криптографии с открытыми ключами Алисе нужен подписанный сертификат, связывающий личность Боба и его открытый ключ. В личностной криптографии открытый ключ Боба *и есть* его личность. Это действительно свежая идея является почти совершенной для почтовой системы - Если Алиса знает адрес Боба, она может безопасно посылать ему почту, что делает криптографию прозрачной, насколько это вообще возможно.

Система основана на выдаче Трентом ключей пользователям в зависимости от их личности. Если закрытый

ключ Алисы будет скомпрометирован, ей придется изменить одно из свойств, определяющих ее личность. Серьезной проблемой является проектирование системы таким образом, чтобы сговор нечестных пользователей не мог привести к подделке ключа.

При разработке математики таких схем, обеспечение безопасности которых оказалось зверски сложным, был выполнен большой объем работы - главным образом в Японии. Многие предложенные решения содержат выбор Трентом случайного числа для каждого пользователя - по моему, это угрожает самой идее таких систем. Ряд алгоритмов, рассматриваемых в главах 19 и 20, могут быть личностными. Подробности алгоритмов и криптосистем можно найти в [191, 1422, 891, 1022, 1515, 1202, 1196, 908, 692, 674, 1131, 1023, 1516, 1536, 1544, 63, 1210, 314, 313, 1545, 1539, 1543, 933, 1517, 748, 1228]. Алгоритм, который не использует случайных чисел, описан в [1035]. Система, обсуждаемая в [1546, 1547, 1507], ненадежна против вскрытия с использованием выбранного открытого ключа, то же самое можно сказать и о системе, предложенной как NIKS-TAS [1542, 1540, 1541, 993, 375, 1538]. По правде говоря, среди предложенного нет ничего одновременно практичного и безопасного.

5.5 Рассеянная передача

Криптограф Боб безнадежно пытается разложить на множители 500-битовое число n. Он знает, что оно является произведением пяти 100-битовых чисел, и ничего больше. (Вот проблема. Если он не восстановит ключ, ему придется работать сверхурочно, и он не попадет на еженедельную игру с Алисой в мысленный покер.) Что же делать? И вот появляется Алиса:

> "Мне посчастливилось узнать один из множителей числа", - говорит она, - "и я продам его тебе за 100 долларов. По доллару за бит." Показывая свою серьезность, она собирается использовать схему вручения бита, вручая каждый бит отдельно.
>
> Боб заинтересован, но только за 50 долларов. Алиса не хочет сбрасывать цену и предлагает продать Бобу половину битов за половину стоимости. "Это заметно сократит тебе работу", -.
>
> "Но как я узнаю, что твое число действительно является множителем n. Если ты покажешь мне число и позволишь мне убедиться, что оно действительно является множителем, я соглашусь с твоими условиями", - говорит Боб.
>
> Они в патовой ситуации. Алиса не может убедить Боба в том, что она знает сомножитель n, не раскрыв его, а Боб не хочет покупать 50 битов, которые вполне могут оказаться бесполезными.

Эта история, утащенная у Джо Килиана [831], вводит понятие **рассеянной передачи**. Алиса передает Бобу группу сообщений. Боб получает некоторое подмножество этих сообщений, но Алиса не знает, какие из сообщений Боб получил. Однако, это не полностью решает проблему. Когда Боб получит случайную половину битов, Алисе придется убеждать его, используя доказательство с нулевым знанием, что она послала часть множителя n.

В следующем протоколе Алиса посылает Бобу одно из двух сообщений. Боб получает сообщение, но какое - Алиса не знает.

(1) Алиса генерирует две пары открытый ключ/закрытый ключ, всего четыре ключа. Она посылает оба открытых ключа Бобу.

(2) Боб выбирает ключ симметричного алгоритма (например, DES). Он выбирает один из открытых ключей Алисы и шифрует с его помощью свой ключ DES. Он посылает шифрованный ключ Алисе, не сообщая, какой из ее открытых ключей он использовал для шифрования.

(3) Алиса дважды расшифровывает ключ Боба, используя оба своих закрытых ключа. В одном из случаев она использует правильный ключ и успешно расшифровывает ключ DES, присланный Бобом. В другом случае она использует неправильный ключ и получает бессмысленную последовательность битов, которая, тем не менее, похожа на случайный ключ DES. Так как ей неизвестен правильный открытый текст, она не может узнать, какой из ключей правилен.

(4) Алиса зашифровывает каждое из своих сообщений каждым из ключей, полученных ею на предыдущем этапе (один из которых - настоящий, а другой - бессмысленный), и посылает оба сообщения Бобу.

(5) Боб получает сообщения Алисы, одно из которых зашифровано правильным ключом DES, а другое - бессмысленным ключом DES. Когда Боб расшифровывает каждое из этих сообщений своим ключом DES, он может прочитать одно из них, а другое будет для него выглядеть полной бессмыслицей.

Теперь у Боба есть два сообщения Алисы, и Алиса не знает, какое из них Бобу удалось успешно расшифровать. К несчастью, если протокол остановится на этом этапе, Алиса сможет смошенничать. Необходим еще один этап.

(6) Когда протокол завершится, и станут известны оба возможных результата передачи, Алиса должна передать Бобу свои закрытые ключи, чтобы он убедиться в отсутствии мошенничества. В конце концов, она могла зашифровать на этапе (4) обоими ключами одно и то же сообщение.

В этот момент, конечно же Боб сможет узнать и второе сообщение.

Протокол надежно защищен от взлома со стороны Алисы, потому что у нее нет возможности узнать, какой из двух ключей DES является настоящим. Оба из них она использует для шифрования своих сообщений, но Боб может успешно расшифровать только одно из них - до этапа (6). Протокол защищен и от взлома со стороны Боба, потому что до этапа (6) он не сможет получить закрытый ключ Алисы, чтобы определить ключ DES, которым зашифровано другое сообщение. На вид этот протокол может показаться просто усложненным способом бросать "честную" монету по модему, но он интенсивно используется во многих сложных протоколах.

Конечно же, ничто не может помешать Алисе послать Бобу два совершенно бессмысленных сообщения: "Мяу-мяу" и "Ты молокосос". Этот протокол гарантирует, что Алиса передаст Бобу одно из двух сообщений, но нет никакой гарантии, что Боб захочет получить любое из них.

В литературе можно найти и другие протоколы рассеянной передачи. Некоторые из них неинтерактивны, т.е. Алиса публикует свои два сообщения, а Боб может прочесть только одно из них. Он может сделать это, когда захочет, ему не нужно для этого связываться с Алисой [105].

В действительности на практике никто не использует протокол рассеянной передачи, но это понятие является важным блоком при построении других протоколов. Хотя существует много типов рассеянной передачи - у меня есть два секрета, а вы получаете один, у меня есть n секретов, а вы получаете один, у меня есть один секрет, который вы получаете с вероятностью 1/2 и так далее - все они эквивалентны [245, 391, 395].

5.6 Рассеянные подписи

Честно говоря, я не могу придумать, чего их можно использовать, но существует два типа рассеянных подписей [346]:

1. У Алисы есть *n* различных сообщений. Боб может выбрать одно из них, чтобы Алиса его подписала, и у Алисы не будет способа узнать, что же она подписала.
2. У Алисы есть единственное сообщение. Боб может выбрать один из *n* ключей, которым Алиса подпишет сообщение, и Алиса не сможет узнать, какой ключ она использовала.

Идея изящна, я уверен, что где-нибудь она найдет применение.

5.7 Одновременная подпись контракта

Подпись контракта с помощью посредника

Алиса и Боб хотят заключить контракт. Они достигли согласия на словах, но никто не хочет ставить свою подпись, пока не поставлена подпись другого. При личной встрече это не вызывает затруднений - оба подписывают вместе. На расстоянии они могут обратиться к посреднику.

(1) Алиса подписывает копию контракта и посылает ее Тренту.

(2) Боб подписывает копию контракта и посылает ее Тренту.

(3) Трент посылает сообщение и Алисе, и Бобу, сообщающее, что другой партнер подписал контракт.

(4) Алиса подписывает две копии контракта и посылает их Бобу.

(5) Боб подписывает обе копии контракта, сохраняет одну для себя, и посылает другую Алисе.

(6) Алиса и Боб сообщают Тренту, что у каждого из них есть подписанная обоими партнерами копия контракта.

(7) Трент уничтожает свои две копии контракта, с единственной подписью под каждым.

Этот протокол работает, потому что Трент защищает любую из сторон от мошенничества другой. Если Боб попытается отказаться от подписи под контрактом на этапе (5), Алиса может обратиться к Тренту за копией контракта, уже подписанного Бобом. Если Алиса попытается отказаться от подписи под контрактом на этапе (4), Боб может сделать то же самое. Когда Трент сообщает, что он получил оба контракта на этапе (3), Алиса и Боб узнают, что другой партнер уже подписал контракт. Если Трент не получит оба контракта на этапах (1) и (2), он уничтожает имеющуюся у него копию, и ни одна из сторон не связана более обязательствами контракта.

Одновременная подпись контракта без посредника (лицом к лицу)

Если Алиса и Боб встречаются лицом к лицу, они могут подписать контракт следующим образом [1244]:

(1) Алиса пишет первую букву своего имени и передает контракт Бобу.

(2) Боб пишет первую букву своего имени и передает контракт Алисе.

(3) Алиса пишет вторую букву своего имени и передает контракт Бобу.

(4) Боб пишет вторую букву своего имени и передает контракт Алисе.

(5) Это продолжается до тех пор, пока Алиса и Боб не напишут свои имена полностью.

Если пренебречь очевидной проблемой протокола (имя Алисы длиннее имени Боба), то он работает дост аточно хорошо. Написав только одну букву из подписи, Алиса знает, что никакой судья не станет заставлять ее выполнять условия контракта. Но написанная буква - это акт доброй воли, и Боб отвечает аналогичным дейс твием.

Когда каждая из сторон напишет несколько букв подписи, судья вероятно сможет убедиться, что обе стороны подписали контракт. Хотя, если вглядеться, ситуация весьма туманна. Конечно же, то, что контракт не вступает в силу после написания первых букв, так же очевидно как и то, что контракт становится действующим после того, как стороны напишут свои имена. В каком месте протокола стороны оказываются связанными контрактом? Написав половину своих имен? Две трети? Три четверти?

Так как ни Алиса, ни Боб не знают точно, с какого момента начинает действовать контракт, то у каждого из них на протяжении всего протокола есть опасение, что для него или для нее контракт уже вступил в силу . Не существует этапа протокола, на котором Боб смог бы сказать : "Вы написали четыре буквы подписи, а я только три. Вы связаны контрактом, а я нет." У Боба нет причин прекращать выполнение протокола. Более того, чем дольше стороны выполняют протокол, тем больше вероятность того, что судья решит, что контракт вступил в силу. И снова, нет причины прерывать протокол. В конце концов, они оба хотели подписать контракт, они пр осто не хотели подписывать его раньше другого партнера.

Одновременная подпись контракта без посредника (без личной встречи)

В этом протоколе используется такая же неопределенность [138]. Алиса и Боб по очереди движутся детскими шажками к подписанию протокола.

В этом протоколе Алиса и Боб обмениваются рядом подписанных сообщений вида : "Я согласен, что с вероятностью p я связан условиями контракта."

Получатель сообщения может предъявить его судье и, с вероятностью p, судья признает контракт подписа нным.

(1) Алиса и Боб согласовывают дату окончания подписания контракта .

(2) Алиса и Боб договариваются о различии вероятностей, которым они собираются пользоваться. Например, Алиса может решить, что ее вероятность быть связанной условиями контракта не должна превышать в е роятность Боба больше, чем на 2 процента. Обозначим различие Алисы как a, различие Боба - как b.

(3) Алиса посылает Бобу подписанное сообщение с вероятностью $p = a$.

(4) Боб посылает Алисе подписанное сообщение с $p = a + b$.

(5) Пусть p - это вероятность из сообщения, полученного Алисой от Боба на предыдущем этапе . Алиса посылает Бобу подписанное сообщение с $p' = p + a$ или 1, смотря что меньше.

(6) Пусть p - это вероятность из сообщения, полученного Бобом от Алисы на предыдущем этапе . Боб посылает Алисе подписанное сообщение с $p' = p + b$ или 1, смотря что меньше.

(7) Алиса и Боб продолжают выполнять этапы (5) и (6) до тех пор, пока они оба не получат сообщения с $p = 1$ или пока не наступит согласованная на этапе (1) дата .

По мере выполнения протокола и Алиса, и Боб соглашаются, что они оказываются связанными контрактом со все большей и большей вероятностью. Например, Алиса может определить свое a как 2 процента, а Боб свое b - как 1 процент. (Лучше бы они выбрали большие приращения, а то мы застрянем на этом месте.) В первом сообщении Алиса сообщает, что она связана контрактом с вероятностью 2 процента . Боб может ответить, что он связан контрактом с вероятностью 3 процента. Следующее сообщение Алисы может утверждать, что она связ ана контрактом с вероятностью 5 процента и так далее, пока вероятности обоих не достигнут 100 процентов.

Если и Алиса, и Боб завершили протокол к оговоренной дате , то все прекрасно. В противном случае, любая из сторон может предъявить контракт судье вместе с подписанным последним сообщением другой стороны. Прежде, чем просмотреть контракт, судья случайным образом выбирает число между 0 и 1. Если это число меньше вероятности, подписанной второй стороной, то обе стороны связаны контрактом. Если это число больше вероятности, подписанной второй стороной, то обе стороны не связаны контрактом. (Затем судья сохраняет использованное число на случай решения другого вопроса, касающегося того же контракта .) Именно это и означает "быть связанным контрактом с вероятностью p".

Это базовый протокол, но могут использоваться и дополнительные усложнения . Судья может принимать ре-

шение в отсутствие одной из сторон. Решение судьи связывает контрактом либо обе стороны, либо ни одну из них. Не существует ситуации, когда одна из сторон связана контрактом, а другая - нет. Более того, протокол завершится, как только одна из сторон захочет иметь хоть немного большую, чем другая, вероятность быть связанной контрактом.

Одновременная подпись контракта без посредника (с помощью криптографии)

Этот криптографический протокол использует тот же принцип детских шажков [529]. Для определенности используется DES, хотя вместо него может быть любой симметричный алгоритм.

(1) И Алиса, и Боб случайным образом выбирают $2n$ ключей DES, сгруппированных попарно. В парах нет ничего особенного, это просто способ группировки для данного протокола.

(2) И Алиса, и Боб создают n пар сообщений, L_i и R_i, например, "Это левая половина моей i-ой подписи" и "Это правая половина моей i-ой подписи". Идентификатор, i, меняется от 1 до n. В каждое сообщение, вероятно, также будет входить цифровая подпись контракта и метка времени. Контракт считается подписанным, если другая сторона может предъявить обе половины, L_i и R_i, одной пары подписей.

(3) И Алиса, и Боб шифруют свои пары сообщений парами ключей DES, левое сообщение - левым ключом в паре, а правое - правым.

(4) Алиса и Боб посылают друг другу свои пачки из $2n$ шифрованных сообщений, поясняя, какие сообщения какими половинами каких пар являются.

(5) Алиса и Боб посылают друг другу все пары ключей, используя протокол рассеянной передачи для каждой пары. То есть, Алиса посылает Бобу независимо для каждой из n пар ключей либо ключ, использованный для шифрования левого сообщения, либо ключ, использованный для шифрования правого сообщения. Боб делает то же самое. Они могут посылать свои половинки по очереди, или сначала один может послать все 100, а потом другой - это не имеет значения. Теперь и у Алисы, и у Боба есть по одному ключу из каждой пары, но никто не знает, какие из половинок получил партнер.

(6) Алиса и Боб, используя полученные ключи, расшифровывают те половинки сообщений, которые они могут расшифровать. Они убеждаются, что расшифрованные сообщения правильны.

(7) Алиса и Боб посылают друг другу первые биты всех $2n$ ключей DES.

(8) Алиса и Боб повторяют этап (7) для вторых битов всех $2n$ ключей DES, затем третьих битов и так далее, пока все биты всех ключей DES не будут переданы.

(9) Алиса и Боб расшифровывают оставшиеся половинки сообщений, и контракт подписан.

(10) Алиса и Боб обмениваются закрытыми ключами, использованными для протокола рассеянной передачи на этапе (5), и каждый из них убеждается в отсутствии мошенничества.

Почему Алисе и Бобу нужно выполнить всю эту нудную последовательность действий? Предположим, что Алиса хочет смошенничать, и посмотрим, что получится. На этапах (4) и (5) Алиса могла бы разрушить протокол, послав Бобу ничего не значащие строки. Боб обнаружил бы это на этапе (6) при попытке расшифровать полученные половинки. Боб с полной безопасностью может остановиться до того, как Алиса сможет расшифровать любую из пар сообщений Боба.

Если бы Алиса была очень хитрой, она могла бы разрушить только половину протокола. Она могла бы послать одну половинку из каждой пары правильно, а вместо второй отправить бессмысленные строки. Вероятность Боба получить правильную половинку составит только 50 процентов, поэтому в половине случаев мошенничество Алисы удастся, но только для одной пары. Если бы использовались только две пары, этот способ мошенничества удастся в 25 процентах случаев. Вот почему n должно быть велико. Алисе необходимо точно угадать результат n протоколов рассеянной передачи, ее вероятность добиться этого составляет 1 шанс из 2^n. Если $n = 10$, у Алисы 1 шанс из 1024 обмануть Боба.

Алиса также может отправить Бобу случайные биты на этапе (8). Возможно, Боб не узнает, что она послала ему случайные биты, пока не получит весь ключ и не попытается расшифровать половинки сообщения. Но снова вероятность на стороне Боба. Он уже получил половину ключей, и Алиса не знает какую. Если n достаточно велико, Алиса наверняка пришлет ему бессмысленный бит для ключа, который у него уже есть, и он немедленно узнает, что она пытается его обмануть.

Возможно, Алиса будет выполнять этап (8) до тех пор, пока она не получит достаточно битов ключей для вскрытия грубым взломом, и затем прекратит передачу битов. Длина ключа DES - 56 битов. Если она получила 40 из 56 битов, ей останется перебрать 2^{16}, или 65536, ключей для дешифровки сообщения, а эта задача, определенно, по силам современным компьютерам. Но Боб получит ровно столько же битов ее ключей (или, в худшем случае, на один бит меньше), следовательно, он сможет сделать то же самое. У Алисы нет другого выбора, кроме как продолжать следовать протоколу.

Идея в том, что Алисе придется играть честно, потому что вероятность обмануть Боба слишком мала. В конце протокола у обеих сторон есть n подписанных пар сообщений, любое из которых достаточно для правильной подписи.

У Алисы есть только один способ смошенничать - она может послать Бобу одинаковые сообщения на этапе (5). Боб не сможет обнаружить этого до окончания протокола, но он сможет использовать стенограмму протокола, чтобы убедить судью в двуличности Алисы.

Протоколы этого типа имеют два слабых места [138]. Во первых, проблема возникает, если вычислительная мощь одной стороны гораздо больше, чем у другой. Если, например, Алиса может выполнить вскрытие грубым взломом быстрее Боба, то она рано прекратит передачу битов на этапе (8) и раскроет ключи Боба самостоятельно. Бобу, которому для таких же действий просто не хватит времени, не повезет.

Во вторых, проблема возникает, если одна из сторон прекращает протокол раньше времени. Если Алиса оборвет выполнение протокола, оба столкнутся с одинаковыми вычислительными проблемами, но у не хватит ресурсов завершить вычисления к нужному сроку. Проблема появляется, к примеру, если контракт определяет, что Алиса должна сделать что-то через неделю, а она прерывает протокол в тот момент, когда Бобу для вычисления ее подписи потребуется целый год расчетов. Реальная сложность при этом заключается в близкой дате предмета контракта, к которой процесс не будет завершен одной или обеими подписывающими сторонами.

Эти проблемы существуют также для протоколов разделов 5.8 и 5.9.

5.8 Электронная почта с подтверждением

Такой же протокол одновременной рассеянной передачи, использованный для подписания контракта, с небольшими изменениями используется для электронной почты с подтверждением [529]. Пусть Алиса хочет послать сообщение Бобу, но не хочет, чтобы он прочитал его, не расписавшись в получении. В реальной жизни это обеспечивается неприветливыми почтовыми служащими, но то же самое может быть сделано при помощи криптографии. Эту проблему первым рассмотрел Уитфилд Диффи в [490].

На первый взгляд, эту проблему мог бы решить протокол одновременного подписания контракта. Алиса просто копирует свое сообщение ключом DES. Ее половина протокола выглядит примерно так: "Это левая половина ключа DES: 32f5", а половина протокола Боба - так: "Это левая половина моей квитанции." Все остальное не меняется.

Чтобы понять, почему это не работает, вспомните, что протокол опирается на то, что рассеянная передача на этапе (5) предохраняет от мошенничества обе стороны. Оба партнера знают, что они послали другой стороне правильную половину, но никто не знает какую. Они не мошенничают на этапе (8), потому что вероятность выйти сухим из воды чрезвычайно мала. Если Алиса посылает Бобу не сообщение, а половину ключа DES, то Боб не может проверить правильность ключа DES на этапе (6). Алиса же может проверить правильность квитанции Боба, поэтому Бобу придется быть честным. Алиса легко может отправить Бобу неправильный ключ а когда он обнаружит это, его квитанция уже будет у Алисы. Вот невезуха, Боб.

Решение этой проблемы потребует некоторой коррекции протокола:

(1) Алиса шифрует свое сообщение случайным ключом DES и посылает его Бобу.

(2) Алиса создает n пар ключей DES. Первый ключ каждой пары генерируется случайным образом, а второй представляет собой XOR первого ключа и ключа шифрования сообщения.

(3) Алиса шифрует сообщение-заглушку каждым из своих $2n$ ключей.

(4) Алиса посылает Бобу всю пачку шифрованных сообщений, проверяя, что он знает, какие сообщения какими половинами каких пар являются.

(5) Боб создает n пар случайных ключей DES.

(6) Боб создает пару сообщений, образующих правильную квитанцию. Хорошим вариантами могут служить "Это левая половина моей квитанции" и "Это левая половина моей квитанции" с добавлением какой-нибудь строки случайных битов. Он создает n пар квитанций, нумеруя каждую. Как и в предыдущем протоколе квитанция считается правильной, если Алиса может предъявить обе половины квитанции (с одним и тем же номером) и все ее ключи шифрования.

(7) Боб шифрует каждую свою пару сообщений парами ключей DES, i-ую пару сообщений - i-ой парой ключей, левое сообщение - левым ключом в паре, а правое - правым. в паре.

(8) Боб посылает Алисе свою пачку шифрованных сообщений, проверяя, что она знает, какие сообщения какими половинами каких пар являются.

(9) Алиса и Боб посылают друг другу все пары ключей, используя протокол рассеянной передачи. То есть,

Алиса посылает Бобу независимо для каждой из n пар ключей либо ключ, использованный для шифрования левого сообщения, либо ключ, использованный для шифрования правого сообщения. Боб делает то же самое. Они могут посылать свои половинки по очереди, или сначала один может послать все n, а потом другой - это не имеет значения. Теперь и у Алисы, и у Боба есть по одному ключу из каждой пары, но никто не знает, какие из половинок получил партнер.

(10) Алиса и Боб расшифровывают те половинки сообщений, которые могут и убеждаются, что расшифрованные сообщения правильны.

(11) Алиса и Боб посылают друг другу первые биты всех $2n$ ключей DES. (Если они беспокоятся, что Ева сможет прочитать эти почтовые сообщения, то они должны шифровать свой обмен битами).

(12) Алиса и Боб повторяют этап (11) для вторых битов всех $2n$ ключей DES, затем третьих битов и так далее, пока все биты всех ключей DES не будут переданы.

(13) Алиса и Боб расшифровывают оставшиеся половинки сообщений. Алиса получает правильную квитанцию от Боба, а Боб может выполнить "исключающее или" для любой пары ключей и получить ключ, которым зашифровано оригинальное сообщение.

(14) Алиса и Боб обмениваются закрытыми ключами, использованными для протокола рассеянной передачи, и каждый из них убеждается в отсутствии мошенничества.

Этапы (5)-(8) для Боба и (9)-(12) для обеих сторон не меняются по сравнению с протоколом подписания контракта. Отличие - в сообщениях-заглушках Алисы. Они предоставляют Бобу возможность проверить правильность ее рассеянной передачи на этапе (10), что заставляет ее оставаться честной на этапах (11)-(13). И, как и для протокола одновременного подписания контракта, для выполнения протокола требуются обе половины одного из сообщений Алисы.

5.9 Одновременный обмен секретами

Алиса знает секрет A, а Боб - секрет B. Алиса собирается сообщить Бобу A, если он расскажет ей B. Боб хочет сообщить Алисе B, если она расскажет ему A. Следующий протокол, подслушанный на школьном дворе, работать не будет:

(1) Алиса: "Я скажу, если ты скажешь мне первым."

(2) Боб: "Я скажу, если ты скажешь мне первой."

(3) Алиса: "Нет, ты первый."

(4) Боб: "Ну, хорошо." Боб шепчет Алисе.

(5) Алиса: "Ха, а я тебе не скажу."

(6) Боб: "Это не честно."

Честным это может сделать криптография. Предыдущие два протокола являются реализациями более общего протокола, который и позволит Алисе и Бобу одновременно обменяться секретами [529]. Чтобы не повторять полностью весь протокол, я набросаю необходимые изменения протокола почты с подтверждением.

Алиса выполняет этапы (1)-(4), используя в качестве сообщения A. Боб выполняет эти же действия, используя в качестве своего сообщения B. Алиса и Боб выполняют рассеянную передачу на этапе (9), расшифровывают на этапе (10) те половинки, которые могут, и выполняют необходимые итерации на этапах (11) и (12). Чтобы защититься от Евы, они должны шифровать свои сообщения. Наконец, и Алиса, и Боб расшифровывают оставшиеся половины пар сообщения и выполняют XOR для любой пары ключей, чтобы получить ключи, которыми зашифрованы оригинальные сообщения.

Этот протокол позволяет Алисе и Бобу одновременно обмениваться секретами, но не гарантирует качества переданных секретов. Алиса может пообещать Бобу план лабиринта Минотавра и прислать ему схему Бостонского метро. Боб получит только тот секрет, который Алиса пришлет ему. Другие протоколы описаны в [1286, 195, 991, 1524, 705, 753, 259, 358, 415].

Глава 6
Эзотерические протоколы

6.1 Безопасные выборы

Компьютерное голосование никогда не будет использовано для обычных выборов, пока не появится прот о-кол, который одновременно предохраняет от мошенничества и защищает тайну личности. Идеальный протокол должен обладать, по меньшей мере, следующими шестью свойствами :

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 .

Кроме того, для некоторых схем голосования может понадобиться следующее требование :

7. Каждый знает, кто голосовал, а кто нет.

Прежде чем описывать сложные протоколы, имеющие приведенные характеристики, давайте взглянем на рад протоколов попроще.

Упрощенный протокол голосования №1

(1) Каждый голосующий шифрует свой бюллетень открытым ключом Центральной избирательной комиссии (ЦИК).

(2) Каждый голосующий посылает свой бюллетень в ЦИК .

(3) ЦИК расшифровывает бюллетени, подводит итоги и опубликовывает результаты голосования .

Этот протокол просто кишит проблемами. ЦИК не может узнать, откуда получены бюллетени, и даже, пр и-надлежат ли присланные бюллетени правомочным избирателям . У нее нет ни малейшего представления о том, не голосовали ли правомочные избиратели больше одного раза . Положительной стороной является невозмо ж-ность изменить бюллетень другого человека , но никто и не будет пытаться это сделать, потому что гораздо г о-лосовать повторно, добиваясь нужных результатов выборов .

Упрощенный протокол голосования №2

(1) Каждый голосующий подписывает свой бюллетень своим закрытым ключом .

(2) Каждый голосующий шифрует свой бюллетень открытым ключом ЦИК .

(3) Каждый голосующий посылает свой бюллетень в ЦИК .

(4) ЦИК расшифровывает бюллетени, проверяет подписи, подводит итоги и опубликовывает результаты г о-лосования.

Этот протокол обладает свойствами 1 и 2 : Только правомочные избиратели могут голосовать, и никто не может голосовать более одного раза - ЦИК может записывать бюллетени, полученные на этапе (3). Каждый бюллетень подписан закрытым ключом голосующего, поэтому ЦИК знает, кто голосовал, а кто нет, и, как гол о-совал каждый избиратель. Если получен бюллетень, который не подписан правомочным пользователем, или бюллетень, подписанный избирателем, который уже проголосовал , то такой бюллетень игнорируется комиссией. Кроме того, из-за цифровой подписи никто не может изменить бюллетень другого избирателя, даже если сумеет перехватить его на этапе (2).

Проблема этого протокола в том, что подпись добавляется к бюллетеню, ЦИК знает, кто за кого голосовал . Шифрование бюллетеней открытым ключом ЦИК мешает посторонним злоупотреблять протоколом и узнавать, кто за кого голосовал, но вам придется полностью доверять ЦИК Это как будто в кабинке для голосования вам через плечо заглядывает электронный судья .

Два следующих примера показывают, как трудно обеспечить хотя бы первые три требования к протоколу безопасного голосования.

Голосование со слепыми подписями

Нам нужно как-то отделить бюллетень от голосующего, сохранив процедуру идентификации личности. Именно это можно сделать с помощью протокола слепой подписи.

(1) Каждый избиратель создает 10 наборов сообщений, каждый набор содержит правильный бюллетень для каждого возможного результата (например, если бюллетенем является один из ответов "да"-"нет", то каждый набор состоит из двух бюллетеней, одного для "да", а другого для "нет"). Каждое сообщение содержит также случайным образом созданный идентификационный номер, достаточно большой, чтобы избежать путаницы с другими избирателями.

(2) Каждый избиратель лично маскирует все сообщения (см. раздел 5.3) и посылает их в ЦИК вместе с маскирующим множителями.

(3) ЦИК по своей базе данных проверяет, что пользователь не присылал раньше для подписания свои замаскированные бюллетени. ЦИК открывает 9 из 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.

(4) Избиратель снимает маскировку с сообщений и получает набор бюллетеней, подписанных ЦИК. (Эти бюллетени подписаны, но не зашифрованы, поэтому избиратель легко увидит, какой из бюллетеней - "да", а какой - "нет".)

(5) Каждый избиратель выбирает один из бюллетеней (о, демократия!) и шифрует его открытым ключом ЦИК.

(6) Избиратель отправляет свой бюллетень.

(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 последовательным номером и соответствующим бюллетенем.

Мэллори, избиратель-жулик, не может обмануть эту систему. Протокол слепой подписи обеспечивает единственность его бюллетени. Если он попытается отправить тот же бюллетень дважды, ЦИК обнаружит дублирование последовательных номеров на этапе (7) и не будет учитывать второй бюллетень. Если он попытается получить несколько бюллетеней на этапе (2), ЦИК обнаружит это на этапе (3). Мэллори не может создать свои собственные бюллетени, потому что он не знает закрытого ключа комиссии. По той же причине он не может перехватить и изменить чужие бюллетени.

Протокол "разрезать и выбрать" на этапе (3) должен обеспечить уникальность бюллетеней. Без этого этапа Мэллори мог бы создать точно такой же, за исключением идентификационного номера, набор бюллетеней и заверить их все в ЦИК.

Мошенническая ЦИК не сможет узнать, как голосовал конкретный избиратель. Так как протокол слепой подписи маскирует последовательные номера бюллетеней до момента подведения итогов, ЦИК не сможет установить связь между подписанным ею замаскированным бюллетенем и подытоживаемым бюллетенем. Опубликование перечня последовательных номеров и связанных с ними бюллетеней позволяет пользователям убедиться, что их бюллетени были правильно учтены.

Но проблемы все еще остаются. Если этап (6) не анонимен, и ЦИК может записать, кто какой бюллетень прислал, то она сможет узнать, кто за кого голосовал. Однако, это невозможно, если комиссия получает бюллетени в запечатанной урне для голосования и считает их позже. Хотя ЦИК и не сможет установить связь между избирателями и их бюллетенями, она сможет создать большое количество подписанных и правильных бюллетеней и смошенничать, прислав их сама себе. И если Алиса обнаружит, что ЦИК подменила ее бюллетень, она не сможет доказать этого. Аналогичный протокол, пытающийся устранить эти проблемы, описан в [1195, 1370].

Голосование с двумя Центральными комиссиями

Одним из решений является разделить ЦИК пополам. Ни у одной из них не будет достаточно власти, чтобы смошенничать по своему усмотрению.

В следующем протоколе используется Центральное управление регистрации (ЦУР), занимающееся проверкой пользователей, и отдельная ЦИК для подсчета бюллетеней [1373].

(1) Каждый избиратель отправляет письмо в ЦУР, запрашивая регистрационный номер.

(2) ЦУР возвращает избирателю случайный регистрационный номер. ЦУР ведет список регистрационных номеров. Кроме того, ЦУР хранит список получателей регистрационных номеров на случай, если кто-то попытается проголосовать дважды.

(3) ЦУР отправляет список регистрационных номеров в ЦИК.

(4) Каждый избиратель выбирает случайный идентификационный номер. Он создает сообщение с этим номером, регистрационным номером, полученным в ЦУР, и своим бюллетенем. Он посылает это сообщение в ЦИК.

(5) ЦИК проверяет регистрационные номера по списку, полученному от ЦУР на этапе (3). Если регистрационный номер есть в списке, ЦИК вычеркивает его (чтобы избежать повторного голосования). ЦИК добавляет идентификационный номер к списку тех, кто проголосовал за определенного кандидата, и прибавляет единичку к соответствующему итоговому числу.

(6) После того, как все бюллетени будут получены, ЦИК публикует результаты вместе со списками, содержащими идентификационные номера и соответствующие бюллетени.

Как и в предыдущем протоколе каждый избиратель может увидеть список идентификационных номеров и найти в нем свой собственный. Так он может убедиться, что его бюллетень учтен. Конечно, все сообщения, которыми обмениваются участники протокола должны быть зашифрованы и подписаны, чтобы помешать кому-нибудь выдать себя за другого или перехватить сообщения.

ЦИК не может изменить бюллетени, потому что каждый избиратель будет искать свой регистрационный номер. Если избиратель не находит свой регистрационный номер или находит его в итоговом списке с другим результатом голосования, он немедленно узнает, что произошел обман. ЦИК не может добавить бюллетень в урну, которая находится под наблюдением ЦУР. ЦУР знает, сколько избирателей зарегистрировалось, их регистрационные номера и обнаружит любые изменения.

Мэллори, не обладающий избирательными правами, может попытаться смошенничать, угадав правильный регистрационный номер. Угроза этого может быть минимизирована, если множество возможных регистрационных номеров намного больше, чем множество реальных регистрационных номеров: 100-битовое число для миллиона избирателей. Конечно же, регистрационные номера должны генерироваться случайным образом.

Несмотря на это, ЦУР должна быть заслуживающим доверия органом власти - ведь она может зарегистрировать неправомочных избирателей. Она также может зарегистрировать правомочных избирателей несколько раз. Этот риск может быть сведен к минимуму, если ЦУР опубликует список зарегистрировавшихся избирателей (но без их регистрационных номеров). Если число избирателей в этом списке меньше, чем число подсчитанных бюллетеней, то что-то не так. Однако, если зарегистрировалось больше избирателей, чем было прислано бюллетеней, то это, возможно, означает, что ряд зарегистрировавшихся избирателей не проголосовал. Многие, зарегистрировавшись, не утруждаются бросить в урну свой бюллетень.

Этот протокол беззащитен перед сговором ЦИК и ЦУР. Если они действуют вместе, они могут объединить свои базы данных и узнать, кто за кого голосует.

Голосование с одной Центральной комиссией

Чтобы избежать опасности сговора между ЦУР и ЦИК можно использовать более сложный протокол [1373]. Этот протокол идентичен предыдущему с двумя изменениями:

— ЦУР и ЦИК являются единой организацией, и

— для анонимного распределения регистрационных номеров на этапе (2) используется ANDOS (см. раздел 4.13).

Так как протокол анонимного распределения ключей не позволяет ЦИК узнать, у какого избирателя какой регистрационный номер, У ЦИК нет способа связать регистрационные номера и полученные бюллетени. Но ЦИК должна быть надежным органом, чтобы не выдавать регистрационных номеров неправомочным избирателям. Эту проблему также можно решить с помощью слепых подписей.

Улучшенное голосование с одной Центральной комиссией

В этом протоколе также используется ANDOS [1175]. Он удовлетворяет всем шести требованиям хорошего протокола голосования. Он не удовлетворяет седьмому требованию, но обладает двумя свойствами, дополняющими перечисленные в начале раздела шесть свойств:

7. Избиратель может изменить свое мнение (т.е., аннулировать свой бюллетень и проголосовать заново) в течение заданного периода времени.

8. Если избиратель обнаруживает, что его бюллетень посчитан неправильно, он может установить и и справить проблему, не рискуя безопасностью своего бюллетеня.

Вот этот протокол:

(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*. If 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 соответствующий значений $E_k(I, v)$.

(9) Если избиратель обнаруживает, что его бюллетень подсчитан неправильно, он протестует, посылая ЦИК :

$I, E_k(I, v), d$

(10) Если избиратель хочет изменить свой бюллетень с *v* на *v′*, он посылает ЦИК:

$I, E_k(I, v'), d$

Другой протокол голосования использует вместо ANDOS слепые подписи, но по сути мало чем отличается [585]. Этапы (1) - (3) являются предварительными. Их цель состоит в том, чтобы узнать и опубликовать всех действительных избирателей. Хотя некоторые из них, вероятно, не примут участи в голосовании, это уменьшает возможность ЦИК добавить поддельные бюллетени .

На этапе (4) два избирателя могут получить один и тот же идентификационный номер . Эта возможность может быть минимизирована, если число возможных идентификационных номеров будет гораздо больше, чем число реальных избирателей. Если два избирателя присылают бюллетени с одинаковым идентификатором, ЦИК генерирует новый идентификационный номер, *I′*, выбирает одного из избирателей и публикует :

$I', E_k(I, v)$

Владелец этого бюллетеня узнает о произошедшей путанице и посылает свой бюллетень снова, повторяя этап (5) с новым идентификационным номером .

Этап (6) дает каждому избирателю возможность проверить, что ЦИК правильно получила его бюллетень . Если его бюллетень неправильно подсчитан, он может доказать это на этапе (9). Предполагая, что бюллетень избирателя на этапе (6) правилен, сообщение, которое он посылает на этапе (9) доказывает, что его бюллетень был неправильно подсчитан.

Одной из проблем этого протокола является то, что жульническая ЦИК сможет воспользоваться людей, к оторые сообщили о намерении голосовать на этапе (2), но не голосовали в действительности . Другой проблемой является сложность протокола ANDOS. Авторы рекомендуют разбивать избирателей на меньшие группы, н апример избирательные округа .

Еще одной, более серьезной проблемой является то, что ЦИК может не подсчитать какой-нибудь бюллетень . Эта проблема неразрешима: Алиса утверждает, что ЦИК намеренно пренебрег ее бюллетенем, а ЦИК утве рждает, что Алиса никогда не голосовала .

Голосование без Центральной избирательной комиссии

В следующем протоколе ЦИК не используется, избиратели следят друг за другом . Этот протокол, созданный Майклом Мерриттом [452, 1076, 453], настолько громоздок, что возможность его реализации больше чем для пяти человек сомнительна, но все же познакомиться с ним будет полезно .

Алиса, Боб, Кэрол и Дэйв голосуют (да/нет или 0/1) по какому-то вопросу . Пусть у каждого избирателя есть открытый и закрытый ключи. Пусть также все открытые ключи известны всем .

(1) Каждый голосующий решает, как голосовать, и делает следующее:

(a) Добавляет случайную строку к своему бюллетеню.

(b) Шифрует результат этапа (a) открытым ключом Дэйва.

(c) Шифрует результат этапа (b) открытым ключом Кэрол.

(d) Шифрует результат этапа (c) открытым ключом Боба.

(e) Шифрует результат этапа (d) открытым ключом Алисы.

(f) Добавляет новую случайную строку к результату этапа (e) и шифрует получившееся открытым ключом Дэйва. Он записывает значение случайной строки.

(g) Добавляет новую случайную строку к результату этапа (f) и шифрует получившееся открытым ключом Кэрол. Он записывает значение случайной строки.

(h) Добавляет новую случайную строку к результату этапа (g) и шифрует получившееся открытым ключом Боба. Он записывает значение случайной строки.

(i) Добавляет новую случайную строку к результату этапа (g) и шифрует получившееся открытым ключом Алисы. Он записывает значение случайной строки.

Если E - это функция шифрования, R_i - случайная строка, а V - бюллетень , то его сообщение будет выглядеть следующим образом:

$E_A(R_5,E_B(R_4,E_C(R_3,E_D(R_2,E_A(E_B(E_C(E_D(V,R_1)))))))$

Каждый голосующий сохраняет промежуточные результаты на каждом этапе вычислений. Эти результаты будут использоваться на дальнейших этапах протокола для подтверждения, что бюллетень данного избирателя будет учтен.

(2) Каждый голосующий отправляет сообщение Алисе.

(3) Алиса расшифровывает все бюллетени, используя свой закрытый ключ, и удаляет все случайные строки на этом уровне.

(4) Алиса перетасовывает все бюллетени и посылает результат Бобу.

Теперь каждый бюллетень будет выглядеть следующим образом:

$E_B(R_4,E_C(R_3,E_D(R_2,E_A(E_B(E_C(E_D(V,R_1)))))))$

(5) Боб расшифровывает все бюллетени, используя свой закрытый ключ, проверяет, есть ли его бюллетень среди присланных бюллетеней, удаляет все случайные строки на этом уровне, тасует бюллетени и посылает результат Кэрол.

Теперь каждый бюллетень будет выглядеть следующим образом:

$E_C(R_3,E_D(R_2,E_A(E_B(E_C(E_D(V,R_1))))))$

(6) Кэрол расшифровывает все бюллетени, используя свой закрытый ключ, проверяет, есть ли ее бюллетень среди присланных бюллетеней, удаляет все случайные строки на этом уровне, тасует бюллетени и посылает результат Дэйву.

Теперь каждый бюллетень будет выглядеть следующим образом:

$E_D(R_2,E_A(E_B(E_C(E_D(V,R_1)))))$

(7) Дэйв расшифровывает все бюллетени, используя свой закрытый ключ, проверяет, есть ли его бюллетень среди присланных бюллетеней, удаляет все случайные строки на этом уровне, тасует бюллетени и посылает результат Алисе.

Теперь каждый бюллетень будет выглядеть следующим образом:

$E_A(E_B(E_C(E_D(V,R_1))))$

(8) Алиса расшифровывает все бюллетени, используя свой закрытый ключ, проверяет, есть ли ее бюллетень среди присланных бюллетеней, подписывает все бюллетени и посылает результат Бобу, Кэрол и Дэйву.

Теперь каждый бюллетень будет выглядеть следующим образом:

$S_A\ (E_B(E_C(E_D(V,R_1))))$

(9) Боб проверяет и удаляет подписи Алисы. Он расшифровывает все бюллетени, используя свой закрытый ключ, проверяет, есть ли его бюллетень среди присланных бюллетеней, подписывает все бюллетени и посылает результат Алисе, Кэрол и Дэйву.

Теперь каждый бюллетень будет выглядеть следующим образом:

$S_B(E_C(E_D(V,R_1)))$

(10) Кэрол проверяет и удаляет подписи Боба. Она расшифровывает все бюллетени, используя свой закрытый

ключ, проверяет, есть ли ее бюллетень среди присланных бюллетеней, подписывает все бюллетени и посылает результат Алисе, Бобу и Дэйву.

Теперь каждый бюллетень будет выглядеть следующим образом:

$S_C(E_D(V,R_1))$

(11) Дэйв проверяет и удаляет подписи Кэрол. Он расшифровывает все бюллетени, используя свой закрытый ключ, проверяет, есть ли его бюллетень среди присланных бюллетеней, подписывает все бюллетени и посылает результат Алисе, Бобу и Кэрол.

Теперь каждый бюллетень будет выглядеть следующим образом:

$S_D(V,R_1)$

(12) Все проверяют и удаляют подпись Дэйва. Они убеждаются, что их бюллетени находятся среди полученных (находя свою случайную строку).

(13) Все удаляют случайные строки из каждого бюллетеня и суммирует бюллетени.

Этот протокол не только работает, он сам является своим арбитром. Алиса, Боб, Кэрол и Дэйв немедленно узнают, если кто-нибудь из них попытается мошенничать. Не нужно никаких ЦИК и ЦУР. Чтобы увидеть, как это работает, попытаемся смошенничать.

Если кто-нибудь пытается добавить бюллетень, Алиса обнаружит эту попытку на этапе (3), когда она получит бюллетеней больше чем количество людей, участвующих в голосовании. Если Алиса попытается добавить бюллетень, Боб обнаружит это на этапе (4).

Более ловкой является подмена одного бюллетеня другим. Так как бюллетени шифруются различными открытыми ключами, каждый может создать столько правильных бюллетеней, сколько нужно. Протокол дешифрирования состоит из двух частей: первая включает этапы (3)-(7), а вторая - этапы (8)-(11). Подмена голоса на различных этапах обнаруживается по разному.

Если кто-нибудь заменит один бюллетень другим во второй части, его действия будут обнаружены неменленно. На каждом этапе бюллетени подписываются и посылаются всем избирателям. Если один избиратель (или несколько) обнаруживает, что его бюллетеня больше нет среди набора бюллетеней, он немедленно прекращает выполнение протокола. Так как бюллетени подписываются на каждом этапе, и так как каждый может вернуться во второй части протокола на несколько шагов назад, то обнаружить мошенника, подменившего бюллетени, легко.

Замена одного бюллетеня другим в первой части протокола более тонка. Алиса не может сделать замену на этапе (3), потому что Боб, Кэрол и Дэйв обнаружат это на этапах (5), (6) или (7). Боб может попробовать на этапе (5). Если он заменит бюллетени Кэрол и Дэйва (помните, он не знает, какой бюллетень чей), Кэрол или Дэйв заметят это на этапах (6) или (7). Они не будут знать, кто подменил их бюллетени (хотя это должен быть кто-то, уже обработавший бюллетени), но они будут знать, что их голоса подменены. Если Бобу повезло, и ему удалось подменить бюллетень Алисы, она не заметит этого до второй части протокола. Тогда она обнаружит исчезновение своего голоса на этапе (8), но не сможет узнать, кто подменил бюллетень. В первой части бюллетени перетасовываются на каждом этапе и не подписываются, поэтому никто не сможет отработать протокол обратно и определить, кто подменил бюллетени.

Другой формой мошенничества является попытка узнать, кто за кого проголосовал. Из-за перетасовки бюллетеней в первой части никто не сможет отработать протокол обратно и связать бюллетени и голосующих. Удаление случайных строк в первой части также является решающим для сохранения анонимности. Если строки не удаляются, перемешивание голосов может быть инвертировано при помощи повторного шифрования получаемых голосов открытым ключом того, кто их тасовал. Когда протокол остановится, конфиденциальность бюллетеней сохранится.

Более того, из-за начальной случайной строки, R_1, даже одинаковые бюллетени шифруются по разному на каждом этапе протокола. Никто не может узнать значение бюллетеня до этапа (11).

Каковы проблемы этого протокола? Во первых, для выполнения протокола нужны грандиозные вычисления. В приведенном примере в голосовании принимают участие только четверо, но и он *уже* сложен. Такой протокол не сможет работать при реальных выборах с десятками тысяч голосующих. Во вторых, Дэйв узнает результаты выборов раньше остальных. Хотя он и не может повлиять на результат, он получает определенное преимущество. С другой стороны такое также возможно и при централизованной схеме голосования.

Третья проблема заключается в том, что Алиса может скопировать бюллетень другого участника, даже не зная его содержания заранее. Чтобы понять, почему это может стать проблемой, рассмотрим выборы для трех голосующих - Алисы, Боба и Евы. Еве не важны результаты выборов, но она хочет знать, как голосовала Алиса. Поэтому она копирует бюллетень Алисы, и результат выборов будет соответствовать бюллетеню Алисы.

Другие схемы голосования

Было предложено много сложных безопасных протоколов выборов. Их можно разделить на два типа. Существуют протоколы с перемешиванием, как "Голосование без Центральной избирательной комиссии", в которых все бюллетени перемешиваются, чтобы никто не мог связать бюллетень и избирателя.

Также существуют протоколы с разделением, в которых личные бюллетени делятся между различными счетными комиссиями так, что ни одна из них не сможет обмануть избирателей [360, 359, 118, 115]. Эти протоколы Эти протоколы защищают анонимность избирателей только, если различные "части" правительства (или кто бы не проводил голосование) не сговариваются против избирателя. (Идея разбить центральный орган на несколько частей, которые пользуются доверием, только когда они действуют параллельно, пришла из [316].)

Один из протоколов с разделением предложен в [1371]. Основная идея состоит в том, что каждый избиратель делит свой бюллетень на несколько частей. Например, если бы бюллетень содержал "да" или "нет", 1 обозначала бы "да", а 0 - "нет", избиратель мог бы создать несколько чисел, которые в сумме давали бы 0 или 1. Эти доли посылаются счетным комиссиям, каждой по одной, и также шифруются и сохраняются. Каждый центр суммирует полученные доли (существуют протоколы, обеспечивающие правильность итога), и окончательный итог является суммой всех промежуточных итогов. Существуют также протоколы, гарантирующие, что доли каждого избирателя будут сложены для получения 0 или 1.

Другой протокол, предложенный Дэвидом Чаумом [322], позволяет проследить избирателя, который пытается мошенничать. Однако, выборы придется проводить повторно, исключив мешающего пользователя. Этот подход не применим на практике для выборов с большим числом избирателей.

Еще один, более сложный протокол, решающий некоторые из этих проблем можно найти в [770, 771]. Существует даже протокол, использующий шифры со многими ключами [219]. Другой протокол, который, как утверждается, подходит для крупномасштабных выборов, приведен в [585]. А [347] позволяет избирателям не голосовать.

Протоколы голосования работают, они даже облегчают продажу и покупку голосов. Когда покупатель может быть уверен, что продавец проголосует, как обещал, стимул купить голоса становится еще сильнее. Ряд протоколов были спроектированы **без подтверждения**, не позволяя избирателю доказать кому-либо еще, что он проголосовал определенным образом [117, 1170, 1372].

6.2 Безопасные вычисления с несколькими участниками

Безопасные вычисления с несколькими участниками представляют собой протокол, с помощью которого группа людей может определенным образом вычислить функцию многих переменных. Каждый в группе обеспечивает одну или несколько переменных. Результат вычислений становится известным каждому в группе, но никому не известны значения, предоставленные другими членами группы, если это не является очевидным из результата вычислений. Ниже приведено несколько примеров:

Протокол №l

Как может группа людей вычислить свою среднюю зарплату без того, чтобы зарплата одного стала известна другому?

(1) Алиса добавляет секретное случайное число к сумме своей зарплаты, шифрует результат открытым ключом Боба и посылает его Бобу.

(2) Боб расшифровывает результат своим закрытым ключом. Он добавляет сумму своей зарплаты к полученному от Алисы значению, шифрует результат открытым ключом Кэрол и посылает его Кэрол.

(3) Кэрол расшифровывает результат своим закрытым ключом. Она добавляет сумму своей зарплаты к полученному от Боба значению, шифрует результат открытым ключом Дэйва и посылает его Дэйву.

(4) Дэйв расшифровывает результат своим закрытым ключом. Он добавляет сумму своей зарплаты к полученному от Кэрол значению, шифрует результат открытым ключом Алисы и посылает его Алисе.

(5) Алиса расшифровывает результат своим закрытым ключом. Она вычитает случайное число, прибавленное на этапе (1), получая сумму всех зарплат.

(6) Алиса делит результат на число людей (в данном случае на четыре) и объявляет результат.

Этот протокол подразумевает, что каждый участник честен - они хотя и могут любопытствовать, но следуют протоколу. Если любой из участников солжет о своей зарплате, средняя зарплата будет рассчитана неправильно. Более серьезная проблема состоит в том, что Алиса может искажать итоговый результат. Она может вычесть на этапе (5) любое число, которое ее устраивает, и никто об этом не узнает. Помешать Алисе сделать это можно, потребовав от нее вручить ее случайное число с помощью одной из схем вручения бита из раздела 4.9, но когда

она откроет свое случайное число в конце протокола Боб сможет узнать ее зарплату.

Протокол №2

Алиса и Боб вместе в ресторане и спорят о том, кто старше. Никто, однако, не хочет сообщить другому свой возраст. Каждый из них мог бы прошептать свой возраст на ушко доверенной нейтральной стороне (например, официанту), кто мог бы сравнить числа в уме и объявить результат и Алисе, и Бобу.

У приведенного протокола есть две проблемы. Во первых, вычислительные способности среднего официант не позволяю ему обработать ситуации более сложный чем определение большего из двух чисел. И во вторых, если бы Алиса и Боб действительно заботились о сохранении своей информации в тайне, им пришлось бы ут опить официанта в ванне с минеральной водой, чтобы он ничего не разболтал буфетчику.

Криптография с открытыми ключами предлагает существенно менее жесткое решение. Существует прот окол, в соответствии с которым Алиса, зная значение a, и Боб, зная b, могут совместно определить верно ли, что $a<b$, так, чтобы Алиса не получила информации о b, а Боб - об a. Кроме того, и Алиса, и Боб убеждены в пр оверке правильности вычислений. Так как используемый криптографический алгоритм является существенной частью протокола, подробности можно найти в ра зделе 23.14.

Конечно, этот протокол не защитит от активных мошенников. Ничто не сможет помешать Алисе (или Бобу, какая разница) солгать о своем возрасте. Если бы Боб был компьютерной программой, которая слепо следовала бы протоколу, Алиса могла бы узнать его возраст (является ли возрастом компьютерной программы отрезок времени с момента ее написания или с момента ее запуска?), повторно выполняя протокол. Алиса могла бы ук азать, что ее возраст - 60 лет. Узнав, что она старше, она могла бы выполнить протокол снова, указав, что ее во зраст - 30 лет. Узнав, что Боб старше, она могла бы снова выполнить протокол, указав, что ее возраст - 45 лет, и так далее, пока Алиса не узнает возраст Боба с любой нужной ей степенью точности.

При условии, что участники не обманывают специально, этот протокол легко расширить для нескольких участников. Любое количество людей может определить порядок их возрастов с помощью последовательных честных применений прот окола, и никакой участник не сможет узнать возраст другого.

Протокол №3

Алиса нравится забавляться с плюшевыми медведями. В эротических фантазиях Боба важное место зан имают мраморные столы. Оба весьма стесняются своих привычек, но с удовольствием нашли бы кого-нибудь, кто разделил бы с ними их... гм... образ жи зни.

В Службе безопасных вычислений с несколькими участниками мы спроектировали протокол для подобных людей. Мы занумеровали впечатляющий список их пристрастий от "африканских муравьедов" до "яблочных пирогов". Разделенные модемной линией связи, Алиса и Боб могут участвовать в безопасном протоколе с н есколькими участниками. Они вместе могут определить, есть ли у них общие привычки. Если есть, они могли бы устремиться к обоюдному счастью. Если нет, то они могли бы безопасно расстаться, сохраняя уверенность, что их привычки остались в тайне. Никто, даже Служба безопасных вычислений с несколькими участниками, ник огда не узнает об их пристрастиях.

Вот как это работает:

(1) С помощью однонаправленной функции Алиса хэширует свою привычку как семизначную строку.

(2) Используя эту семизначную строку как телефонный номер, Алиса звонит по этому номеру и оставляет с ообщение Бобу. Если никто не отвечает, или номер не обслуживается. Алиса применяет однонаправленную функцию к телефонному номеру до тех пор, пока не найдется кто-нибудь, кто подхватит протокол.

(3) Алиса сообщает Бобу, сколько раз ей пришлось применять однонаправленную функцию к своей привы чке.

(4) Боб хэширует свою привычку столько же раз. Он также использует семизначную строку как телефонный номер и спрашивает человека на другом конце провода, нет ли для него сообщений.

Обратите внимание, что у Боба есть возможность вскрытия с использованием выбранного открытого текста. Он может хэшировать распространенные привычки и позвонить по получившемуся телефону, разыскивая соо бщения для него. Это протокол реально работает только такое вскрытие непрактично из-за достаточного числа возможных открытых текстов сообщений.

Также существует математический протокол, похожий на Протокол № 2. Алиса знает a, Боб знает b, и они вместе пытаются определить, верно ли, что $a = b$, причем так, чтобы Боб ничего не узнал об a, а Алиса - о b. Подробности можно найти в разделе 23.14.

Протокол №4

Вот другая проблема для безопасных вычислений со многими участниками [1373]: совет семи регулярно

встречается, чтобы тайно проголосовать по некоторым вопросам. (Все в порядке, они управляют миром - не говорите никому, что я вам проговорился.) Все члены совета могут голосовать "да" или "нет". Кроме того, две стороны обладают "супер-бюллетенями": 5-да и 5-нет. Они не обязаны использовать эти "супер-бюллетени" и, если хотят, могут воспользоваться обычными бюллетенями. Если никто не использует "супер-бюллетени", то вопрос решается простым большинством голосов. В случае применения одного или двух эквивалентных "супер-бюллетеней" все обычные голоса игнорируются. В случае двух противоречащих вопрос решается большинством обычных голосов. Нам нужен протокол, который надежно осуществляет такую форму голосования.

Следующие два примера иллюстрируют процесс голосования. Пусть участвуют пять обычных избирателей, от N_1 до N_5, и два суперизбирателя: S_1 и S_2. Вот голосование по вопросу №1:

S_1	S_2	N_1	N_2	N_3	N_4	N_5
Супер-да	нет	нет	нет	нет	да	да

В этом примере действует только один "супер-бюллетень" S_1, и результат голосования - "да". А вот голосование по вопросу №2:

S_1	S_2	N_1	N_2	N_3	N_4	N_5
Супер-да	Супер-нет	нет	нет	нет	да	да

В этом примере два "супер-бюллетеня" нейтрализуют друг друга, и вопрос решается большинством обычных "нет".

Если не требуется скрыть информацию о том, обычный или супербюллетель был решающим, то это простое применение безопасного протокола голосования. Сокрытие этой информации потребует более сложного безопасного протокола вычислений с несколькими участниками.

Этот вид голосования может произойти в реальной жизни. Это может быть часть организационной структуры корпорации, где некоторые люди обладают большей властью чем другие, или часть процедуры ООН, где одни государства имеют большее значение, чем другие.

Безусловные безопасные протоколы с несколькими участниками

Это только частный случай общей теоремы: любая функция с *n* входами может быть вычислена *n* игроками способом, который позволит всем узнать значение функции, но любое количество игроков, меньшее, чем *n/2*, не сможет получить никакой дополнительной информации, не следующей из их собственных входов и результата вычислений. Подробности можно найти в [136, 334, 1288, 621].

Безопасная оценка схемы

Вход Алисы - *a*, а Боба - *b*. Они вместе хотят вычислить некоторую функцию *f(a,b)* так, чтобы Алиса не смогла ничего узнать о входе Боба, а Боб - о входе Алисы. Главная проблема безопасных вычислений с несколькими участниками также называется **безопасной оценкой схемы**. Алиса и Боб могут создать произвольную логическую схему. Эта схема получает на вход значения Алисы и Боба и выдает результат. Безопасная оценка схемы является протоколом, который реализует следующие три требования:

1. Алиса может ввести свое значение так, что Боб не сможет его узнать.
2. Боб может ввести свое значение так, что Алиса не сможет его узнать.
3. И Алиса, и Боб могут вычислить результат, причем обе стороны будут убеждены в том, что результат правилен и не подтасован ни одной стороной.

Детали протокола безопасной оценки схемы можно найти в [831].

6.3 Анонимная широковещательная передача сообщений

Вам не удастся пообедать с компанией криптографов и не оказаться среди ожесточенной перепалки. В [321] Дэвид Чаум вводит Проблему обедающих криптографов:

> Три криптографа сидят за обедом в своем любимом трехзвездочном ресторане. Их официант сообщает им, что метрдотель принял необходимые меры, чтобы счет можно было бы оплатить анонимно. За обед мог бы заплатить один из криптографов или NSA. Три криптографа очень уважают право каждого из них заплатить анонимно, но им хотелось бы знать, заплатит ли NSA.

Как криптографам Алисе, Бобу и Кэрол узнать, не заплатил ли за обед кто-нибудь из них, и в то же время не

нарушить анонимность плательщика? Чаум решает проблему:

> Каждый криптограф бросает несмещенную монету, прикрывшись своим меню, между ним и криптографом справа от н е-го так, что только они двое могут видеть результат. Затем каждый криптограф громко объявляет, упаали ли две монеты - одна его и одна его левого соседа - на одну или на различные стороны. Если плательщик - один из криптографов, то его утвержд ение противоположно тому, что он видит. Нечетное число заявленных различий за столом указывает, что обед оплачивает криптограф; четное число различий - что NSA (при условии, что обед может быть оплачен только один раз). Однако, если обед оплачивает криптограф, никто из двух других не узнает из сделанных заявлений, кто же конкретно опл атил обед.

Чтобы увидеть, как это работает, вообразите, что Алиса пытается понять, кто из двух других криптографов заплатил за обед (при условии, что не она и не NSA). Если она видит две различных монеты, то либо оба других криптографа (Боб и Кэрол) сказали, "одинаковые" или оба сказали, "разные". (Помните, нечетное число кри п-тографов, говорящих "разные" указывает, что оплатил кто-то из них.). Если оба сказали "разные", то плател ь-щик - криптограф, сидящий ближе всего к монете, результат броска которой тот же, что и у скрытой монеты (брошенной между Бобом и Кэрол). Если оба сказали "одинаковые", то плательщик - криптограф, сидящий ближе всего к монете, результат броска которой отличается от результата броска скрытой монеты. Однако, если Алиса видит две одинаковых монеты, то или Боб сказал, "одинаковые", а Кэрол - "разные", или Боб сказал "разные", а Кэрол - "одинаковые". Если ли скрытая монета - такая же как и видимые ей две монеты, то пл а-тельщик - криптограф, который сказал, "разные". Если скрытая монета отлична от видимых ей двух монет, то плательщик - криптограф, который сказал "одинаковые". Чтобы определить, кто платил, во всех этих случаях Алиса должна знать результат броска монеты между Бобом и Кэрол.

Этот протокол может быть обобщен на любое количество криптографов, которые сидят по кругу и бросают монеты между собой. Каждая пара криптографов выполняет протокол. Конечно, они знают, кто платит, но кто-то, наблюдающий за протоколом может сказать только, что заплатил один из криптографов или NSA, но не сожет указать, какой из криптографов платил.

Применение этого протокола выходит далеко за пределы обеденного стола. Вот пример **безусловного отправителя и неотслеживаемого отправителя**. Группа пользователей сети может использовать этот протокол для отправления анонимных сообщений.

(1) Пользователи упорядочиваются по кругу.

(2) Через регулярные интервалы времени соседние пары пользователей бросают между собой монету, испол ь-зуя какой-нибудь безопасный от злоумышленников протокол бросания "честной" монеты.

(3) После каждого броска каждый пользователь объявляет либо "одинаковые", либо "разные".

Если Алиса хочет передать широковещательное сообщение, она просто начинает инвертировать свое утве р-ждение в тех раундах, которые соответствуют 1 в двоичном представлении ее сообщения. Например, если ее сообщение было "1001", она инвертирует ее утверждение, сообщит правду, сообщит правду, и затем инвертир у-ет снова утверждение. При условии, что результатом ее бросков было были "разные", "одинаковые", "одинаковые", "одинаковые", она будет говорить "одинаковые", "одинаковые ", "одинаковые", "разные".

Если Алиса замечает, что полный результат протокола не соответствует сообщению, которое она пробует п осылать, она понимает, что в это же время кто-то еще пытается посылать сообщение. Тогда она прекращает п е-редачу сообщения и выжидает случайное количество раундов перед очередной попыткой. Точные параметры должны быть выработаны на основе трафика сообщений в сети, но идея достаточно понятна.

Чтобы сделать дело еще более интересным, эти сообщения могут быть зашифрованы открытым ключом др у-гого пользователя. Затем, когда каждый принимает сообщение (практическая реализация должна включать стандартные заголовки и окончания сообщений), только определенный получатель сможет расшифровать и прочесть сообщение. Никто другой ничего не узнает про автора сообщения и не сможет определить получателя сообщения. Даже если удастся расшифровать сами сообщения, то анализ трафика, отслеживающий и собира ю-щий формы межпользовательского обмена, бесполе зен.

Альтернативой бросанию монет между соседними сторонами могло бы быть использование файла случа й-ных битов. Возможно, стороны могли бы хранить файл на CD-ROM, или один член пары мог бы генерировать пачку битов и посылать их другой стороне (конечно, в зашифрованном виде). Или, они могли бы договориться использовать совместно криптографически безопасный генератор псевдослучайных чисел, и каждый из них смог бы генерировать для протокола ту же самую последовательность псевдослучайных битов.

Проблемой этого протокола является то, что хотя мошенничающий участник и не сможет читать никаких с о-общений, он может незаметно испортить всю систему, постоянно обманывая на этапе (3). Существует модификация предыдущего протокола, позволяющая обнаружить вредительство [1578, 1242]. Эта проблема называется "Обедающие криптографы в дискотеке".

6.4 Электронные наличные

Наличные деньги - это проблема. Раздражает их носить, они способствуют распространению микробов, л юди могут красть их у Вас. Чеки и кредитные карточки уменьшили количество наличных денег, оборачивающи хся в обществе, но полное удаление наличных денег фактически невозможно. Этого никогда не произойдет; то рговцы наркотиками и политические деятели никогда этого не допустят. Чеки и кредитные карточки можно пр оследить, вы не можете скрыться от того, кому дали деньги.

С другой стороны, чеки и кредитные карточки позволяют людям вторгаться в вашу личную жизнь как ник огда прежде. Вы никогда не допустили бы, чтобы полиция всю жизнь ходила за вами по пятам, но полицейские могут проследить ваши финансовые операции. Они могут видеть, где вы покупаете газ, где вы покупаете еду, кому вы звоните по телефону, и все это не отрываясь от своих компьютерных терминалов. Люди должны уметь защитить свою анонимность, чтобы защитить свои личные тайны.

К счастью, существует сложный протокол, который разрешает использование заверенных, но неотслежива емых сообщений. Лоббист Алиса может передать **электронные деньги** конгрессмену Бобу так, чтобы газетный репортер Ева ничего не узнала бы об Алисе. Боб может затем вносить эти электронные деньги на свой банко вский счет, даже если банк не имеет об Алисе никакого представления. Но если Алиса пробует покупать кокаин на ту же самую порцию электронных денег, которую она использовала для подкупа Боба, она будет обнаружена банком. И если Боб пробует вносить порцию электронных денег на два различных счета, это будет обнаружено, но Боб, как и Алиса, останется он анонимным. Иногда это называется **анонимными электронными деньгами**, чтобы можно было отличить их от отслеживаемых электронных денег, типа креди тных карточек.

В подобных вещах существует большая общественная необходимость. С ростом использования Internet для коммерческих операций растет и потребность в секретности передаваемой по сети информации и анонимности при ведении дел. (Имеется немало причин для того, чтобы люди отказывались посылать номер их кредитной карточки по Internet.) С другой стороны, банки и правительства, по видимому, не пожелают уступить контроль над современными банковскими системами. Хотя им придется это сделать. Все, что потребуется, чтобы эле ктронные деньги вошли в моду, - это появление некоторого заслуживающего доверия учреждения, желающего преобразовывать цифры в реальные деньги.

Протоколы электронных денег очень сложны. Дальше мы шаг за шагом построим один из них. Более подробно об этом протоколе можно прочитать в [318, 339, 325, 335, 340]. Но помните, это только один из протоколов электронных денег, существуют и другие.

Протокол №1

Первые несколько протоколов представляют собой физические аналоги криптографических протоколов. Следующий протокол является упрощенным физическим протоколом для анонимных денежных чеков:

(1) Алиса готовит 100 анонимных денежных чеков по $1000 каждый.

(2) Алиса вкладывает каждый из них и листок копировальной бумаги в 100 различных конвертов и относит все конверты в банк.

(3) Банк открывает 99 конвертов и убеждается, что каждый чек выписан на $1000.

(4) Банк подписывает единственный оставшийся нераспечатанным конверт. С помощью копировальной бумаги подпись переводится на чек. Банк возвращает нераспечатанный конверт Алисе и списывает $1000 с ее счета.

(5) Алиса вскрывает конверт и отдает денежный чек продавцу.

(6) Продавец проверяет банковскую подпись, убеждаясь в законности денежного чека.

(7) Продавец относит денежный чек в банк.

(8) Банк проверяет свою подпись и начисляет $1000 на счет продавца.

Этот протокол работает. Банк не видит денежный чек, который он подписывает, поэтому, когда продавец принесет чек в банк, банк никогда не узнает, что это чек Алисы. Благодаря подписи банк убежден в законности чека. А из-за протокола "разрезать и выбрать" (см. раздел 5.1) банк уверен, что нераспечатанный денежный чек - на сумму $1000 (а не $100000 или $100000000). Он проверяет остальные 99 конвертов, поэтому вероятность обмана банка Алисой составляет только 1 процент. Конечно, банк назначит за обман достаточно большой штраф, такой, чтобы не стоило мошенничать. Ведь если банк просто откажется подписать последний чек (если Алиса поймана на обмане), не штрафуя Алису, она продолжит свои попытки, пока ей не повезет. Лучшее сре дство устрашения - это тюремное заключение.

Протокол №2

Предыдущий протокол не дает Алисе написать чек на сумму, отличную от заявленной, но он не мешает ей откс ерокопировать чек и использовать его дважды. Это называется **проблемой повторной оплаты**; для ее решение придется усложнить протокол:

(1) Алиса готовит 100 анонимных денежных чеков по $1000 каждый. К каждому денежному чеку она добавляет уникальную строку, выбранную случайным образом и достаточно длинную, чтобы вероятность использования этой строки другим человеком была пренебрежимо мала.

(2) Алиса вкладывает каждый из них и листок копировальной бумаги в 100 различных конвертов и относит все конверты в банк.

(3) Банк открывает 99 конвертов и убеждается, что каждый чек выписан на $1000.

(4) Банк подписывает единственный оставшийся нераспечатанным конверт. С помощью копировальной бумаги подпись переводится на чек. Банк возвращает нераспечатанный конверт Алисе и списывает $1000 с ее счета.

(5) Алиса вскрывает конверт и отдает денежный чек продавцу.

(6) Продавец проверяет банковскую подпись, убеждаясь в законности денежного чека.

(7) Продавец относит денежный чек в банк.

(8) Банк проверяет свою подпись и по своей базе данных убеждается, что денежный чек с такой уникальной строкой ранее не депонировался. Если это так, банк начисляет $1000 на счет продавца и записывает уникальную строку в базу данных.

(9) Если денежный чек уже был депонирован ранее, банк отказывается принять его.

Теперь, если Алиса попытается расплатиться ксерокопией денежного чека или продавец попытается депонировать денежный чек повторно, используя ксерокопию, банк узнает об этом.

Протокол №3

Предыдущий протокол защищает банк от мошенников, но не устанавливает их личность. Банк не знает, попытался ли человек, который получил чек (банк ничего не знает об Алисе), обмануть продавца, или продавец пытается обмануть банк. Эта неоднозначность исправляется следующим протоколом:

(1) Алиса готовит 100 анонимных денежных чеков по $1000 каждый. К каждому денежному чеку она добавляет уникальную строку, выбранную случайным образом и достаточно длинную, чтобы вероятность использования этой строки другим человеком была пренебрежимо мала.

(2) Алиса вкладывает каждый из них и листок копировальной бумаги в 100 различных конвертов и относит все конверты в банк.

(3) Банк открывает 99 конвертов и убеждается, что каждый чек выписан на $1000, и что все случайные строки различны.

(4) Банк подписывает единственный оставшийся нераспечатанным конверт. С помощью копировальной бумаги подпись переводится на чек. Банк возвращает нераспечатанный конверт Алисе и списывает $1000 с ее счета.

(5) Алиса вскрывает конверт и отдает денежный чек продавцу.

(6) Продавец проверяет банковскую подпись, убеждаясь в законности денежного чека.

(7) Продавец просит Алису написать случайную идентификационную строку на денежном чеке.

(8) Алиса выполняет это.

(9) Продавец относит денежный чек в банк.

(10) Банк проверяет свою подпись и по своей базе данных убеждается, что денежный чек с такой уникальной строкой ранее не депонировался. Если это так, банк начисляет $1000 на счет продавца и записывает уникальную строку в базу данных.

(11) Если уникальная строка уже есть в базе данных, банк отказывается принять денежный чек и сравнивает идентификационную строку на денежном чеке с хранимой в базе данных. Если они совпадают, то банк убеждается, что копия была снята с чека продавцом. Если идентификационные строки различны, то банк знает, что чек был скопирован человеком, который его готовил.

В этом протоколе предполагается, что продавец не может изменить идентификационную строку после того,

как Алиса напишет ее на денежном чеке. На денежном чеке мог бы находиться ряд небольших квадратов, которые по требованию торговца Алиса должна заполнить крестиками или ноликами. Денежный чек мог бы быть сделан из бумаги, которая рвется при исправлениях.

Так как продавец и банк взаимодействуют после того, как Алиса потратит деньги, продавцу могут всучить плохой денежный чек. Практические реализации этого протокола могли бы потребовать от Алисы подождать у кассового аппарата, пока продавец будет разбираться с банком, точно также, как это происходит сегодня при обработке платежей с использованием кредитных карточек.

Алиса также может приспособиться и к этому. Она может потратить копию денежного чека второй раз, написав ту же самую идентификационную строку на этапе (7). Если продавец не ведет базу данных уже полученных денежных чеков, он будет введен в заблуждение. Эту проблему устраняет следующий протокол.

Протокол №4

Если окажется, что человек, выписавший банковский чек, попытался обмануть продавца, то банк может захотеть личность этого человека. Чтобы сделать это, придется вернуться от физических аналогий в мир криптографии.

Чтобы спрятать имя Алисы в электронном чеке, можно воспользоваться методикой разделения секрета .

(1) Алиса готовит *n* анонимных денежных чеков на заданную сумму.

Каждый из чеков содержит уникальную строку, *X*, полученную случайным образом и достаточно длинную, чтобы вероятность появления двух одинаковых строк была пренебрежимо мала .

На каждом чеке есть также *n* пар битовых строк идентификации, I_1, I_2, ..., I_n. (Именно так, *n* различных пар на *каждом* чеке.) Каждая из этих пар генерируется следующим образом: Алиса создает строку, содержащую ее имя, адрес и прочие сведения, требуемые банком. Затем она делит эту строку на две части, используя протокол деления секрета (см. раздел 3.6) и вручает каждую часть, используя протокол вручения битов.

Например, I_{37} состоит из двух частей: I_{37_L} и I_{37_R}. Каждая часть представляет собой пакет врученных битов, который Алису могут попросить открыть, и чье открытое содержание может быть мгновенно проверено. Любая пара (например, I_{37_L} и I_{37_R}, но не I_{37_L} и I_{38_R}), раскрывает личность Алисы. Каждый из чеков выглядит следующим образом:

Сумма

Уникальная строка: *X*

Строки идентификации: $I_1 = (I_{1_L}, I_{1_R})$

$I_2 = (I_{2_L}, I_{2_R})$

....

$I_n = (I_{n_L}, I_{n_R})$

(2) Алиса маскирует все *n* чеков с помощью протокола слепой подписи и относит чеки в банк.

(3) Банк просит Алису снять маскировку с *n-1* денежных чеков и убеждается, что все они правильно оформлены. Банк проверяет сумму, уникальную строку и просит Алису раскрыть все строки идентификации.

(4) Если банк удовлетворен, не обнаружив попыток мошенничества, он подписывает оставшийся замаскированный денежный чек. Банк возвращает замаскированный чек Алисе и списывает сумму с ее счета.

(5) Алиса снимает маскировку с чека и тратит его у продавца.

(6) Продавец проверяет банковскую подпись, убеждаясь в законности денежного чека .

(7) Продавец случайным образом просит Алису раскрыть либо левые, либо правые половины всех строк идентификации на чеке. По сути, продавец выдает Алисе случайную *n*-битовую **строку-селектор**, b_1, b_2, ..., b_n. Левую или правую половину I_i откроет Алиса, зависит от значения b_i, 0 или 1.

(8) Алиса выполняет это.

(9) Продавец относит денежный чек в банк.

(10) Банк проверяет свою подпись и по своей базе данных убеждается, что денежный чек с такой уникальной строкой ранее не депонировался. Если это так, банк начисляет указанную сумму на счет продавца и запи-

сывает уникальную строку в базу данных.

(11) Если уникальная строка уже есть в базе данных, банк отказывается принять денежный чек и сравнивает идентификационную строку на денежном чеке с хранимой в базе данных. Если они совпадают, то банк убеждается, что чек был скопирован продавцом. Если идентификационные строки различны, то банк знает, что чек был скопирован человеком, который готовил этот денежный чек. Так как второй продавец, получивший чек, выдал Алисе другую, чем первый, строку-селектор, банк обнаружит, что для какой-то из позиций Алиса открыла левую половину одному продавцу, а правую - другому. Выполнив над этими половинами строки идентификации операцию XOR, банк определит личность Алисы.

Это весьма интересный протокол, поэтому посмотрим на него с разных сторон.

Может ли Алиса смошенничать? Ее электронные деньги представляют собой просто строку битов, которую она легко может скопировать. Потратить их в первый раз - не проблема, она просто выполнит протокол, и все пройдет без проблем. Продавец выдаст ей на этапе (7) случайную n-битовую строку-селектор, и Алиса откроет либо левую, либо правую половину каждой I_i на этапе (8). На этапе (10) банк запишет все эти данные вместе с уникальной строкой денежного чека.

Когда она попытается использовать те же электронные деньги второй раз, продавец (тот же или иной) выдаст ей на этапе (7) другую случайную n-битовую строку-селектор. Алиса должна выполнить этап (8), ее отказ немедленно встревожит продавца. Теперь, когда продавец приносит деньги в банк на этапе (10), банк немедленно заметит, что денежный чек с этой уникальной строкой уже был депонирован. Банк сравнивает открытые половины строк идентификации. Вероятность совпадения двух случайных строк-селекторов составляет один шанс из 2^n, этого не случится до следующего оледенения. Теперь банк находит пару, первая половина которой была открыта в первый раз, а вторая - во второй, выполняет над этими половинами операцию XOR и извлекает имя Алисы. Так банк узнает, кто попытался воспользоваться чеком дважды.

Что этот протокол не мешает Алисе мошенничать, но ее мошенничество почти наверняка будет обнаружено. Смошенничав, Алиса не сможет сохранить в тайне свою личность. Она не может изменить ни уникальную строку, ни какую-нибудь из строк идентификации, иначе испортится банковская подпись, и продавец немедленно заметит это на этапе (6).

Алиса могла бы попытаться подсунуть банку плохой денежный чек, такой, на котором строки идентификации не раскрывают ее имени, или, еще лучше, раскрывают имя кого-то еще. Вероятность, что такая уловка проскочит мимо банка на этапе (3), составляет 1 из n. Это не невозможно, но если штраф за мошенничество достаточно суров, Алиса не будет испытывать судьбу. Или вы можете увеличить число избыточных чеков, предъявляемых Алисой на этапе (1).

Может ли смошенничать продавец? Его шансы даже хуже. Он не может депонировать денежный чек два жды, банк заметит повторное использование строки-селектора. Он не сможет мошенничать, обвиняя Алису, так как только она может открыть любую строку идентификации.

Не поможет обмануть банк и любой сговор между Алисой и продавцом. Если банк подписал денежный чек с уникальной строкой, он может быть уверен в том, что этот чек будет оплачен только один раз.

А как насчет банка? Может ли он вычислить, что денежный чек, полученный от продавца, это и есть тот самый чек, который был подписан для Алисы? На этапах (2)-(5) Алиса защищена протоколом слепой подписи. Банк не сможет связать Алису и чек, даже если он полностью сохраняет запись каждой транзакции. Более того, даже объединившись, банк и продавец не смогут установить личность Алисы. Алиса может пройтись по магазину и, оставаясь полностью анонимной, купить то, что ей надо.

Может смошенничать Ева. Если она сможет подслушать линию связи между Алисой и продавцом, и если она сможет добраться до банка раньше продавца, она сможет первой депонировать чек. Банк примет его и, что хуже, когда продавец попытается депонировать свой чек, то он будет обвинен в мошенничестве. Если Ева украдет электронные деньги Алисы и успеет потратить их прежде Алисы, то в мошенничестве будет обвинена Алиса. Не существует способа помешать этому, и это является прямым следствием анонимности наличных. И Алиса, и продавец должны защищать свои биты так, как они защищали бы свои деньги.

Место этого протокола где-то между протоколом с посредником и самодостаточным протоколом. И Алиса, и продавец доверяют банку в том, что касается денег, но Алиса не должна доверять банку сведения о своих покупках.

Электронные наличные и идеальное приведение

У электронных наличных есть и своя темная сторона. Иногда людям не нужно так много секретности. Смотрите, как Алиса совершает идеальное преступление [1575]:

(1) Алиса крадет ребенка.

(2) Алиса готовит 10000 анонимных денежных чеков по $1000 (или другое количество чеков нужного ей достоинства).

(3) Алиса маскирует все 10000 денежных чеков, используя протокол слепой подписи. Она посылает их властям с угрозой убить ребенка, если не будут выполнены следующие инструкции :

(a) Все 10000 денежных чеков должны быть подписаны банком.

(b) Результаты должны быть опубликованы в газете.

(4) Власти соглашаются.

(5) Алиса покупает газету, снимает маскировку с денежных чеков и начинает тратить их . Не смогут найти ее, проследив за денежными чеками.

(6) Алиса освобождает ребенка.

Заметьте, что эта ситуация гораздо хуже чем при использовании любых физических носителей, например, наличных. Без физического контакта у полиции гораздо меньше шансов задержать похитителя .

Однако, в общем случае электронные наличные не слишком удобны для преступников . Проблема в том, что анонимность работает только для одной стороны - покупатель анонимен, а продавец нет . Более того, продавец не сможет скрыть факт получения денег. Электронные наличные помогут правительству определить, сколько денег вы зарабатываете, но определить, как вы их тратите, останется невозможным .

Реальные электронные наличные

Голландская компания, DigiCash, владеет большей частью патентов в области электронных наличных и реализовала протоколы электронных наличных в работающих продуктах owns. Если вы заинтересовались этим, обратитесь в DigiCash BV, Kruislaan 419, 1098 VA Amsterdam, Nethe rlands.

Другие протоколы электронных наличных

Существуют и другие протоколы электронных наличных, см. [707, 1554, 734, 1633, 973]. Ряд из них использует весьма изощренную математику. Различные протоколы электронных наличных можно разделить на различные категории. **Диалоговые** системы требуют, чтобы продавец связывался с банком при каждой продаже, что очень похоже на сегодняшний протокол для кредитных карточек . Если возникает какая-нибудь проблема, банк не принимает наличные, и Алиса не может смошенничать.

Автономные системы, подобные протоколу №4, не требуют соединения между продавцом и банком до окончания транзакции между продавцом и покупателем. Эти системы не помешают Алисе мошенничать, но вместо этого обнаружат ее мошенничество. Протокол №4 обнаруживает мошенничество Алисы, раскрывая ее личность при попытке мошенничать. Алиса знает о последствиях и, поэтому, не мошенничает .

Другой путь состоит в создании специальной интеллектуальной карты (см. Раздел 24.13), содержащей защищенную микросхему, называемую **наблюдателем** [332, 341, 387]. Микросхема-наблюдатель хранит мини-базу данных всех частей электронных наличных, потраченных этой интеллектуальной платой. Если Алиса попытается скопировать какие-то электронные наличные и потратить их дважды, внедренная микросхема-наблюдатель обнаружила бы такую попытку и не разрешила транзакцию. Так как микросхема-наблюдатель защищена от вмешательства извне, Алиса не сможет стереть мини-базу данных без разрушения интеллектуальной карты. Наличные деньги могут оборачиваться в экономике, когда они, наконец будут депонированы, банк может проверить наличные и определить мошенника, если произошел обман.

Протоколы электронных наличных можно разделить и по другому признаку . Номинал **электронных монет** фиксирован, людям, использующим такую систему, нужен ряд монет различных номиналов . **Электронные чеки** могут быть использованы для любых сумм до заданного максимума, а непотраченный остаток может быть возвращен на счет.

Двумя отличными совершенно отличающимися друг от друга автономными протоколами электронных монет являются [225, 226, 227] и [563, 564, 565]. Также можно предложить система NetCash (Сетевые наличные) с более слабыми свойствами [1048, 1049]. Другой новой системой является [289].

В [1211] Тацуаки Окамото (Tatsuaki Okamoto) и Казуо Охта (Kazuo Ohta) перечислили шесть свойств идеальной системы электронных наличных:

1. Независимость. Безопасность электронных наличных не зависит от местонахождения . Наличные могут быть переданы по компьютерным сетям.
2. Безопасность. Электронные наличные нельзя скопировать и повторно использовать .
3. Тайна личности (Неотслеживаемость). Тайна личности пользователя защищена, связь между пользо-

вателем и его покупками обнаружить невозможно.

4. Автономный платеж. Когда пользователь расплачивается за покупку электронными наличными, протокол между пользователем и продавцом выполняется автономно. То есть, магазину не нужно соединяться с центральным компьютером для обработки платежа пользователя.
5. Перемещаемость. Наличные могут передаваться другим пользователям.
6. Делимость. Заданная сумма электронных наличных может быть поделена на части меньшей суммы. (Конечно, общая сумма в конце должна сойтись.)

Ранее приведенные протоколы удовлетворяют требованиям 1, 2, 3 и 4, но не удовлетворяют требованиям 5 и 6. Ряд диалоговых систем электронных наличных удовлетворяет всем требованиям кроме 4 [318, 413, 1243]. Первая автономная система, удовлетворяющая требованиям 1, 2, 3 и 4, похожая на одну из описанных, была предложена [339]. Окамото и Охта предложили систему, удовлетворяющая требованиям с 1 по 5 [1209], они также предложили систему, удовлетворяющую требованиям с 1 по 6, объем данных для одного платежа составил приблизительно 200 мегабайт. Другая автономная система электронных монет с возможностью деления описана в [522].

Схема электронных наличных, предложенная теми же авторами [1211], удовлетворяет требованиям с1 по 6 без необходимости такого огромного объема данных. Общий объем данных для одного электронного платежа составляет около 20 килобайт, и протокол может быть выполнен за несколько секунд. Авторы рассматривают эту схему как первую идеальную систему неотслеживаемых электронных наличных.

Анонимные кредитные карточки

Этот протокол [988] для защиты личности клиента использует несколько различных банков. Каждый клиент имеет счет в двух различных банках. Первый банк, которому известна личность человека, может зачислять деньги на его счет. Второй банк знает клиента только под псевдонимом (подобно номерному счету в швейцарском банке).

Клиент может брать деньги из второго банка, доказывая, что он является владельцем счета. Но, этот банк не знает личности человека и не может зачислять деньги на его счет. Первый банк знает клиента и перечисляет деньги во второй банк, не зная псевдонима. Затем клиент анонимно тратит эти деньги. В конце месяца второй банк выставляет счет первому банку, веря, что он его оплатит. Первый банк передает счет клиенту, веря, что тот его оплатит. Когда клиент оплачивает счет, первый банк перечисляет дополнительные деньги во второй банк. Все транзакции проводятся через посредника, который действует подобно электронному Федеральному Резерву: оплачивает банковские счета, регистрирует сообщения и создает контрольный след.

Обмены между клиентом, продавцом и различными банками особо выделены в [988]. Если все не сговариваются против клиента, его анонимность гарантирована. Однако, это не электронные наличные, банк слишком легко может мошенничать. Протокол позволяет клиентам пользоваться преимуществами кредитных карточек, не раскрывая своей личности.